Received June 24, 2020, accepted July 15, 2020, date of publication July 28, 2020, date of current version August 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3012581

Double SHA-256 Hardware Architecture With Compact Message Expander for Bitcoin Mining

HOAI LUAN PHAM[1], (Graduate Student Member, IEEE), THI HONG TRAN[1], (Member, IEEE), TRI DUNG PHAN[1], VU TRUNG DUONG LE[2], DUC KHAI LAM[2], AND YASUHIKO NAKASHIMA[1], (Senior Member, IEEE)
[1]Graduation School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma 630-0192, Japan
[2]Computer Engineering Department, University of Information and Technology-Vietnam National University, Ho Chi Minh City 700000, Vietnam

Corresponding author: Thi Hong Tran (hong@is.naist.jp)

This work was supported by the Japan Science and Technology Agency (JST) under the Strategic Basic Research Programs PRESTO Grant number 2019A039.

ABSTRACT In the Bitcoin network, computing double SHA-256 values consumes most of the network energy. Therefore, reducing the power consumption and increasing the processing rate for the double SHA-256 algorithm is currently an important research trend. In this paper, we propose a high-data-rate low-power hardware architecture named the compact message expander (CME) double SHA-256. The CME double SHA-256 architecture combines resource sharing and fully unrolled datapath technologies to achieve both a high data rate and low power consumption. Notably, the CME algorithm utilizes the double SHA-256 input data characteristics to further reduce the hardware cost and power consumption. A review of the literature shows that the CME algorithm eliminates at least 9.68% of the 32-bit XOR gates, 16.49% of the 32-bit adders, and 16.79% of the registers required to calculate double SHA-256. We synthesized and laid out the CME double SHA-256 using CMOS 0.18 μm technology. The hardware cost of the synthesized circuit is approximately 13.88% less than that of the conventional approach. The chip layout size is 5.9 $mm \times 5.9$ mm, and the correctness of the circuit was verified on a real hardware platform (ZCU 102). The throughput of the proposed architecture is 61.44 Gbps on an ASIC with Rohm 180nm CMOS standard cell library and 340 Gbps on a FinFET FPGA 16nm Zynq UltraScale+ MPSoC ZCU102.

INDEX TERMS Bitcoin mining, SHA-256, unrolling, ASIC.

I. INTRODUCTION

Bitcoin is the most popular cryptocurrency and was invented by Satoshi Nakamoto in 2008 [1], [2]. Leveraging blockchain technology, Bitcoin uses a distributed public ledger to record all transactions without any third party [3]. Each block added to the public distributed ledger is created by hashing a 1024-bit message, including a version number, a hash of the previous block, a hash of the Merkle root, timestamp, target value, and a nonce. In the 1024-bit message, the nonce must be valid to create a hashing output smaller than the specified target value. Therefore, miners relentlessly seek valid nonces when adding new blocks. The process of finding a valid nonce is called Bitcoin mining [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

In Bitcoin mining, the double SHA-256 algorithm is used to compute the hash value of the bitcoin block header, which is a 1024-bit message. The use of double SHA-256 protects against the length extension attack [5]. Technically, SHA-256 consists of a message expander (ME) and a message compressor (MC). During the SHA-256 operation, the ME expands the 512-bit input message into 64 chunks of 32-bit data. The MC compresses these 64 32-bit data chunks into a 256-bit hashed output.

Most of the energy consumption required for maintaining the Bitcoin network stems from calculating double SHA-256 values. Therefore, reducing the hardware cost and energy consumption of the SHA-256 circuit is a popular research trend. In [6], the authors optimized the double SHA-256 operation for Bitcoin mining from an algorithmic perspective, but no hardware design was available to evaluate the power consumption. From a hardware perspective, [7]–[22] proposed solutions to improve SHA-256. For instance, the authors of [7] employed the carry-save adder to improve the computation time of the critical path, which increased the maximum frequency and processing rate, while [8]–[12] used pipeline technology to improve the SHA-256 throughput. A cache memory technique was presented in [13] to reuse data, minimize the critical paths, and reduce the number of memory accesses for SHA-256 processing. The authors of [14] adopted the unfolding technique to reduce the computing latency for SHA-256. The authors of [15] proposed using a 7-3-2 array compressor to reduce the critical path delay for SHA-256. The carry-save adders technique is used in [16] to reduce the latency of additions in the SHA-256 algorithm. The authors of [17] used a combination of techniques such as carry-save-adders and pipelines to increase the performance of SHA-256. Pipeline and unrolled techniques are presented in [18] and [19] to increase the throughput of SHA-256. The authors of [20]–[22] presented a SHA-256 implementation on an FPGA for performance evaluation, with no technique optimization. Despite providing improvements in terms of hardware cost and power consumption, the hardware circuits developed in [7]–[22] have low processing rates because they require several (up to 64) clock cycles to compute a single 256-bit hash value.

To be applicable for Bitcoin mining, a SHA-256 circuit needs not only efficient hardware and power cost but also a high processing rate. To reach a high processing rate, the authors in [23] proposed the fully unrolled SHA-256 datapath for Bitcoin mining hardware. Additionally, the fully unrolled SHA-256 datapath can be designed to run on an application-specific integrated circuit (ASIC) [24], which can reach even higher processing rates. However, because an ASIC implementation of a fully unrolled datapath has high power consumption and hardware costs, [25]–[28] proposed eliminating an 8-round unrolled datapath in the double SHA-256 architecture to reduce the chip area. Furthermore, several technical solutions, such as carry-save adders and optimized message compressor (MC) architectures have been proposed and applied to reduce the hardware and power costs.

In this study, we propose a new approach for reducing the hardware cost and power consumption of high processing rate fully unrolled SHA-256 architecture. We analyze the characteristics of the 1024-bit input data of double SHA-256 and propose compact message expander (CME) algorithms that significantly reduce the hardware cost required to compute the message expander (ME) process of SHA-256. In addition, we propose a CME double SHA-256 accelerator architecture that adopts the proposed CME algorithms to reduce the power consumption. Our architecture generates one 256-bit hash value per clock cycle. We implemented the proposed double SHA-256 accelerator architectures in ASIC CMOS 0.18 μm technology to demonstrate their energy efficiency. The Verilog code and synthesized results of the experiment are publicly available from GitHub.

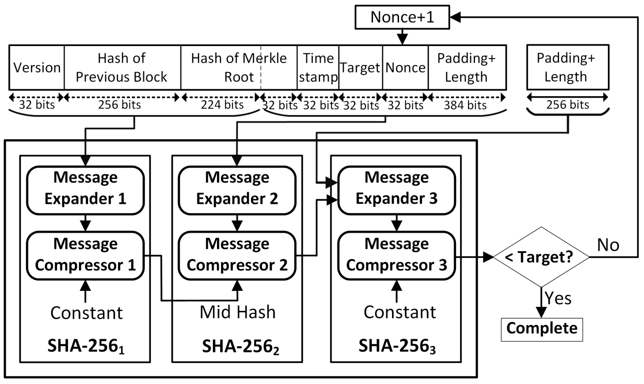


FIGURE 1. Overview architecture of double SHA-256 in Bitcoin Mining.

The remainder of this paper is organized as follows. Section II presents a preliminary study. Section III describes our proposed CME double SHA-256 architecture, and the CME algorithms and hardware circuits are explained in detail. Section IV reports our evaluation in terms of theory, ASIC, and FPGA experiments. Finally, Section V concludes the paper.

II. PRELIMINARIES

A. DOUBLE SHA-256 ARCHITECTURE FOR BITCOIN MINING

Fig. 1 shows the overview architecture of double SHA-256 applied for Bitcoin mining. The input to the double SHA-256 process is a 1024-bit message, which includes a 32-bit *version*, a 256-bit *hash of the previous block*, a 256-bit *hash of the Merkle root*, a 32-bit *timestamp*, a 32-bit *target*, a 32-bit *nonce*, and 384 bits of *padding*. The 1024-bit message is split into two 512-bit message parts; then SHA-256_1 calculates a hash value of the first 512-bit message, and SHA-256_2 computes a hash value of the final 512-bit message. Due to the double SHA-256 requirement, the 256-bit hash output from SHA-256_2 must be compressed into the final 256-bit hash by using SHA-256_3. In the Bitcoin mining process, the final 256-bit hash output from SHA-256_3 is compared to the target value. If the final hash is smaller than the target value, the valid 32-bit *nonce* is specified, and a new Bitcoin block is successfully created. Otherwise, the 32-bit *nonce* is increased by one and the double SHA-256 circuit recomputes to find a new hash value. This process is repeated until the 256-bit hash of SHA-256_3 meets the target requirement.

Computation inside all three blocks (SHA-256_1, SHA-256_2, and SHA-256_3) follows the SHA-256 algorithm, which has two processes: a message expander (ME) and a message compressor (MC).

Algorithm 1 shows the ME process, which expands the 512-bit input message into 64 chunks of 32-bit data W_j ($0 \leq j \leq 63$). In the first 16 rounds, the ME parses the 512-bit message into 16 32-bit data chunks (denoted as $W_j, j = 0$ to 15 where j is the round index). In the final 48 rounds, the ME calculates 48 chunks of 32-bit data W_j ($17 \leq j \leq 63$). Three 32-bit adders and two logical functions $\sigma_0(x)$ and $\sigma_1(x)$ are

Algorithm 1 Message Expander (ME)

```
• For $j$ from 0 to 15 {
    $W_j = M_j$ }
• For $j$ from 16 to 63 {
    $W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$ }
```

needed to compute each W_j ($17 \leq j \leq 63$) value. Fig. 2 shows the conventional circuit C required to calculate W_j ($17 \leq j \leq 63$), in which the logical functions $\sigma_0(x)$ and $\sigma_1(x)$ are respectively defined as follows:

$$\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x) \quad (1)$$
$$\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x) \quad (2)$$

Algorithm 2 shows the MC process, which compresses the 64 chunks of W_j ($0 \leq j \leq 63$) into a 256-bit hash value. The process involves three main steps: *initialization*, *loop*, and *add*. In the *initialization* step, eight internal hash values (denoted as a, b, c, d, e, f, g, h) are assigned to eight initial hashes $H_1, H_2, \ldots, H_8$ defined by the SHA-256 algorithm. In the *loop* step, the internal hash values a, b, c, d, e, f, g, h are calculated and updated through 64 loops. To compute a, b, c, d, e, f, g, h in each loop, logical functions such as $\Sigma_0(x)$, $\Sigma_1(x)$, $Ch(x, y, z)$, and $Maj(x, y, z)$ are used.

$$\Sigma_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \quad (3)$$
$$\Sigma_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x) \quad (4)$$
$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (5)$$
$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (6)$$

In the *add* step, the final hash is computed by adding the initial hashes $H_1, H_2, \ldots, H_8$ to the final internal hashes a, b, c, d, e, f, g, h resulting from the 64 loops.

Algorithm 2 Message Compressor (MC)

```
(1) Initialization:
    $a = H_1; b = H_2; c = H_3; d = H_4; e = H_5; f = H_6;$
    $g = H_7; h = H_8$
(2) Loop:
    For $j$ from 0 to 63 {
      • $T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_j + W_j$
      • $T_2 = \Sigma_0(a) + Maj(a, b, c)$
      • $h = g; g = f; f = e; e = d + T_1; d = c; c = b; b =$
        $a; a = T_1 + T_2$ }
(3) Add:
    $HO_1 = a + H_1; HO_2 = b + H_2; HO_3 = c + H_3;$
    $HO_4 = d + H_4; HO_5 = e + H_5; HO_6 = f + H_6;$
    $HO_7 = g + H_7; HO_8 = h + H_8;$
```

B. THE PROTOTYPE DOUBLE SHA-256 ARCHITECTURE

To be applicable for Bitcoin mining, double SHA-256 hardware should provide a high processing rate. The current optimal solution is to develop and implement a double SHA-256 accelerator in ASIC chips. In [23], the authors proposed an ASIC-based double SHA-256 accelerator that implemented *ME* and *MC* processes in a fully unrolled datapath for high processing. Technically, the fully unrolled SHA-256 datapath enables the 64 rounds of *ME* and *MC* to run in parallel and be pipelined.

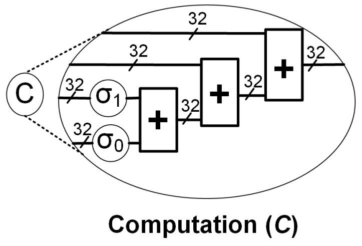


FIGURE 2. Conventional circuit *C* required for message expander (ME).

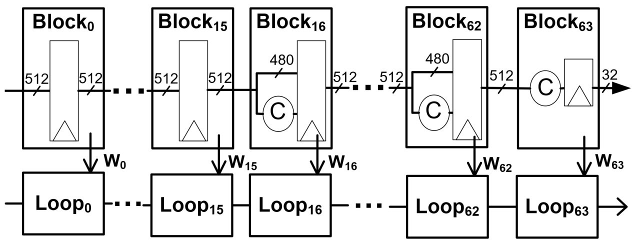


FIGURE 3. The Prototype 64-round unrolled datapath architecture for ME and MC processes of each SHA-256 circuit.

Fig. 3 illustrates a prototype SHA-256 architecture with 64-round unrolled datapaths for the *MC* and *ME* processes. The unrolled *ME* datapath is denoted as $Block_j$ ($j = 0, \ldots, 63$), while the unrolled *MC* datapath is denoted as $Loop_j$ ($j = 0, \ldots, 63$).

Because the goal of this study is to optimize the *ME* process, we focus specifically on a hardware implementation for *ME*. For the first 16 blocks (i.e., $Block_j$ ($j = 0, \ldots, 15$)), each ME block requires a 512-bit register (or 16 32-bit registers) to pipeline and store the 16 W_j ($j = 0, \ldots, 15$) values. For the last 48 blocks, i.e., $Block_j$ ($j = 16, \ldots, 63$), each block needs a 512-bit register (or 16 32-bit registers) and C circuits (Fig. 2) to compute W_j ($j = 16, \ldots, 63$). As shown in Fig. 1, the double SHA-256 accelerator for Bitcoin mining requires three individual SHA-256 circuits. This means that the accelerator must implement $48 \times 3 = 144$ C circuits (in the 16th to 63th blocks of SHA-256_1, SHA-256_2, and SHA-256_3). Thus, it is necessary to both optimize the C circuit and reduce the number of C circuits required for double SHA-256.

C. THE OPTIMIZED DOUBLE SHA-256 ARCHITECTURE

The prototype double SHA-256 accelerator has high power consumption because the fully unrolled datapath results in a large chip area. To reduce the power consumption, [25]–[28] proposed the optimized double SHA-256 accelerator, in which a 64-round unrolled datapath is optimized into a 60-round unrolled datapath.

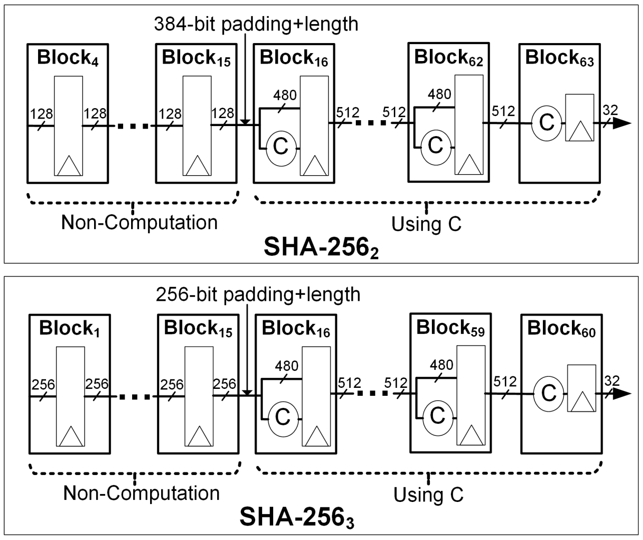


FIGURE 4. **The optimized 60-round unrolled datapath architecture for the ME process of SHA-256$_2$ and SHA-256$_3$.**

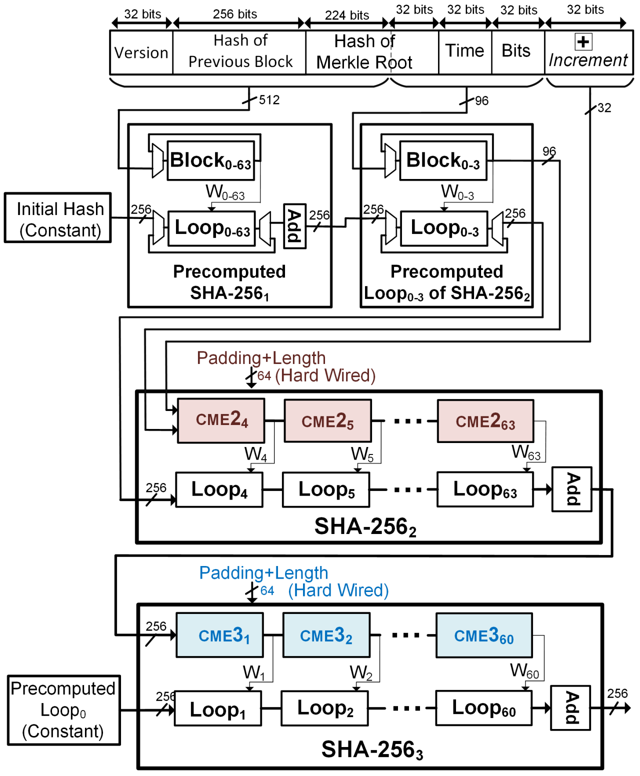


FIGURE 5. **Block diagram of the proposed CME double SHA-256 accelerator for Bitcoin mining.**

Fig. 4 shows a schematic diagram of the 60-round unrolled *ME* datapath used in SHA-256$_2$ and SHA-256$_3$. In SHA-256$_2$, the 60-round unrolled *ME* datapath includes rounds 4 to 63 (denoted as $Block_j$ ($j = 4, \ldots, 63$)). In SHA-256$_3$, the 60-round unrolled ME datapath includes rounds 1 to 60 (denoted as $Block_j$ ($j = 1, \ldots, 60$)). Consequently, 8 ME blocks are eliminated compared with the prototype architecture mentioned above.

III. THE PROPOSED CME DOUBLE SHA-256 ARCHITECTURE

A. ARCHITECTURAL OVERVIEW

In Bitcoin mining, the 512 bits of data input to SHA-256$_1$ does not change frequently because it does not include the 32-bit *nonce* field. Conversely, the 512 bits of data input to SHA-256$_2$ are updated frequently because of the changing value of the *nonce* field. Whenever the output of SHA-256$_2$ changes, SHA-256$_3$ also needs to be recomputed. Because the *nonce* field has 32 bits, each computation of SHA-256$_1$ requires SHA-256$_2$ and SHA-256$_3$ to recompute their values up to 2^{32} times.

Therefore, we propose the CME double SHA-256 accelerator architecture, as shown in Fig. 5. To achieve a high processing rate as well as efficient hardware and power cost, we implement a resource-sharing architecture for SHA-256$_1$ and a fully unrolled datapath architecture for SHA-256$_2$ and SHA-256$_3$. The SHA-256$_1$ has a single $Block_{0-63}$ circuit for calculating W_j ($j = 0, \ldots, 63$) and a single $Loop_{0-63}$ circuit for calculating the internal hashes a, b, c, d, e, f, h in 64 clock cycles. Each clock cycle computes one W_j value and updates the internal hash one time.

Similar to the conventional optimized double SHA-256 architecture, our SHA-256$_2$ has 60-round unrolled datapaths ($j = 4, \ldots, 63$), and our SHA-256$_3$ has 60-round unrolled datapaths ($j = 1, \ldots, 60$). To reduce the hardware and power costs of SHA-256$_2$ and SHA-256$_3$, we propose using CME algorithms and their equivalent hardware circuits. In Fig. 5, the CME for SHA-256$_2$ is denoted as CME2$_j$ ($j = 4, \ldots, 63$), and the CME for SHA-256$_3$ is denoted as CME3$_j$ ($j = 1, \ldots, 60$).

Using pipelined and parallel operations, SHA-256$_2$ and SHA-256$_3$ can produce an output hash every clock cycle. However, the resource-sharing SHA-256$_1$ circuit produces one hash value every 64 clock cycles. The low processing rate of the SHA-256$_1$ circuit does not affect the final processing rate of the CME double SHA-256 accelerator because one SHA-256$_1$ output value can be used to calculate SHA-256$_2$ and SHA-256$_3$ up to 2^{32} times. The final processing rate of the CME-based double SHA-256 is one 256-bit hash value per clock cycle.

In the following subsections, we explain our proposed CME algorithms and the equivalent hardware designs.

B. COMPACT MESSAGE EXPANDER (CME) ALGORITHM

We propose the CME algorithms by analyzing the characteristics of the input data of SHA-256$_2$ and SHA-256$_3$.

1) CME FOR SHA-256$_2$

As seen in Fig. 1, the 512 bits of data input to SHA-256$_2$ include a 32-bit *Merkle root hash*, a 32-bit *time stamp*, a 32-bit *target*, a 32-bit *nonce*, and a 384-bit *padding+length*

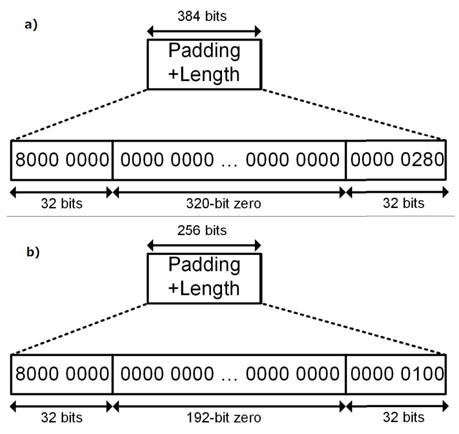


FIGURE 6. **Contents of the padding+length field for SHA-256$_2$ (a), and SHA-256$_3$ (b).**

field. It is worth noting that most of the content of the *padding+length* field consists of zeros (refer to Fig. 6a).

Assume that the 512 bits of data are separated into 16 32-bit words M_j ($j = 0, \ldots, 15$). The CME operation for SHA-256$_2$ is illustrated in Algorithm 3. The algorithm processes the data in 64 loops. During the first 16 loops, W_j ($j = 0, \ldots, 15$) are assigned to M_j ($j = 0, \ldots, 15$). The values of W_j ($j = 5, \ldots, 14$) are all zero because they are equivalent to the zero values of the *padding+length* field. In addition, W_4 and W_{15} are constants. During the last 48 loops, the CME calculates W_j ($j = 16, \ldots, 63$) by using (7):

$$W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}. \qquad (7)$$

The logical functions $\sigma_0(x)$ and $\sigma_1(x)$ are shown in (1) and (2), respectively.

Utilizing the zeros or constant values of W_j ($j = 4, \ldots, 15$), we can optimize the calculation of (7). For example, the W_{16} calculation can be analyzed as follows:

$$\begin{aligned} W_{16} &= \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0 \\ &= 0 + 0 + \sigma_0(W_1) + W_0 \\ &= \sigma_0(W_1) + W_0 \end{aligned} \qquad (8)$$

Note that $W_{14} = 0$ and $W_9 = 0$. By comparing (7) with (8) for calculating W_{16}, it can be seen that the logical function $\sigma_1(x)$ and two 32-bit adders have been eliminated.

The computations of W_j($j = 17, \ldots, 63$) are analyzed and optimized similarly. The final results are shown in Algorithm 3.

2) CME FOR SHA-256$_3$

The 512 bits of input data to SHA-256$_3$ include the 256-bit hash output from SHA-256$_2$ concatenated with a 256-bit *padding+length* field. The value of the first 32 bits of padding is $32'h80000000$, while the value of the last 32 bits *padding+length* is $32'h00000100$. The remaining values are all zeros (refer to Fig. 6b).

We divide the 512-bit input data into 16 32-bit words M_j ($j = 0, \ldots, 15$). The CME operation for SHA-256$_3$ is illustrated in Algorithm 4. It processes the data in 64 loops. In the first 16 loops, W_j ($j = 0, \ldots, 15$) are assigned to M_j ($j = 0, \ldots, 15$). The values of W_j ($j = 9, \ldots, 14$) are all zero because they are equivalent to the zero values of the *padding+length* field. In addition, W_8 and W_{15} are constants. In the last 48 loops, CME calculates W_j ($j = 16, \ldots, 63$) using (7).

Utilizing the zero or constant characteristics of W_j ($j = 8, \ldots, 15$), we optimize the calculation of (7) for calculating W_j($j = 16, \ldots 63$). The final results are shown in Algorithm 4.

Utilizing Algorithms 3 and 4, we can significantly reduce the number of 32-bit adders and the number of logical func-

Algorithm 3 Compact Message Expander in SHA-256$_2$

```
• For j from 0 to 3 {
      $W_j = M_j$ }
• $W_4$ = 32'h80000000
• For j from 5 to 14 {
      $W_j$ = 32'h00000000 }
• $W_{15}$ = 32'h00000280
• $W_{16} = \sigma_0(W_1) + W_0$
• For j from 17 to 19 {
      $W_j = \sigma_1(W_{j-2}) + \sigma_0(W_{j-15}) + W_{j-16}$ }
• $W_{20} = \sigma_1(W_{18}) + W_4$
• $W_{21} = \sigma_1(W_{19})$
• For j from 22 to 29 {
      $W_j = \sigma_1(W_{j-2}) + W_{j-7}$ }
• $W_{30} = \sigma_1(W_{28}) + W_{23} + \sigma_0(W_{15})$
• For j from 31 to 63 {
      $W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$ }
```

Algorithm 4 Compact Message Expander in SHA-256$_3$

```
• For j from 0 to 7 {
      $W_j = M_j^{(i)}$ }
• $W_8$ = 32'h80000000
• For j from 9 to 14 {
      $W_j$ = 32'h00000000 }
• $W_{15}$ = 32'h00000100
• $W_{16} = \sigma_0(W_1) + W_0$
• For j from 17 to 21 {
      $W_j = \sigma_1(W_{j-2}) + \sigma_0(W_{j-15}) + W_{j-16}$ }
• For j from 22 to 23 {
      $W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$ }
• $W_{24} = \sigma_1(W_{22}) + W_{17} + W_8$
• For j from 25 to 29 {
      $W_j = \sigma_1(W_{j-2}) + W_{j-7}$ }
• $W_{30} = \sigma_1(W_{28}) + W_{23} + \sigma_0(W_{15})$
• For j from 31 to 63 {
      $W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$ }
```

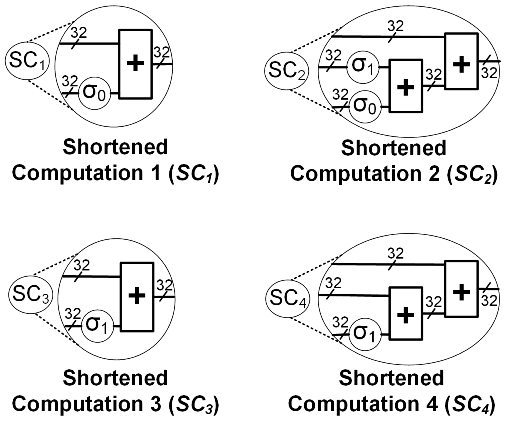

Shortened Computation 1 (SC1)
Shortened Computation 2 (SC2)
Shortened Computation 3 (SC3)
Shortened Computation 4 (SC4)

FIGURE 7. The proposed shortened computation circuits: SC_1, SC_2, SC_3, and SC_4 for the CME process.

tions $\sigma_0(x)$ and $\sigma_1(x)$ required to calculate W_{16} to W_{63} in SHA-256_2 and SHA-256_3.

C. CME HARDWARE CIRCUITS

From Algorithm 3 and 4, we propose four types of shortened computation (*SC*) circuits as shown in Fig. 7. Compared with the traditional *C* circuit shown in Fig. 2, the proposed SC_1 eliminates two 32-bit adders and the logical function $\sigma_1(x)$; SC_2 eliminates one 32-bit adder; SC_3 eliminates two 32-bit adders and the logical function $\sigma_1(x)$; and SC_4 eliminates one 32-bit adder and the logical function $\sigma_0(x)$. Note that eliminating either $\sigma_0(x)$ or $\sigma_1(x)$ also eliminates two 32-bit rotations, one 32-bit shift, and two 32-bit XOR circuits.

Based on the *C* circuit shown in Fig. 2 and the four types of *SC* circuits shown in Fig. 7, we develop hardware architectures for the CME processes of SHA-256_2 and SHA-256_3 as shown in Fig. 8 and Fig. 10, respectively.

The proposed CME circuit for SHA-256_2 (Fig. 8) is divided into three phases. Phase 1 includes $CME2_4$ to $CME2_{19}$. Each operation requires a 128-bit register (or four 32-bit registers) to store and pipeline W_0 to W_3. In phase 1, instead of using the conventional *C* circuit in Fig. 2, the SC_1 and SC_2 circuits in Fig. 7 are implemented to reduce hardware costs. Phase 2 includes $CME2_{20}$ to $CME2_{30}$, for which the SC_2 and SC_3 circuits are appropriately implemented (refer to algorithm 3). Phase 3 includes $CME2_{31}$ to $CME2_{63}$, and the *C* circuit is implemented in all the blocks of this phase.

The three phases are classified based on the characteristics of the datapath bit width. In phase 1, the datapath bit-width is constant (128 bits). The 384-bits of W_4 to W_{15} are fixed constants. Hence, phase 1 do not need to store and pipeline W_4 to W_{15}. In phase 2, W_{20} to W_{30} must be stored and pipelined. Thus, the datapath bit-width in phase 2 is appropriately increased from 160 bits to 480 bits. In phase 3, the datapath bit-width of $CME2_{31}$ to $CME2_{57}$ is 512 bits without optimization. To eliminate unnecessary values of W_j in subsequent blocks, the datapath bit-width of $CME2_{57}$ to $CME2_{63}$ appropriately reduces from 480 bits to 32 bits. To understand the reason for the datapath bit-width adjustment,

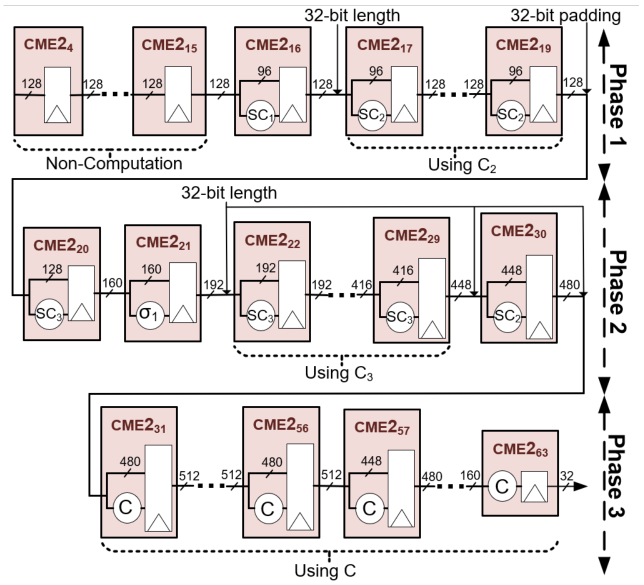


FIGURE 8. Block diagram of the 60-round unrolled datapath CME2 process for SHA-256_2.

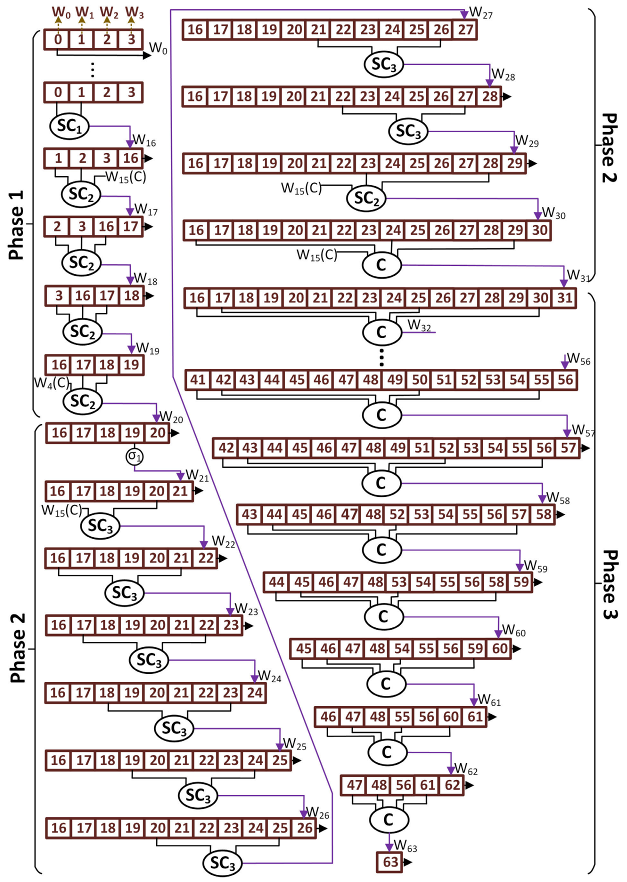


FIGURE 9. Detailed computational circuit of the CME2 process for SHA-256_2.

we show the detailed data flow and computational circuit of the CME2 process in Fig. 9. In this figure, the number represents the *j* index of W_j. For example, we need four 32-bit

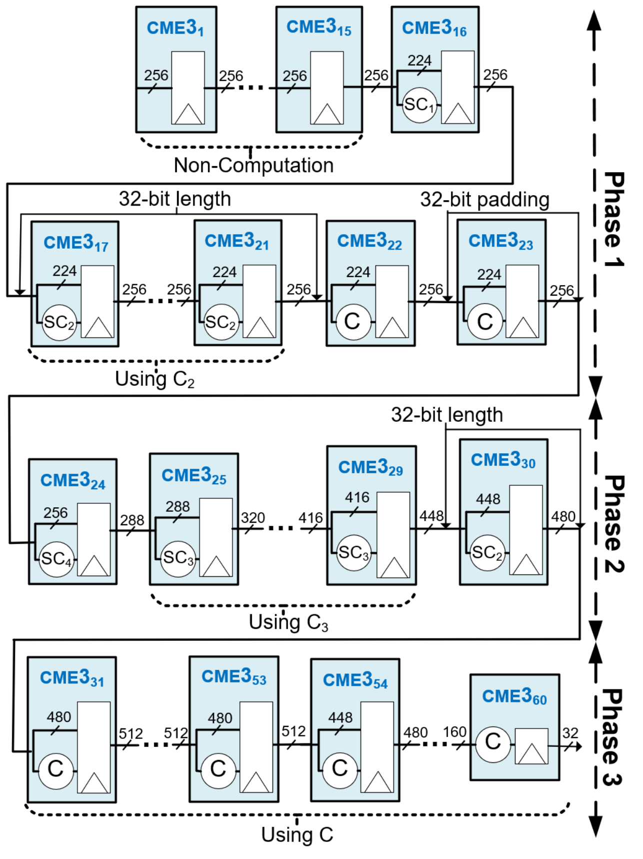


FIGURE 10. Block diagram of the 60-round unrolled datapath CME3 process for SHA-256$_3$.

registers (equivalent to 128 bits) to store W_0 to W_3 in blocks CME2$_4$ to CME2$_{15}$. As another example, CME2$_{32}$ needs sixteen 32-bit registers ($16 \times 32 = 512$ bits) to pipeline store 16 values of W_j (j=16, 17,...,31), which are required for the calculation of its following blocks.

Similarly, the proposed CME circuit for SHA-256$_3$ has three phases (Fig. 10). Phase 1 includes CME3$_1$ to CME3$_{23}$. Because of the zero and constant property of input data W_8 to W_{15}, all blocks of phase 1 have the same datapath of 256 bits only (which is required to pipeline store eight 32-bit values W_0 to W_7). A large number of registers are thus eliminated. In this phase, circuits SC$_1$, SC$_2$, and C are appropriately implemented (refer to algorithm 4). Phase 2 includes blocks from CME3$_{24}$ to CME3$_{30}$. Circuits SC$_4$, SC$_3$, and SC$_2$ are appropriately implemented (refer to algorithm 4). Phase 3 includes blocks from CME3$_{31}$ to CME3$_{60}$. We do not implement blocks from CME3$_{61}$ to CME3$_{63}$ because we can detect early whether the final hash is smaller than the target value without waiting for results from CME3$_{61}$ to CME3$_{63}$. Circuit *C* is implemented in all blocks.

Three phases are classified based on the characteristics of the datapath bit-width. In phase 1, the datapath bit-width is constant (256 bits).The 256-bits of W_8 to W_{15} are fixed constants and do not need to be stored and pipelined in phase 1. In phase 2, W_{24} to W_{30} must be stored and pipelined. Therefore, the datapath bit-width of CME3$_{24}$ to CME3$_{30}$ is

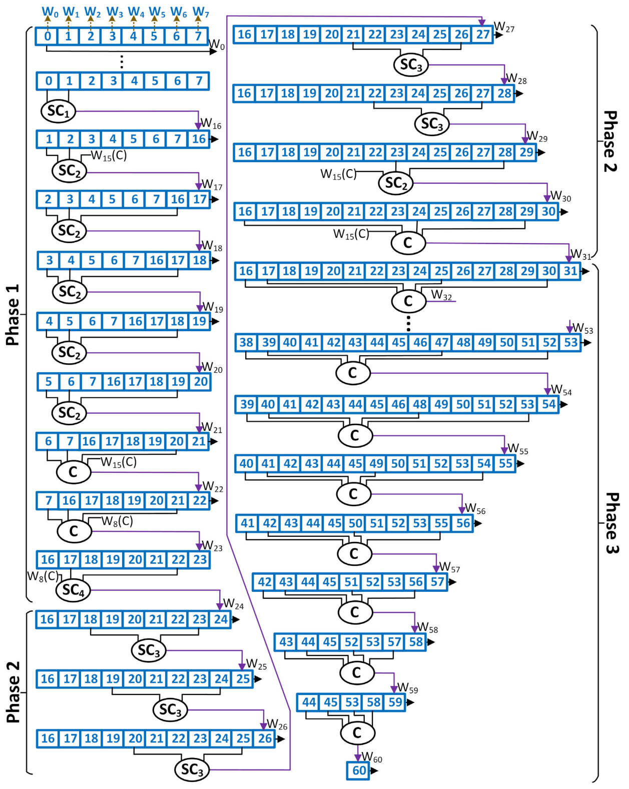


FIGURE 11. Detailed computational circuit of the CME3 process for SHA-256$_3$.

appropriately increased from 288 bits to 480 bits. In phase 3, the datapath bit-width of CME3$_{31}$ to CME3 $_{53}$ is 512 bits without optimization. The datapath bit-width of CME3$_{54}$ to CME3 $_{60}$ is reduced from 480 bits to 32 bits. To prove that the datapath bit-width adjustment is appropriate, we show the detailed data flow and the computational circuit of the CME3 process in Fig. 11. In this figure, the number represents the *j*-th index of W_j. For example, each block from CME3$_0$ to CME2$_{15}$ requires eight 32-bit registers (equivalent to $8 \times 32 = 256$ bits) to store W_0 to W_7. These values are required to calculate the blocks from CME3$_{16}$ to CME2$_{22}$. As another example, block CME3$_{59}$ requires five 32-bit registers ($5 \times 32 = 160$ bits) to store W_{44}, W_{45}, W_{53}, W_{58}, and W_{59}, which are required for the CME3$_{60}$ calculation.

IV. EVALUATION

In this section, we evaluate the efficiency of the CME method when it is applied in the CME double SHA-256 accelerator. We evaluate the performance from three aspects: theory, ASIC, and FPGA experimental results.

A. THEORETICAL REVIEW

For comparison purposes, we developed three hardware circuits, all of which follow the architecture proposed in Fig. 5. The three circuits differ only in how they implement the ME processes of SHA-256$_2$ and SHA-256$_3$. The first circuit (named Prototype double SHA-256) was proposed in [23] and mentioned in section II-B. The second circuit (named

TABLE 1. Theory comparison: hardware-resource required for ME process.

Stage	Resource	Architecture		
		Prototype	Optimized	**Proposed**
SHA-256_2	32-bit adders	144	144	**117**
	32-bit XOR gates	192	192	**170**
	32-bit Rotation	240	240	**217**
	32-bit Shift	48	48	**38**
	32-bit registers	1009	801	**651**
SHA-256_3	32-bit adders	144	135	**116**
	32-bit XOR gates	192	180	**166**
	32-bit Rotation	240	225	**210**
	32-bit Shift	48	45	**39**
	32-bit registers	1009	825	**697**
SHA-256_2 + SHA-256_3	32-bit adders	288	279	**233**
	32-bit XOR gates	384	372	**336**
	32-bit Rotation	480	465	**427**
	32-bit Shift	96	93	**77**
	32-bit registers	2018	1620	**1348**

Optimized double SHA-256) was proposed in [25]–[27], and [28], and is mentioned in Section II-C. The last circuit is our proposed CME double SHA-256.

Table 1 shows the theoretical hardware resources required by the three architectures in terms of the number of adders, XOR gates, rotations, shifts, and registers. In Table 1, SHA-256_2 and SHA-256_3 are the evaluation targets because they are the most hardware-intensive parts.

Compared to the prototype and optimized architectures, the proposed architecture respectively decreases the total number of 32-bit adders by approximately 19.1% and 16.49%, the total number of 32-bit XOR gates by approximately 12.5% and 9.68%, and the total number of 32-bit rotation operations, by approximately 11% and 8.17%.

In addition, the proposed architecture reduces the total number of 32-bit shift operations by approximately 19.8% and 17.2% compared to the prototype and optimized architectures, respectively.

Notably, the proposed architecture eliminates 33.2% and 16.79% of the total number of registers compared to the prototype and optimized architectures, respectively.

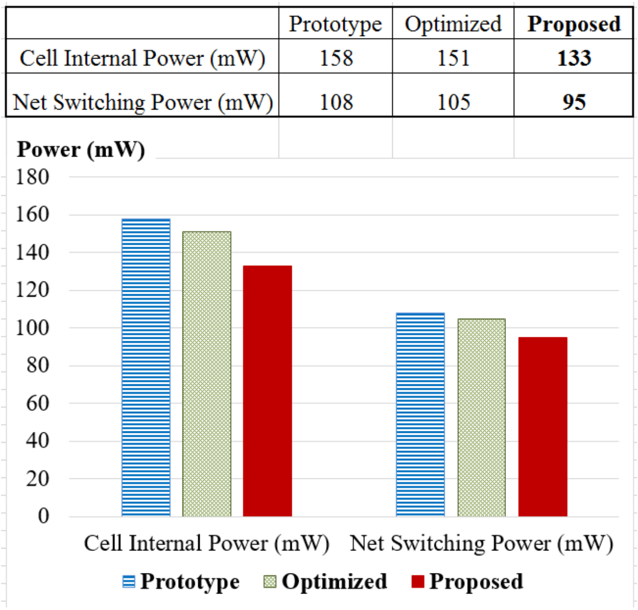
B. ASIC EXPERIMENT

1) AREA AND POWER APPROACH

To ensure a fair comparison, the three double SHA-256 circuits were coded in Verilog and synthesized in an ASIC using the Synopsys Design Compiler with the Rohm 0.18μm CMOS standard cell library [29]. Table 2 shows the synthesized area of the three architectures. Note that the total area is the sum of the combinational and non-combinational area (registers), as well as other types of circuits, including buff/Inv, wires, etc. The total area of the proposed

TABLE 2. Practical comparison: The ASIC synthesized area of three *double SHA-256* architectures.

Design	Combinational Area (μm^2)	Non-combinational Area (μm^2)	Total Area (μm^2)
Prototype	5,741,967	6,716,928	13,823,243
Optimized	5,480,520	6,428,772	13,224,169
Proposed	**4,876,523**	**5,409,863**	**11,388,986**



	Prototype	Optimized	**Proposed**
Cell Internal Power (mW)	158	151	**133**
Net Switching Power (mW)	108	105	**95**

FIGURE 12. The ASIC synthesis power of the prototype, the optimized, and the proposed double SHA-256 accelerators.

CME double SHA-256 is smaller by 17.6% and 13.9% compared to the prototype and optimized architectures, respectively.

Fig. 12 summarizes the energy consumption of the three architectures obtained from the ASIC synthesis results. In terms of cell internal power, the proposed double SHA-256 circuit consumes 133 mW, which is a reduction of 15.82% and 11.92% compared to the prototype and optimized architectures, respectively. In terms of net switching power, the proposed CME double SHA-256 circuit consumes 95 mW, which constitutes reductions of 12.04% and 9.52% compared to prototype and optimized architectures, respectively. These energy consumption reductions are due to the smaller hardware circuit, which matches our expectations.

Based on the timing report of ASIC synthesis, the maximum frequency of the three architectures is 60 MHz. This means that the architectures achieve throughput of 1024 bits $\times$ 60 MHz = 61.44 Gbps.

In addition, we successfully laid out the proposed CME double SHA-256 circuit in ASIC technology with the Rohm 0.18μm CMOS standard cell library. Fig. 13 shows the chip layout, and Fig. 14 shows the chip energy distribution map. The size of the chip layout is 5.9 mm $\times$ 5.9 mm.

TABLE 3. A comparison of ASIC synthesis results.

Technology	Design	Frequency (MHz)	No. of Cycles Single SHA-256	No. of Cycles Double SHA-256	Throughput R_d (Gbps)	Area (μm^2)	Hardware Efficiency E_d (Kbps/μm^2)
SMIC 0.18μm	[15]	208	65	195	1.092	211,955	5.16
UMC 0.18μm	[16]	302	66	198	1.562	185,256	8.44
TSMC 0.18μm	[17]	380	65	195	1.995	562,704	3.54
	Prop.	**92**	**-**	**1**	**94.208**	**9,038,148**	**10.42**

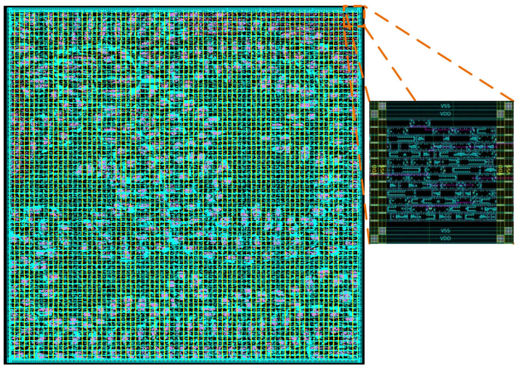


FIGURE 13. Post-layout circuit of the proposed CME double SHA-256 accelerator.

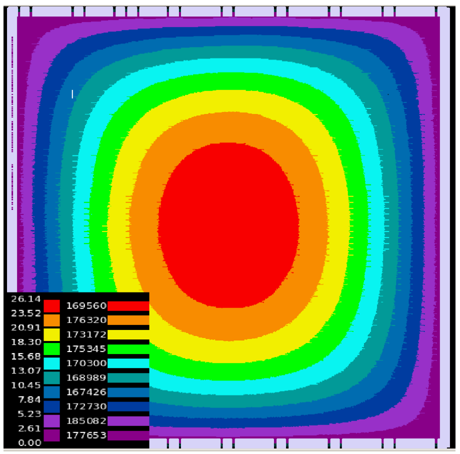


FIGURE 14. Energy distribution map of the post-layout *CME double SHA-256* circuit.

2) PROCESSING RATE AND HARDWARE EFFICIENCY APPROACH

In this experiment, we prove that the ASIC design of our proposed CME double SHA-256 architecture outperforms previous works in terms of processing rate and hardware efficiency. To ensure a fair comparison, we also synthesized our architecture in ASIC TSMC 0.18μm technology using the CMOS standard cell library. We then compare our results with the previous works in [15], [16], and [17].

The comparison is shown in Table 3. It is worth noting that the designs of [15], [16], and [17] are single SHA-256 circuits. To be applied to Bitcoin mining, these circuits must repeat their calculations three times to generate a double SHA-256 hash value from the 1024-bit input message. The number of cycles required to compute the double SHA-256 (denoted by C_d) is thus triple the number of cycles required to compute a single SHA-256 (denoted by C_s); refer to (9).

$$C_d = 3 \times C_s \tag{9}$$

Then, we calculate the processing rate for double SHA-256 (R_d) by using (10). The *BlockSize* is 1024 bits.

$$R_d = \frac{BlockSize \times Frequency}{C_d} \tag{10}$$

From the R_d and area results, the hardware efficiency for double SHA-256 (denoted by E_d) is computed by (11).

$$E_d = \frac{R_d}{Area} \tag{11}$$

Table 3 summarizes the synthesized area results, the calculated processing rate, and the hardware efficiency. The processing rate (R_d) and hardware efficiency (E_d) of our proposed architecture are significantly improved compared to those of the works in [15], [17], and [16]. The numerical results are as follows.

In terms of processing rate (R_d), our CME double SHA-256 architecture is faster than the designs proposed in [15], [16], and [17] by 86, 60, and 47 times, respectively.

In terms of hardware efficiency (E_d), our CME double SHA-256 architecture improves the efficiency by 102%, 23%, and 194% compared to the designs in [15], [16], and [17], respectively.

3) FPGA SYNTHESIS RESULTS

To ensure a fair comparison with other existing SHA-256 architectures, such as [18]–[21], [22], and [23], we synthesized the proposed CME double SHA-256 circuit on four Xilinx FPGA boards, including Kintex UltraScale (XCKU5P-ffva676-3-e), Virtex 7(XC7VX1140T-FLG 1926-2), Artix 7 (XC7A200T-FBG484-1), and Zynq UltraScale+ ZCU102 (XCZU9EG-FFVB1156-2-e).

The results are shown in Table 4. It is worth noting that the existing architectures in [18]–[22] and [23] are single SHA-256 architectures that must repeat the computation three times to generate a double SHA-256 hash value for Bitcoin mining. Thus, the number of clock cycles required

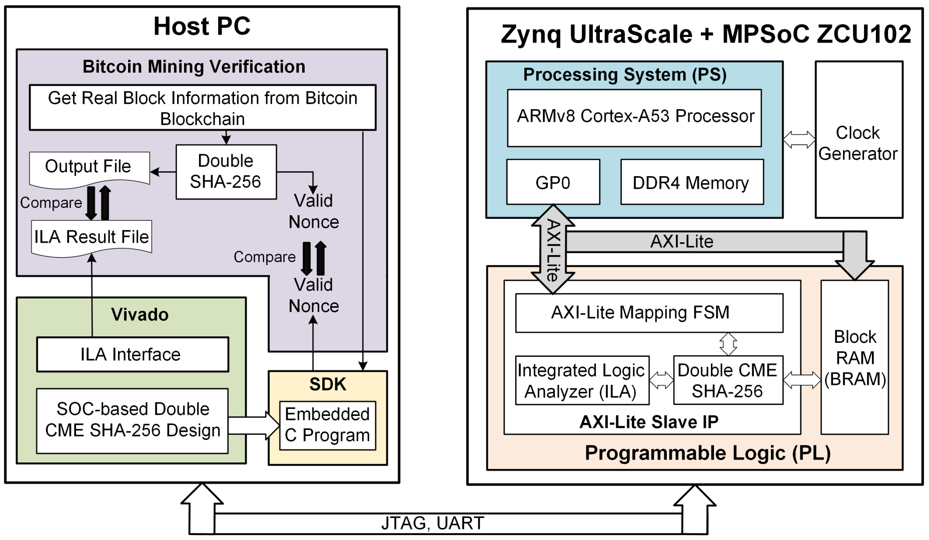


FIGURE 15. Diagram of the SoC-based CME double SHA-256 system for implementation and verification.

TABLE 4. A comparison of FPGA synthesis results.

FPGA Device	Design	Frequency (MHz)	No. of Cycles Single SHA-256	No. of Cycles Double SHA-256	Throughput R_d (Gbps)	Area LUTs	Area FFs	Hardware Efficiency E_d (Mbps/LUT)
Kintex Ultrascale+ XCKU5P	[18]	364.3	16	48	7.772	9,316	4,188	0.83
	[19]	377.9	16	48	8.062	4,986	4,312	1.62
	[23]	428.1	-	1	438.374	49,144	54,674	8.92
	Prop.	**428.1**	**-**	**1**	**438.374**	**46,053**	**52,428**	**9.52**
Virtex 7 XC7VX1140T	[20]	367	65	195	1.928	350	-	5.5
	[21]	196.1	65	195	1.03	1,505	-	0.68
	[23]	263.2	-	1	269.516	49,104	54,674	5.48
	Prop.	**263.2**	**-**	**1**	**269.516**	**46,013**	**52,428**	**5.86**
Artix 7 XC7A200T	[22]	174	65	195	0.914	1,359	1,618	0.68
	[23]	132.5	-	1	135.168	49,104	54,674	2.76
	Prop.	**132.5**	**-**	**1**	**135.168**	**46,013**	**52,428**	**2.94**
Zynq Ultrascale+ ZCU102	[23]	374.2	-	1	383.18	49,145	54674	7.8
	Prop.	**374.2**	**-**	**1**	**383.18**	**46,013**	**52,428**	**8.32**

to compute a double SHA-256 is tripled. We focus on evaluating the hardware efficiency (Mbps/LUT) of the single and double SHA-256 architectures in this subsection. In general, the proposed CME double SHA-256 outperforms the existing SHA-256 architectures in terms of hardware efficiency. The numerical results are as follows.

On the Kintex UltraScale FPGA, the hardware efficiency (Mbps/LUT) of the proposed CME double SHA-256 architecture is enhanced by 1,047% (9.52 vs. 0.83), 488% (9.52 vs. 1.62), and 7% (9.52 vs. 8.92) compared to the hardware efficiencies of the architectures in [18], [19], and [23], respectively.

On the Virtex 7 FPGA, the hardware efficiency of the proposed architecture is enhanced by 7% (5.86 vs. 5.5), 762% (5.86 vs. 0.68), and 7% (5.86 vs. 5.48) compared to the the hardware efficiencies of the architectures in [20], [21], and [23], respectively.

On the Artix 7 FPGA, the hardware efficiency of the proposed architecture is enhanced by 332% (2.94 vs. 0.68) and 7% (2.94 vs. 2.76) compared to the the hardware efficiencies of the architectures in [22] and [23], respectively.

On Zynq UltraScale+ ZCU102 FPGA, the hardware efficiency of the proposed architecture is enhanced by 7% (8.32 vs. 7.8) compared to the hardware efficiency of the architecture in [23].

C. FPGA EXPERIMENT

1) FUNCTIONAL VERIFICATION ON A REAL SoC HARDWARE PLATFORM

To prove that the circuit operates correctly not only in the software simulation tool but also on real hardware, we built a System on Chip (SoC) platform to execute the proposed CME double SHA-256 circuit. The SoC platform overview is shown in Fig. 15.

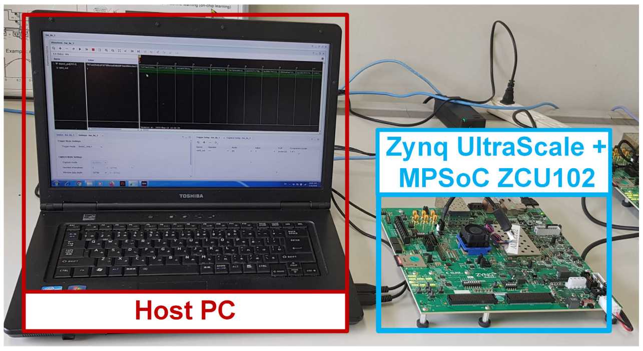


FIGURE 16. Real-world experiments with the SoC-based CME double SHA-256 devices.

TABLE 5. Execution time of double SHA-256 on different hardware platforms.

Device	Execution time (millisecond)				
	100,000 outputs	200,000 outputs	300,000 outputs	400,000 outputs	500,000 outputs
CPU Core i7-6950X@3.50GHz (10 cores), Memory128GB	150	310	460	620	770
CPU XEON6144x2 (16cores) + V100, Memory 256GB	140	290	380	560	740
GPU NVIDIA Tesla V100-PCIE, Memory 16GB	27	64	82	110	140
FPGA Zynq UltraScale+ MPSoC ZCU102 (333MHz clock input)	0.3	0.6	0.9	1.2	1.5

The platform includes two primary components: a host PC and a Zynq UltraScale+ ZCU102 evaluation board. The host PC exchanges data with the ZCU102 board via JTAG and UART cables.

The ZCU102 board includes an ARMv8 microprocessor, a programmable logic (PL), and a clock generator. Our developed circuit, CME double SHA-256, is embedded in the PL of ZCU102. The PL also has block ram (BRAM) and an integrated logic analyzer (ILA). We used the BRAM to store the valid nonce value for Bitcoin mining and ILA to monitor the outputs of the CME double SHA-256 circuit. The maximum operating frequency of the ZCU102 board is 333 MHz.

The host PC consists of a Vivado, a Software Development Kit (SDK), and a Bitcoin Mining Verification (BMV) program. Vivado is a software suite for SoC development. We use the Vivado suite to design and load the SoC-based system onto the Zynq UltraScale+ ZCU102 board. Moreover, the Vivado helps to export the outputs of the CME double SHA-256 circuit in the ZCU102 into an ILA result file for verification by the BMV program. The SDK is intended for the development of embedded software applications for SoC systems. We use the SDK to embed the real block information from the Bitcoin blockchain network onto our SoC-based system. The BMV is a C-code program that verifies the correctness of the embedded CME double SHA-256 circuit. The BMV executes a double SHA-256 on the host PC and compares the results with the outputs of the CME double SHA-256 circuit.

TABLE 6. Hash rate and power consumption of several SHA-256 architectures.

Research	Device	Hash rate (MHash/s)	Power (W)
[30]	CPU Intel I7-2600K@ 3.3GHz, 4 Cores	1.9	-
	GPU NVIDIA GeForce GTX 550 TI	20.4	-
[31]	CPU Intel Core i7-990x@ 3.46GHz, 6 Cores	33	-
	GPU NVIDIA GeForce GTX 570	155	-
[32]	CPU Intel Core i7 i7-950@ 3.06GHz	18.9	150
	GPU ATI Radeon HD 5770	214.5	108
	ASIC Block Erupter Sapphire	333	2.55
This work	**FPGA Zynq UltraScale+ ZCU102 (333MHz)**	**333**	**2.49**

The abovementioned SoC system has been used to thoroughly verify the correctness of the CME double SHA-256 circuit at different operating frequencies, such as 333 MHz (maximum frequency) and 200 MHz. All the cases result in 100% accuracy, which proves that the proposed CME double SHA-256 architecture works correctly in a real hardware platform. The maximum processing rate of the circuit on the ZCU102 board is 333 MHash/s (or 333 MHz $\times$ 1024 bit/CLK $=$ 340.992 Gbps).

Fig. 16 shows an image of the SoC evaluation platform, which includes a host PC (Toshiba Satellite B652 / G Core i5 3320M 2.6GHz / 4GB) and the UltraScale+ ZCU102 evaluation board.

2) PROCESSING-RATE EVALUATION ON A REAL HARDWARE PLATFORM

In this subsection, we evaluate the processing rate and power consumption of the proposed CME double SHA-256 on real hardware platform ZCU102 to prove that our architecture outperforms other high-performance platforms, including CPUs, GPUs, and the existing SHA-256 architectures.

Table 5 shows the execution time of the double SHA-256 algorithm on several hardware platforms, including a CPU, GPU, and FPGA. To compute the same number of hashes (e.g., 500.000 hashes) the proposed architecture running on the FPGA ZCU102 requires only 1.5 ms, while the CPU i7-6950X, CPU XEON 6144, and GPU Tesla V100 require 770 ms, 740 ms, and 140 ms, respectively, which means that the proposed architecture reduces the execution time by 513 times, 493 times, and 93 times, respectively.

Table 6 summarizes the hash rate and power consumption from several studies that reported double SHA-256 results. As the table shows, the hash rate of our proposed architecture running on an FPGA is significantly higher than those of the works in [30] and [31]. Although [32] was executed on an ASIC and our architecture was executed on an FPGA, our architecture still achieves the same hash rate but consumes less power.

V. CONCLUSION

Bitcoin mining is an important process in keeping the Bitcoin network secure; however, it consumes massive amounts of energy. To reduce the power consumption and increase the processing rate of the Bitcoin mining process, we proposed a CME double SHA-256 hardware circuit in this paper. The architecture includes three SHA-256 circuits in which the first circuit (SHA-256_1) is a resource-sharing architecture while the last two circuits (SHA-256_2 and SHA-256_3) are fully unrolled datapath architectures. The combination of these two types of architecture results in a high processing rate but low hardware costs. Specifically, we propose several compact message expander (CME) algorithms and associated hardware architectures to further reduce the power consumption and hardware costs. Our proposed circuit generates one 256-bit hash value per clock cycle. We thoroughly verified and evaluated the proposed circuit on both ASIC and FPGA platforms. The experimental results showed that the proposed circuit outperforms other high-performance CPU and GPU platforms for computing double SHA-256 values. The proposed circuit also outperforms existing works with specific hardware circuits for computing the double SHA-256 values. The double SHA-256 circuit was laid out on the ASIC with Rohm 0.18 μm CMOS standard cell library, resulting in a chip size of 5.9 $mm \times 5.9$ mm and the throughput of 61.44 Gbps. The circuit is also proven to work correctly in a real hardware platform (ZCU102), achieving a processing rate of 340.992 Gbps.

Blockchain is not only the Bitcoin network. Blockchain technology is outgrowing in its potential to be applied in many fields of life, such as smart health care, autonomous cars, and supply chains. Other blockchain networks may employ not only SHA-256 but also other cryptography hash functions, such as SHA-512 or SHA-3. Therefore, developing a flexible and programmable accelerator that can compute several hash functions is a future need. By developing a low-cost low-power-consumption blockchain accelerator, we help to enhance the security and decentralized features of the blockchain network. Therefore, we believe that developing a blockchain accelerator that can compute multiple cryptography hash functions at low cost and with low power consumption will be an important research trend in the near future.

APPENDIX

The Verilog code and the synthesized results of the prototype, optimized, and proposed architectures can be found at https://github.com/archlab-naist/Double-CME-SHA256/

REFERENCES

[1] P. D. DeVries, "An analysis of cryptocurrency, bitcoin, and the future," *Int. J. Bus. Manage. Commerce*, vol. 1, no. 2, pp. 1–9, 2016.

[2] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. Manubot, Nov. 2019.

[3] M. Rahouti, K. Xiong, and N. Ghani, "Bitcoin concepts, threats, and machine-learning security solutions," *IEEE Access*, vol. 6, pp. 67189–67205, 2018.

[4] H. Vranken, "Sustainability of bitcoin and blockchains," *Current Opinion Environ. Sustainability*, vol. 28, pp. 1–9, Oct. 2017.

[5] J. Taskinsoy, "Bitcoin and turkey: A good match or a perfect storm," *SSRN Electron. J.*, Oct. 2019, doi: 10.2139/ssrn.3477849.

[6] N. T. Courtois, M. Grajek, and R. Naik, "Optimizing SHA256 in bitcoin mining," in *Proc. Int. Conf. Cryptogr. Secur. Syst.*, Berlin, Germany: Springer, 2014, pp. 131–144.

[7] X. Zhang and H. Hu, "Optimization of hash function implementation for bitcoin mining," in *Proc. 3rd Int. Conf. Mechatronics Eng. Inf. Technol. (ICMEIT)*, 2019.

[8] M. D. Rote, V. N, and D. Selvakumar, "High performance SHA-2 core using the round pipelined technique," in *Proc. IEEE Int. Conf. Electron., Comput. Commun. Technol. (CONECCT)*, Jul. 2015, pp. 1–6.

[9] L. Dadda, M. Macchetti, and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Feb. 2004, pp. 70–75.

[10] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," in *Proc. 7th Int. Symp. Embedded Comput. Syst. Design (ISED)*, Dec. 2017, pp. 1–4.

[11] X. Zhang, R. Wu, M. Wang, and L. Wang, "A high-performance parallel computation hardware architecture in ASIC of SHA-256 hash," in *Proc. 21st Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2019, pp. 52–55.

[12] S. B. Suhaili and T. Watanabe, "Design of high-throughput SHA-256 hash function based on FPGA," in *Proc. 6th Int. Conf. Electr. Eng. Informat. (ICEEI)*, Nov. 2017, pp. 1–6.

[13] R. García, I. Algredo-Badillo, M. Morales-Sandoval, C. Feregrino-Uribe, and R. Cumplido, "A compact FPGA-based processor for the secure hash algorithm SHA-256," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 194–202, Jan. 2014.

[14] H. Michail, G. Athanasiou, A. Kritikakou, C. Goutis, A. Gregoriades, and V. Papadopoulou, "Ultra high speed SHA-256 hashing cryptographic module for ipsec hardware/software codesign," in *Proc. Int. Conf. Secur. Cryptogr. (SECRYPT)*, Jul. 2010, pp. 1–5.

[15] L. Bai and S. Li, "VLSI implementation of high-speed SHA-256," in *Proc. IEEE 8th Int. Conf. ASIC*, Oct. 2009, pp. 131–134, doi: 10.1109/ASICON.2009.5351591.

[16] S. Tillich, M. Feldhofer, M. Kirschbaum, P. Thomas, S. Jörn-Marc, and S. Alexander, "High-speed hardware implementations of BLAKE, blue midnight wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein," *IACR Cryptol. ePrint Arch.*, vol. 510, 2009.

[17] F. Opritoiu, S. L. Jurj, and M. Vladutiu, "Technological solutions for throughput improvement of a secure hash algorithm-256 engine," in *Proc. IEEE 23rd Int. Symp. Design Technol. Electron. Packag. (SIITME)*, Oct. 2017, pp. 159–164.

[18] R. Martino, and A. Cilardo, "A flexible framework for exploring, evaluating, and comparing SHA-2 designs," *IEEE Access*, vol. 7, pp. 72443–72456, 2019.

[19] R. Martino, and A. Cilardo, "A configurable implementation of the SHA-256 hash function," in *Proc. Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.*, Cham, Switzerland: Springer, Nov. 2019, pp. 558–567.

[20] F. Kahri, B. Bouallegue, M. Machhout, and R. Tourki, "An FPGA implementation and comparison of the SHA-256 and blake-256," in *Proc. 14th Int. Conf. Sci. Techn. Autom. Control Comput. Eng. STA*, Dec. 2013, pp. 152–157, doi: 10.1109/STA.2013.6783122.

[21] R. Florin and R. Ionut, "FPGA based architecture for securing IoT with blockchain," in *Proc. Int. Conf. Speech Technol. Hum.-Comput. Dialogue (SpeD)*, Oct. 2019, pp. 1–8, doi: 10.1109/SPED.2019.8906595.

[22] D. K. N. and R. Bhakthavatchalu, "Parameterizable FPGA implementation of SHA-256 using blockchain concept," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2019, pp. 0370–0374, doi: 10.1109/ICCSP.2019.8698069.
[23] J. Barkatullah and T. Hanke, "Goldstrike 1: CoinTerra's first-generation cryptocurrency mining processor for bitcoin," *IEEE Micro*, vol. 35, no. 2, pp. 68–76, Mar. 2015, doi: 10.1109/MM.2015.13.
[24] M. Vilim, H. Duwe, and R. Kumar, "Approximate bitcoin mining," in *Proc. 53rd Annu. Design Autom. Conf. DAC*, Jun. 2016, pp. 1–6.
[25] V. B. Suresh, S. K. Satpathy, and S. K. Mathew, "Energy-efficient bitcoin mining hardware accelerators," U.S. Patent 10 313 108 B2, Jun. 4, 2019.
[26] V. B. Suresh, S. K. Satpathy, and S. K. Mathew, "Optimized SHA-256 datapath for energy-efficient high-performance Bitcoin mining," U.S. Patent 1 014 098 B2, Nov. 27, 2018.
[27] V. B. Suresh, S. K. Satpathy, and S. K. Mathew, "Bitcoin mining hardware accelerator with optimized message digest and message scheduler datapath," U.S. Patent 2 018 008 642 A1, Mar. 29, 2018.
[28] V. Suresh, S. Satpathy, R. Kumar, M. Anders, H. Kaul, A. Agarwal, S. Hsu, R. Krishnamurthy, V. De, and S. Mathew, "A 250Mv, 0.063J/Ghash bitcoin mining engine in 14nm CMOS featuring dual-vcc Sha256 datapath and 3-Phase latch based clocking," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C32–C33.
[29] T. H. Tran, S. Kanagawa, D. P. Nguyen, and Y. Nakashima, "ASIC design of MUL-RED Radix-2 pipeline FFT circuit for 802.11ah system," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS XIX)*, Apr. 2016, pp. 1–3.
[30] J. Anish Dev, "Bitcoin mining acceleration and performance quantification," in *Proc. IEEE 27th Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2014, pp. 1–6, doi: 10.1109/CCECE.2014.6900989.
[31] M. Bedford Taylor, "The evolution of bitcoin hardware," *Computer*, vol. 50, no. 9, pp. 58–66, 2017, doi: 10.1109/MC.2017.3571056.
[32] S. Ghimire and H. Selvaraj, "A survey on bitcoin cryptocurrency and its mining," in *Proc. 26th Int. Conf. Syst. Eng. (ICSEng)*, Dec. 2018, pp. 1–6, doi: 10.1109/ICSENG.2018.8638208.

HOAI LUAN PHAM (Graduate Student Member, IEEE) received the bachelor's degree in computer engineering from the University of Information Technology-Vietnam National University Ho Chi Minh City (VNU-HCM), Vietnam, in 2018. He is currently pursuing the M.S. degree with the Nara Institute of Science and Technology (NAIST), Japan. His research interests include blockchain technology and cryptography.

THI HONG TRAN (Member, IEEE) received the bachelor's degree in physics and the master's degree in microelectronics from the Vietnam National University-Ho Chi Minh University of Science (VNU-HCMUS), Vietnam, in 2008 and 2012, respectively, and the Ph.D. degree in information science from the Kyushu Institute of Technology, Japan, in 2014. Since 2015, she has been with the Nara Institute of Science and Technology (NAIST), Japan, as a Full Time Assistant Professor. Her research interests include digital hardware circuit design, algorithm relate to wireless communication, communication security, blockchain technologies, SHA-2, SHA-3, cryptography, and so on. She is a Regular Member of the IEEE, IEICE, REV-JEC, and so on.

TRI DUNG PHAN received the Engineer degree in computer engineering (hardware design) from Vietnam National University Ho Chi Minh City-University of Information Technology (VNUHCM-UIT), in 2019. He is currently pursuing the M.S. degree in Nara Institute of Science and Technology (NAIST), Japan. His research interests include secure hash algorithm (SHA) in Hardware Design, such as FPGA and ASIC design.

VU TRUNG DUONG LE received the bachelor's degree in IC and hardware design from the Vietnam National University Ho Chi Minh City-University of Information Technology, in 2020. His research interests include blockchain technologies, deep learning, cryptography, and so on.

DUC KHAI LAM received the B.E. and M.S. degrees from the University of Science, Vietnam National University Ho Chi Minh City (VNU-HCM), in 2006 and 2011, respectively, and the Ph.D. degree from the Kyushu Institute of Technology, Japan, in 2016. He is currently with the University of Information Technology, VNU-HCM, as a Lecturer and a Researcher. His research interests include wireless communication systems, digital signal processing, ASIC, and VLSI design.

YASUHIKO NAKASHIMA (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer engineering from Kyoto University, in 1986, 1988, and 1998, respectively. He was a Computer Architect with the Computer and System Architecture Department, Fujitsu Ltd., from 1988 to 1999. From 1999 to 2005, he was an Associate Professor with the Graduate School of Economics, Kyoto University. Since 2006, he has been a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include computer architecture, emulation, circuit design, and accelerators. He is a Fellow of IEICE, a Senior Member of IPSJ, and a member of the IEEE CS and ACM.

BCA: A 530-mW Multicore Blockchain Accelerator for Power-Constrained Devices in Securing Decentralized Networks

Thi Hong Tran, *Member, IEEE*, Hoai Luan Pham, *Member, IEEE*, Tri Dung Phan, and Yasuhiko Nakashima, *Senior Member, IEEE*

***Abstract*— Blockchain distributed ledger technology (DLT) has widespread applications in society 5.0 because it improves service efficiency and significantly reduces labor costs. However, employing blockchain DLT entails considerable energy consumption in the mining process. This paper proposes a blockchain accelerator (BCA) with ultralow power consumption and a high processing rate to address the problem. The BCA focuses on accelerating the double secure hash algorithm (SHA) 256 function required in the mining process at a system-on-chip (SoC) level. We propose three ideas, namely, multiple local memories (multimem), double-cell processing element (D-PE), and nonce autoupdate (NAU), to reduce the external data transfer time and improve the BCA hardware efficiency. We propose a cascaded multiple BCA chip model to enhance the system throughput by several-fold. Our experiments on an ASIC and FPGA prove that the proposed BCA successfully performs the mining process for multiple blockchain networks with much lower power consumption than that of the state-of-the-art CPUs and GPUs. The BCA is laid out with Renesas 65 nm technology with a chip area of 25 mm^2 and consumes 530 mW at 100 MHz. The power efficiency of the layout chip is improved by 2428 and 143 times compared with that of the fastest CPU Intel i9-10940X and GPU RTX 3090, respectively.**

***Index Terms*— Blockchain, Bitcoin mining, double SHA-256, accelerator, embedded system, SoC, society 5.0.**

I. Introduction

SUPER Smart Society 5.0 is Japan's concept of a technology-based, human-centered society that balances economic development with the resolution of social problems, such as an aging society, greenhouse gas emissions, and labor shortages [1]. In society 5.0, developing and securing a shared cyberspace for storing the data collected from physical space is vital. The shared cyberspace allows many services and systems to share the same data resources to cooperate smartly and automatically. As a result, the system service efficiency and labor cost are expected to improve significantly. Recently, while centralized cyber-physical systems (CPSs) have faced many types of security threats, decentralized blockchain (BC) CPSs have become an important research theme for securing shared cyberspace.

BC has proven successful in securing decentralized cryptocurrency systems such as Bitcoin, Ripple, and Ethereum. Recently, it has received considerable research attention to benefit a wide variety of life aspects, such as self-driving cars, smart healthcare systems, robotics, supply chains, global trade, insurance, financial markets, logistics, and autonomous military weapons [2]–[4]. This is why Fukao stated in his book [5] that BC is a once-in-500-years revolution. In a BC network, data are recorded on a chain of blocks known as ledgers [6]. Every block added to the ledger is created by hashing a message header, which may include a hash value of the previous block, hash of the Merkle root of the data, timestamp, target value, and nonce. For the sake of network security, the addition of a new block is a hard-working process known as mining in which miners may relentlessly find a valid *nonce* to make a hashing output smaller than the target value [7]. The double secure hash algorithm (SHA)-256 is the most popular algorithm to compute the hash of the block header. The use of double SHA-256 protects the network against the length extension attack [8]. The Bitcoin network has been reported to repeatedly compute the double SHA-256 function approximately 10^{37} times on average to successfully add a new block to the ledger. This results in Bitcoin's well-known disadvantage of extremely high energy consumption. Improving the processing rate and hardware efficiency of SHA-256 and/or double SHA-256 circuits has thus been a research trend in recent years.

Conventional studies have proposed many techniques to improve the performance of SHA-256 and/or double SHA-256. For instance, [9] developed an FPGA-based compact SHA-256 processor using resource sharing. Pipelined architectures have been developed in many studies, such as [10], [14], to reduce the critical path delay of the SHA-256 circuit. Unrolled architectures [15]–[18] were implemented to achieve a high processing rate but a large circuit area and high power consumption. [19]–[22] further reduced the circuit area of the double SHA-256 unrolled architecture by eliminating an 8-round unrolled datapath. Furthermore, [11] introduced

Manuscript received April 13, 2021; revised June 25, 2021, July 18, 2021, and August 2, 2021; accepted August 2, 2021. Date of publication August 17, 2021; date of current version September 30, 2021. This work was supported by Japan Science and Technology Agency (JST) through Precursory Research for Embryonic Science and Technology (PRESTO) under Grant 2020A031. This article was recommended by Associate Editor M. Mozaffari Kermani. *(Corresponding author: Thi Hong Tran.)*

The authors are with the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma 630-0192, Japan (e-mail: hong@is.naist.jp).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2021.3102618.

Digital Object Identifier 10.1109/TCSI.2021.3102618

the parallel counter technique to reduce the critical path delay and circuit area. Recently, [23] proposed an SHA-256 round computation rescheduling method to reduce the critical path delay and improve the processing rate of the SHA-256 accelerator. Patent [24] used the clock gate technique to selectively inactivate some part of the double SHA-256. In addition, the datapath of double SHA-256 in the case of Bitcoin mining is optimized for high-efficiency energy consumption and hardware costs. In patent [25], the message digestion and message scheduler datapaths of double SHA-256 in the case of Bitcoin mining are optimized for low hardware cost. Recently, [26] and [27] utilized the input data characteristic of double SHA-256 in the case of Bitcoin mining to significantly optimize the circuit area and power consumption. Overall, these works focused on improving the power and hardware cost performance of SHA-256 and/or double SHA-256 as a standalone accelerator. Unfortunately, the double SHA-256 circuit is not able to work alone; it must work under the control of a microprocessor of an embedded system. Improving the performance of double SHA-256 as a standalone core, as the aforementioned conventional works have done, is insufficient to render the core applicable in real situations. Therefore, general purpose hardware platforms, such as multicore Intel central processing units (CPUs) and high-performance GPUs (Tesla V100), are still of interest for performing the mining process of several BC networks, such as Bitcoin and Bitcoin Cash, although the energy consumption of these platforms is extremely high. The use of these platforms results in the well-known issue of high electricity consumption. As of November 2020, Bitcoin energy consumption was estimated to be approximately 76.87 terawatt hours per year, which is higher than the electricity consumption of many countries, such as the Czech Republic and Switzerland. Obviously, developing an ultralow power consumption, high processing rate accelerator that is compatible with an embedded system for application in BC mining has been an urgent matter in recent years. The system-on-chip (SoC) compatibility of the accelerator and data transfer rate between the accelerator and external double data random access memory (DDRAM) must be carefully considered.

Motivated by Global Sustainable Development Goal (SDG) No. 7 (clean energy), our aim is to develop a sustainable blockchain accelerator (BCA) for power-constrained edge devices, such as autonomous cars, smartphones, smart watches, and sensors to enable these devices to contribute to securing decentralized BC networks (see Fig. 1). With the BCA, our aim is not only to solve the aforementioned energy consumption issue but also to enhance the security and decentralization of the networks. The current blockchain consensus, such as proof of work, obviously allows a large mining pool to achieve high performance, which results in the centralized-like nature and degradation of network security. In the future, we believe that there should be more efficient blockchain consensuses that degrade the performance of large mining pools and encourage numerous power-constrained devices to participate in the mining task to enhance the network security and decentralized level. We developed the BCA to achieve this aim. In [28], we developed an SoC-compatible SHA-256 accelerator with a high computation rate and efficient data transfer rate. However, the developed accelerator attempts to compute the SHA-256 hash function for general purposes. There are still three main problems when applying the accelerator in [28] for computing double SHA-256. First, multiple processing elements (PEs) are required to compute the double SHA-256 algorithm. Thus, the processing rate is degraded. Second, the size of local memories is too large, while the number of trials on hash calculation per configuration time is limited. Third, the architecture in [28] requires the data transfer between the accelerator and the external memory for all hash functions to be calculated by the accelerator, which raises the data transfer rate bottleneck issue. This issue obviates the usefulness of further improving the processing rate of the accelerator.

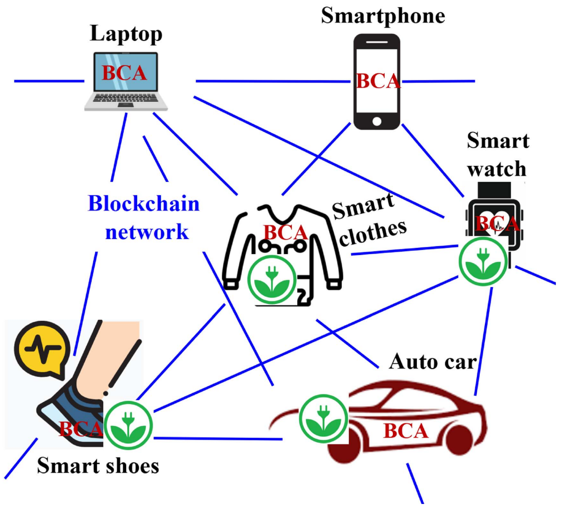


Fig. 1. Future vision of the BCA in helping power-constrained devices secure blockchain networks.

In this study, we solve the aforementioned problems in [28] by developing a novel BCA architecture for accelerating the double SHA-256 in the BC mining process. The name "BCA" is used owing to its purpose of accelerating the mining process of blockchain networks, even though its function is to compute double SHA-256. In terms of innovation, we introduce the idea of a double-cell processing element (D-PE) to compute the double SHA-256 algorithm inside a single D-PE. The D-PEs work in parallel as multiple independent processing cores (multicore). We modify the multiple local memory structure in [28] to significantly reduce the memory size. In addition, we implement the NAU mechanism to eliminate the data transfer rate bottleneck issue. Our experiments on Xilinx FPGA Alveo U280 and ASIC Renesas Silicon On Thin Buried Oxide (SOTB) 65 nm show that the BCA successfully performs the mining task for as many as 14 BC networks, including Bitcoin and Bitcoin Cash, with much lower power consumption than that of the state-of-the-art CPU and GPU platforms. The layout chip of the BCA consumes 530 mW on ASIC 65 nm and is expected to consume only 65 mW on ASIC 8 nm technology.

The remainder of this paper is structured as follows. Section II presents the background of this research. Section III

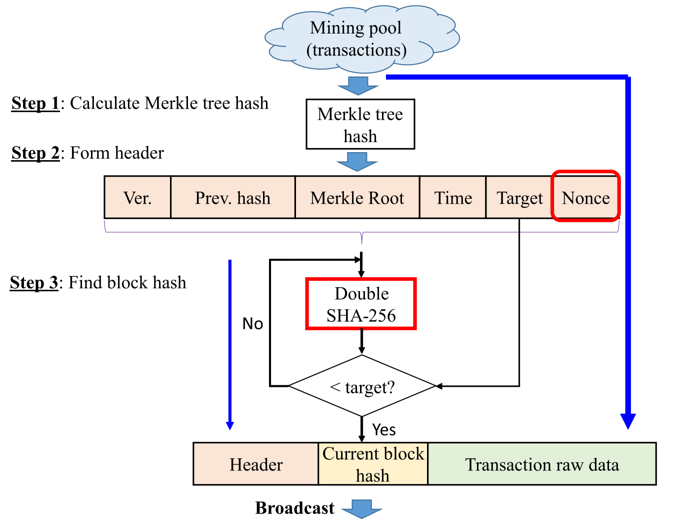


Fig. 2. Three steps for creating a new block in the ledger in the mining process of blockchain networks.

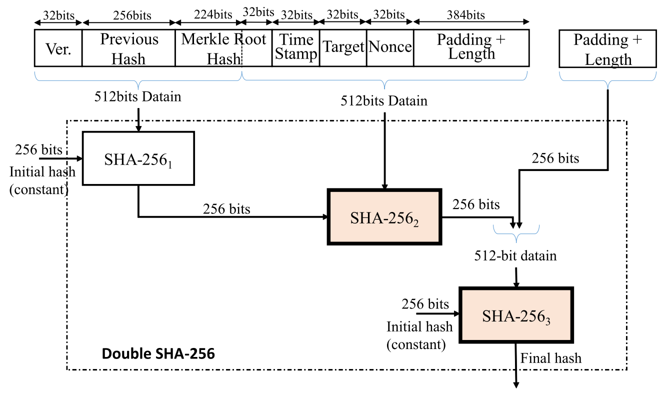


Fig. 3. Double SHA-256 computation for a block header.

describes our proposed BCA architecture. Section IV reports the BCA verification and evaluation in both ASIC and FPGA experiments. Finally, section V concludes the paper.

II. Background

A. Proof of Work Consensus and Double SHA-256

One of the greatest challenges of a decentralized BC network is to guarantee that all nodes of the network maintain the same integrity ledger although nodes may be honest or dishonest. To achieve this purpose, several consensus protocols have been proposed, such as proof of work (PoW), proof of stake (PoS), and delegated proof of stake (DPoS). Among them, PoW is widely belied to be the only consensus that is truly secure enough for a public network. In a PoW-based BC network, an extensive hash computing effort is applied to trade off system security. In essence, miners on the network perform three steps to add a new block to the ledger, as shown in Fig. 2. Step 1 selects a set of transaction data that the miner plans to write into the current block and calculates the Merkle tree hash value of this set of data. Step 2 forms the block header, including the version, hash value of the previous block, Merkle root hash value, timestamp, target, and nonce fields. Step 3 is the most time- and energy-consuming process. This step hashes the block header by utilizing an SHA. Double SHA-256 has been the most popular algorithm in recent years. In this step, a large number of trials are conducted by computing the double SHA-256 of the header with all possible values of the *nonce* field until a valid hash smaller than a predefined target is found.

Fig. 3 shows the computation of double SHA-256 for a block header in many popular BC networks, such as Bitcoin and Bitcoin Cash. Before being hashed, to the header will be added *padding+length* to a length of exactly 1024 bits (2 blocks of 512-bit data). Two blocks of SHA-256, denoted SHA-256$_1$ and SHA-256$_2$, are needed to hash the 1024-bit header. SHA-256$_1$ hashes the first 512 bits, and SHA-256$_2$ hashes the last 512 bits of the header. SHA-256$_2$ requires the result of SHA-256$_1$ as its initial hash value. For the sake of security, another SHA-256 block (SHA-256$_3$) is required to hash the hash result of the header (*double hash*). To obtain the double hash value of the header, a new 512-bit data block must be created by concatenating the 256-bit hash result of SHA-256$_2$ and 256-bit *padding+length* data. SHA-256$_3$ then hashes this data block.

Note that once the value of the final double hash is larger than the target, the double SHA-256 computation is performed again by changing the value of the 32-bit *nonce* field. *nonce* is the only field of the header that will be changed. This means that the miner may need to compute SHA$_2$ and SHA$_3$ up to 2^{32} times for each calculation of SHA-256$_1$. Therefore, our aim is to develop a hardware circuit to accelerate SHA$_2$ and SHA$_3$ with the assumption that SHA$_1$ has been computed by the microprocessor of the embedded system.

B. SHA-256 Algorithm

This section briefly describes the key points of the SHA-256 algorithm that are worth reviewing for our proposed architecture. SHA-256 receives 512 bits of input message and 256 bits of initial hash to compute 256 bits of message hash. The input message and initial hash are divided into 16 32-bit data words ($W_0, \ldots, W_{15}$) and 8 32-bit data words ($Hi_0, \ldots, Hi_7$). The message expander (ME) and message compressor (MC) processes are then applied to compute 8 32-bit words of message hash ($Ho_0, \ldots, Ho_7$).

The ME process follows eqs. (1) to (3) to compute W_j ($16 \leq j \leq 63$).

$$\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x) \tag{1}$$
$$\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x) \tag{2}$$
$$W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16} \tag{3}$$

Note that $S^n(.)$ and $R^n(.)$ represent right rotation and right shift of a word by n bits, respectively.

The MC process includes two main steps: *loops* and *hash updates*. In the *loop* step, eight loop hash words a, b, c, d, e, f, g, h are first assigned to the initial hash words ($Hi_0, \ldots, Hi_7$). Then, these loop hash words a, b, c, d, e, f, g, h are computed and updated in 64 loops.

Loop j $(0 \leq j \leq 63)$ computes equations (4) to (8).

$$T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_j + W_j \tag{4}$$
$$T_2 = \Sigma_0(a) + Maj(a, b, c) \tag{5}$$
$$a = T_1 + T_2 \tag{6}$$
$$e = d + T_1 \tag{7}$$
$$b = a; \quad c = b; \;\; d = c; \;\; f = e; \;\; g = f; \;\; h = g \tag{8}$$

where logical functions such as $\Sigma_0(x)$, $\Sigma_1(x)$, $Ch(x, y, z)$, and $Maj(x, y, z)$ are calculated by the following equations.

$$\Sigma_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \tag{9}$$
$$\Sigma_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x) \tag{10}$$
$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \tag{11}$$
$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \tag{12}$$

In the *hash update* step, the message hash words $Ho_0, \ldots, Ho_7$ are found by obtaining the sum of the initial hashes $Hi_0, \ldots, H_7$ and the loop hashes a, b, c, d, e, f, g, h, as illustrated in eq. (13).

$$Ho_0 = Hi_0 + a; \ldots; Ho_7 = Hi_7 + h \tag{13}$$

C. Preliminary BCA Architecture Idea

According to eq. (3), there are data dependencies among loops of the hash function. For example, the computation of W_j in the ME process in loop j $(16 \leq j \leq 63)$ requires data from the previous loops, such as W_{j-2} in loop $j-2$, W_{j-7} in loop $j-7$, W_{j-15} in loop $j-15$, and W_{j-16} in loop $j-16$. This data dependence increases the need for data transfer between the calculation unit and memory in every loop, which is the main reason for the low efficiency of the general hardware platforms (e.g., Intel multicore CPUs, GPU GTX 3090) in computing SHA-256. In the CPU and GPU platforms, memory resources such as caches L1 and L2 are placed far from the calculation unit, which results in a long data transfer time. Therefore, the processing rate is significantly degraded even though the CPUs and GPUs are rich in hardware resources.

In this study, we develop an ultralow power, high processing rate BCA by eliminating the weakness of the CPU and GPU platforms from our architecture. In detail, we employ the following ideas. **Idea 1: Multiple local memories.** We propose the use of multiple local memories near the calculation unit inside a PE to reduce the data transfer time and improve the hardware efficiency. **Idea 2: D-PE architecture.** If a PE can compute only some but not all loops of the hash function, then the calculation unit inside a PE may need to access data from the local memory of other PEs. This results in the requirement of complicated mesh wire connections among PEs. The circuit area and power consumption for the data bus and data access among PEs may constitute a large portion of the total chip. To avoid this situation, we propose a D-PE architecture that computes all loops of SHA-256$_2$ and SHA-256$_3$ of the double SHA-256 inside a single D-PE, which allows all D-PEs to operate in parallel and independently without requiring any wire connection

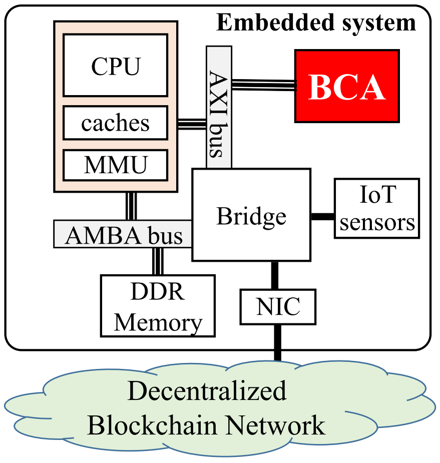


Fig. 4. Overview of the embedded system-on-chip (SoC).

among D-PEs. **Idea 3: NAU and valid hash checker (VHC) mechanisms**. The computing of 1 hash function requires transferring at least 1024 bits of data (which includes 512-bit input messages, 256-bit initial hashes, and 256-bit hash results) via an advanced extensible interface (AXI) bus. This amount is multiplied if a large number of hash functions should be computed per configuration time. The data transfer rate thus becomes a bottleneck that degrades the processing rate of the entire system. An improvement in the accelerator processing rate may become meaningless if the embedded system is not able to bear such a high data transfer rate. In this study, we eliminate the data transfer rate bottleneck issue by employing the NAU mechanism to automatically update the *nonce* value and the VHC mechanism to check and discard the invalid hash values. Only the valid hash value will be informed and read out of the BCA. In addition, we minimize the size of local memories to ensure enough space for computing only one hash function. These memories should be reused for the BCA to uninterruptedly compute an unlimited number of hash functions until a valid hash value is found. **Idea 4: Cascaded multiple chip model.** After eliminating the data transfer rate bottleneck issue, we further enhance the BCA processing rate multiple times by developing a cascaded multiple BCA chip model. The model allows multiple BCA chips to cascade to the same AXI bus. The total processing rate and power consumption of the cascaded mode can be traded off to meet the actual requirements.

III. Proposed Blockchain Accelerator (BCA)

A. Overview of the Architecture

Fig. 4 presents an overview of an embedded SoC implementing our proposed BCA for performing mining tasks. The BCA connects with other parts of the system via an AXI bus. The CPU is the microprocessor that controls the entire SoC. During the mining process, the CPU sends commands to obtain transaction data from the BC network through a network interface controller (NIC) and writes these data into double data random access (DDR) memory. Some light processing, such as computing the Merkle tree hash of the

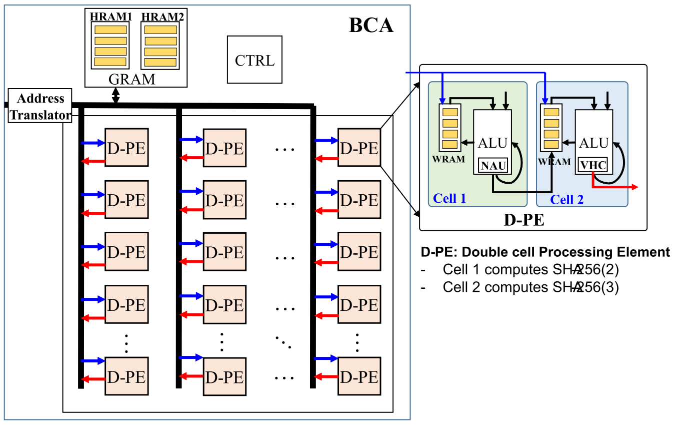

BCA
HRAM1
HRAM2
GRAM
CTRL
Address Translator
D-PE
WRAM
ALU
NAU
VHC
Cell 1
Cell 2
D-PE: Double cell Processing Element
- Cell 1 computes SHA256(2)
- Cell 2 computes SHA256(3)

Fig. 5. Overview of the proposed BCA architecture.

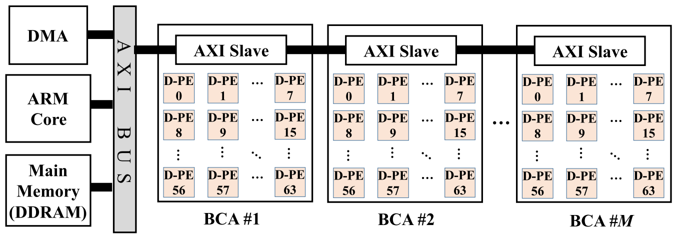


Fig. 6. Cascaded multiple BCA chip model.

transaction data and calculating SHA-256$_1$, should be performed in advance by the CPU. The CPU then sends the heavy task of extensively calculating SHA-256$_2$ and SHA-256$_3$ to the BCA to achieve faster processing. Data required for SHA-256$_2$ and SHA-256$_3$ computation will be transferred from DDR memory to the BCA. As soon as the BCA finds the valid hash value matching the target, it returns the valid hash to DDR memory, switches to idle status, and waits for new commands from the CPU. Data are transferred between the DDR memory and BCA through an advanced microcontroller bus architecture (AMBA) bus and AXI bus. Finally, the CPU forms a new block based on the valid hash found by the BCA and broadcasts the new block to the BC network.

Fig. 5 presents an overview of the proposed BCA architecture. The main component of the BCA is an array of D-PEs responsible for accelerating the final two processes of double SHA-256 (SHA-256$_2$ and SHA-256$_3$). In addition, it includes an *address translator*, a global memory (*GRAM*), and a controller (*CTRL*). The *GRAM* includes two sets of memory named HRAM1 and HRAM2 that store the global parameters used by all D-PEs of the BCA. The *address translator* translates the data address of the AXI bus into the appropriate address of local memories inside the GRAM and D-PEs. The *CTRL* provides the control signals to govern the computation and data flows inside the D-PEs to ensure that the D-PEs operate correctly as scheduled. The D-PEs are computation units that work independently and in parallel. A D-PE is configured to compute SHA-256$_2$ and SHA-256$_3$ for a specific range of *nonce* values, which we call *nonce_step* A. Let us say that we have a BCA with 64 D-PEs. The BCA should scan all possible hash values equivalent to 2^{32} values of *nonce* to find a valid hash value smaller than the target value. Then, the BCA is programmed to have *nonce_step* $A = 2^{26}$, in which "D-PE 0", "D-PE i", and "D-PE 63" compute for *nonce* ranging from 0 to A, $(i \times A)$ to $((i+1) \times A-1)$, and $(63 \times A)$ to $(64 \times A-1)$, respectively.

Let us say that the CPU needs to find the valid hash for a block header. It first computes SHA-256$_1$ for the first 512 bits of header to obtain the intermediate hash value denoted $Hi0(1), \ldots, Hi7(1)$. It then asks the BCA to search for a valid hash of a block header by scanning all possible values of *nonce*. It must transfer the following data from DDR memory to the BCA: 1) *nonce_start* and *nonce_step*, where *nonce_start* is the starting value of *nonce* that BCA should use and *nonce_step* is the number of *nonce* values that should be searched by each D-PE of the BCA; 2) the input messages $(W0(1), \ldots, W15(1))$ and initial hash values $(Hi0(1), \ldots, Hi7(1))$ for computing SHA-256$_2$ (note that $W2(1)$ is also the target that BCA should use to specify the valid hash); and 3) the input message $(W8(2), \ldots, W15(2))$ and initial hash $(Hi0(2), \ldots, Hi7(2))$ for computing SHA-256$_3$. Once the BCA successfully finds the valid hash, the valid hash $(Ho0(r), \ldots, Ho7(r))$ and the equivalent *valid_nonce* value will be read out of the BCA.

In our design, each D-PE is able to find a hash value every clock cycle on average. The processing rate of the BCA depends on the number of implemented D-PEs. Some BC systems may require faster processing rates, while others may prioritize lower power consumption and hardware costs. To satisfy the requirements of a wide variety of systems, we develop a scalable cascaded multichip model, as shown in Fig. 6. In this model, only a small number of D-PEs (i.e., 64 D-PEs) are implemented in a BCA chip to achieve low hardware cost and low power consumption. If a system requires a faster processing speed, it can cascade multiple BCA chips into the same AXI bus. Different BCA chips may scan *nonce* values of different *timestamps*. If a BCA chip successfully finds a valid hash, it reports the result to the CPU, and the CPU sends commands to interrupt the current task of the other BCA chips. If the CPU receives a valid hash value from another node, it interrupts the current task of all BCA chips and assigns new tasks to all BCA chips.

Fig. 7 shows the memory map of the cascaded BCA chips from the perspective of the CPU. Because each BCA has been equipped with multiple local memory blocks, the amount of transferred data between the BCA and external DDRAM memory is significantly reduced to only 50 address words. Therefore, we reserve a slot of 64 address words for each BCA. To avoid any contention on the AXI bus, the BCA chips are assigned different address spaces. A cascade of M chips of the BCA needs $64 \times M$ address words of the CPU. The CPU specifies address words from $64 \times (i-1)$ to $(64 \times i) - 1$ to read/write data to/from BCA #i. Before activating a BCA, the CPU uses the AXI bus to transfer 41 input words to the BCA (*nonce_start*, *nonce_step*, $W0(1) \ldots W15(1)$, $Hi0(1) \ldots Hi7(1)$, $W8(2) \ldots W15(2)$, and $Hi0(2) \ldots Hi7(2)$). The *address translator* of the BCA then translates the address of the AXI bus into an appropriate

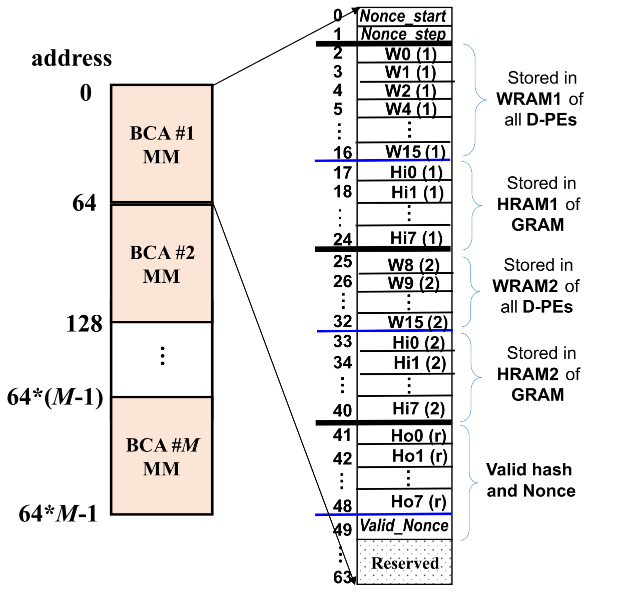


Fig. 7. Memory map of the proposed cascaded BCA chips from the perspective of the CPU.

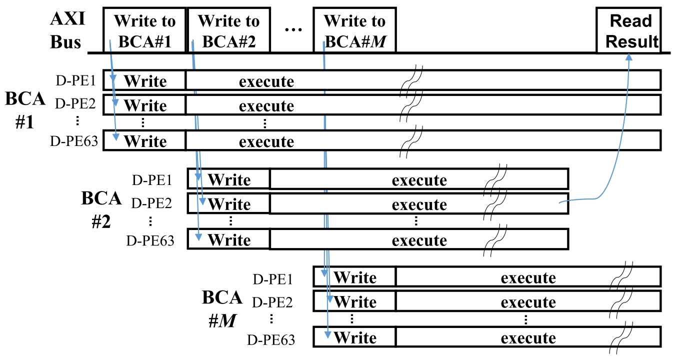


Fig. 8. Operating timing chart of the proposed cascaded BCA chips.

address of the local memories of all D-PEs. Fig. 8 illustrates the timing chart of cascaded BCA chips, in which valid hash and nonce values are found by D-PE2 of BCA #2. Fig. 8 shows that data from the CPU are written in parallel into local memories inside all D-PEs of the same BCA chip. Then, the D-PEs are executed in parallel. Data written into different BCA chips are utilized in different time slots by sharing the same AXI bus. If a D-PE successfully finds valid hash and nonce values, then the other D-PEs of the same BCA will stop working. The D-PEs of other BCAs will continue their execution until receiving commands from the CPU.

Regarding the scalability of the proposed architecture, the cascading of M chips of the BCA can enhance the processing speed by M times, in which the maximum value of M depends on the available hardware resources of the system and the available range of addresses that can be controlled by the CPU. If there is no problem with the hardware resources, then a 32-bit CPU can theoretically control up to $2^{32}/(64\times4) = 2^{24}$ chips of the BCA. Different BCAs may find valid hashes for different timestamps or different block headers simultaneously.

B. Double-Cell Processing Element (D-PE)

Fig. 9 depicts the architecture inside each D-PE of the BCA. It includes two cells for computing SHA-256_2 and SHA-256_3 of the double SHA-256. The architectures of cells 1 and 2 are similar, and a cell includes two main components: a pipelined arithmetic logic unit (ALU) and a set of local memories named WRAM. Cell 1 contains ALU1 and WRAM1, and cell 2 contains ALU2 and WRAM2. The ALU is responsible for computing all loops of hash functions. To achieve a high maximum frequency, the ALU is designed to compute a hash loop in 4 pipelined clock cycles. WRAM stores input messages W_0, ..., and W_{63}, which are required for the ALU computation. To maximize the utilization of ALU hardware resources, 4 two-port RAM blocks (WM1, WM2, WM3, and WM4) are implemented in the WRAM. These RAM blocks are scheduled to generate 4 pipelined data flows of 4 independent hash functions. Input shift buffer (SBi) and output shift buffer (SBo) blocks are implemented to rearrange the data flows between the WRAM and ALU. Essentially, the SBi includes 4 sets of shift registers and a 4-to-1 multiplexer to adjust the timing of 4 data flows from the WRAM to ALU. The SBo includes 4 sets of shift registers and a 1-to-4 demultiplexer to deliver the outputs of the ALU to the corresponding RAM block inside the WRAM.

In addition, cell 1 implements an *NAU* mechanism that automatically updates the *nonce* value for the next trial if a valid hash has not yet been found. According to our investigation, most cryptocurrency networks, including Bitcoin and Bitcoin Cash, have a *nonce* field at the position of input message W_3 of SHA-256_2. Therefore, the value W_3 in all local memories of WRAM1 will be adjusted in increments of 1 immediately after its old value is no longer required for the hash calculation of the current function. We apply increments of 1 for two reasons. First, this results in a small circuit. Second, double SHA-256 is a pseudorandom function in which the relationship between the *nonce* value and the probability of finding a valid hash is unknown. A random number of *nonce* fields can still be selected for each trial; however, this method is inefficient because the hash calculation for the same *nonce* field may be meaninglessly repeated more than once, and the hardware circuit for generating random *nonce* values is complex. In our design, the *nonce* value (W_3) is updated after the 19^{th} loop of the current hash calculation is completed (Fig. 10). Cell 2 has a *VHC* to check whether the hash result is smaller than the target. When a valid hash is found, a *chip_stop* signal is activated to inform all D-PEs of the BCA about the finding and tstop the processing of all D-PEs. The valid hash and its equivalent *nonce* value will then be read out by the CPU.

A D-PE computes 4 independent hash functions in parallel and pipelined manners in $64 \times 4 = 256$ clock cycles on average (64 clock cycles per hash function). Thus, the processing rate of a BCA chip with 64 D-PEs is 1 hash/cycle.

1) Pipelined ALU: Fig. 11 depicts the circuit inside the pipelined ALU. For the sake of low critical path delay, the ME process is computed in two pipelined stages (ME 1 and ME 2), and the MC process is computed in 4 pipelined stages (MC 1, MC 2, MC 3, and MC 4). To reduce latency, we design

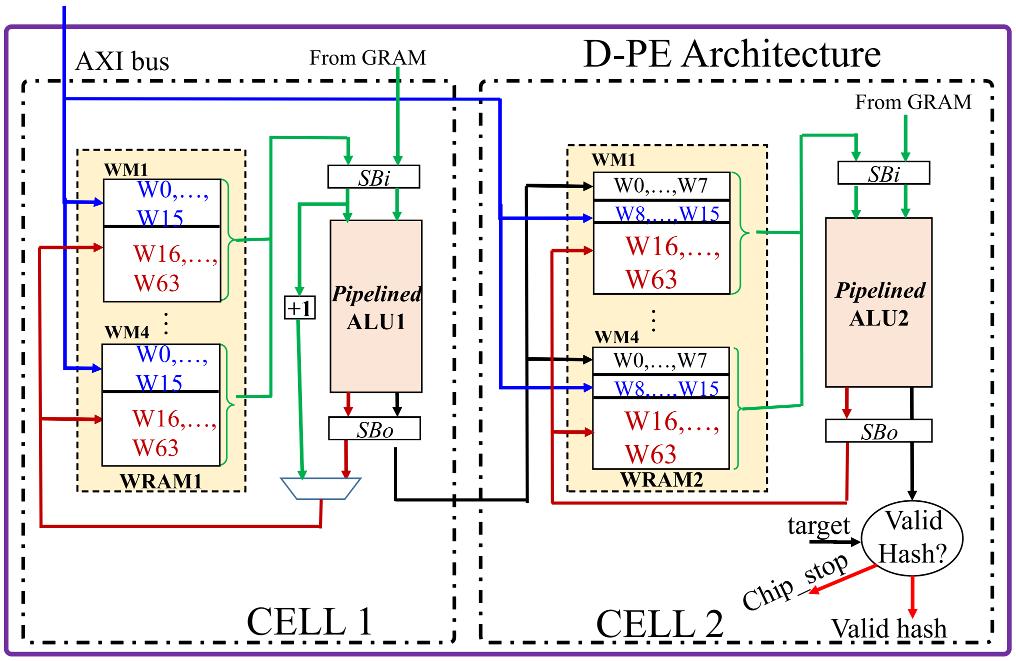


Fig. 9. Inside architecture of a double-cell processing element (D-PE).

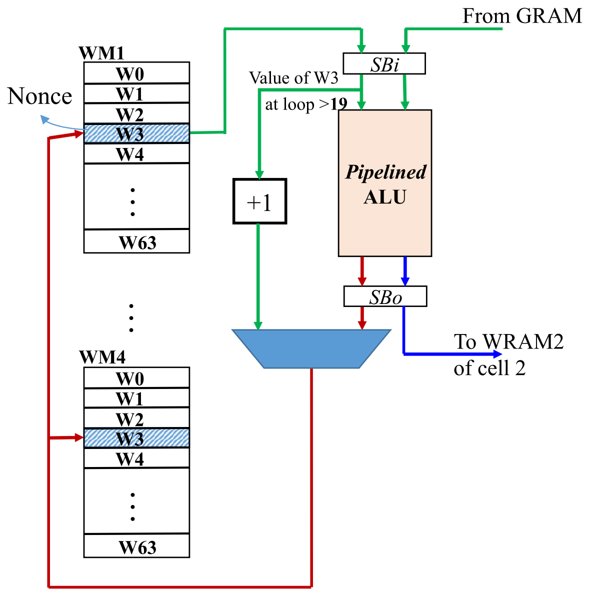


Fig. 10. Illustration of nonce autoupdate (NAU).

ME 1 to work in parallel with MC 1, and ME 2 works in parallel with MC 2. The result of ME 2 (W_j) is passed to MC 3 for further calculation and is written into the WRAM for future usage. The ALU has a recursive computation mode so that all loops of the hash function can be computed inside a single ALU. According to this mode, the loop hash results $(a, \ldots, h)$ of MC 4 are fed back to MC 1 for the next loop computation until all the loops are completed. Once all 64 loops are completed, the final hash values $(Ho_0, \ldots, Ho_7)$ are computed in the 5th stage by adding the results of MC 4 $(a, \ldots, h)$ and the initial hash values $(Hi_0, \ldots, Hi_7)$. The ALU can compute hash values for 4 independent hash functions by receiving 4 data flows from the local memories WM1, WM2, WM3, and WM4 of the WRAM. While data flow 1 is executed at stage MC 4, data flows 2, 3, and 4 are executed at stages MC 3, MC 2, and MC 1, respectively. This means that the ALU achieves 100% hardware efficiency.

2) Multiple Local Memory Structure: Multiple local memory structures are implemented inside the global memory GRAM and the D-PE WRAM to speed up the hash calculation and enhance the ALU hardware efficiency.

a) GRAM: The GRAM includes two sets of local memory, HRAM1 and HRAM2 (Fig. 5). HRAM1 stores initial hash values $Hi_0(1), \ldots, Hi_7(1)$ that are required for the calculation inside ALU1 of cell 1 of all D-PEs. These hashes are actually the results of SHA-256$_1$ computed by the CPU in advance. HRAM2 stores initial hash values $Hi_0(2), \ldots, Hi_7(2)$ that are required for the calculation inside ALU2 of cell 2 of all D-PEs. These hashes are constant parameters defined by the SHA-256 algorithm. We implement 4 identical single-port RAM blocks inside each HRAM1 and HRAM2 to provide 4 pipelined data flows to the ALUs. Each single-port RAM size is 8×32 bits.

b) WRAM1 of cell 1: WRAM1 includes four identical two-port RAM blocks with sizes of 64×32 bits to store the input messages $W_0(1), \ldots, W_{63}(1)$ required for calculating SHA-256$_2$ of four independent hash functions (Fig. 9). The first 16 words $W_0(1), \ldots, W_{15}(1)$ are read from the external DDR memory in advance, and the last 48 words $W_{16}(1), \ldots, W_{63}(1)$ are calculated by ALU1. Because the D-PE of the BCA is designed to compute an unlimited number of hash functions until a valid hash is found, $W_3(1)$ (equivalent to the *nonce* value) and $W_{16}(1), \ldots, W_{63}(1)$ are updated every hash function. In this manner, the memory resources necessary for only 1 hash function can be reused to store

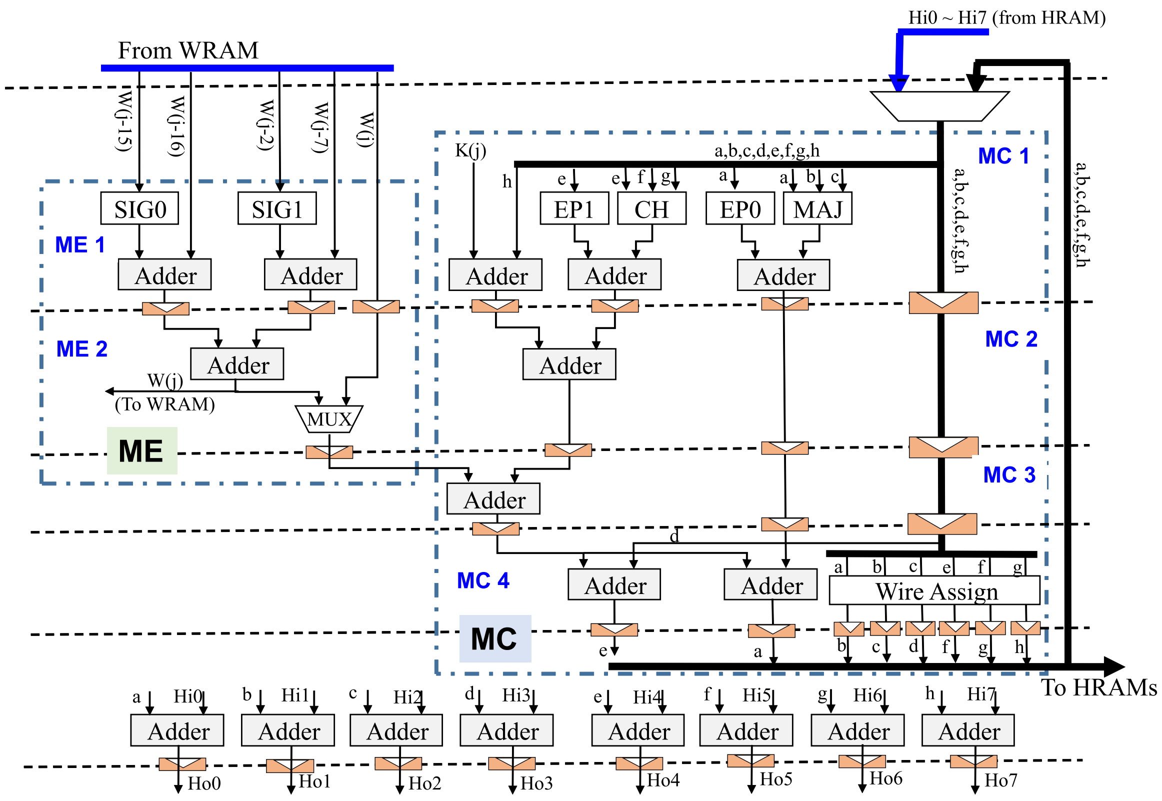


Fig. 11. Inside circuit of the pipelined ALU.

the data required for computing an unlimited number of hash functions. The hash results computed by cell 1 (SHA-256_2 results) become part of the input message $W_0(2), \dots, W_7(2)$ for computing SHA-256_3 in cell 2. These results will be written into the equivalent RAM block of WRAM2 of cell 2.

c) WRAM2 of cell 2: WRAM2 includes four identical two-port RAM blocks with sizes of 64×32 bits to store the input messages $W_0(2), \dots, W_{63}(2)$ required for calculating SHA-256_3 of four independent hash functions (Fig. 9). Before the BCA is activated, part of the input message of SHA-256_3 ($W_8(2), \dots, W_{15}(2)$) read from the external DDR memory is stored in addresses 8 to 15 of each RAM block of WRAM2. During the hash calculation, the hash results computed by cell 1, which become part of the input message of SHA-256_3 ($W_0(2), \dots, W_7(2)$), will eventually be updated and stored in addresses 0 to 7 of the equivalent RAM block of WRAM2. The last 48 words of the WRAM2 blocks store ($W_{16}(2), \dots, W_{63}(2)$), which are calculated by ALU2.

IV. Experimental Results

A. BCA Verification on Real Hardware

To prove that the BCA works correctly on a real hardware device, we developed and ran a BCA SoC platform on a Xilinx FPGA Alveo U280, as shown in Fig. 12. The platform consists of two main devices: a host PC and an Alveo U280 data center accelerator card. The two devices exchange data via JTAG and UART cables. The host PC runs a "Data Generator" C program and two Xilinx tools (a Vivado and a Vitis). The "Data Generator" obtains information from BC networks,

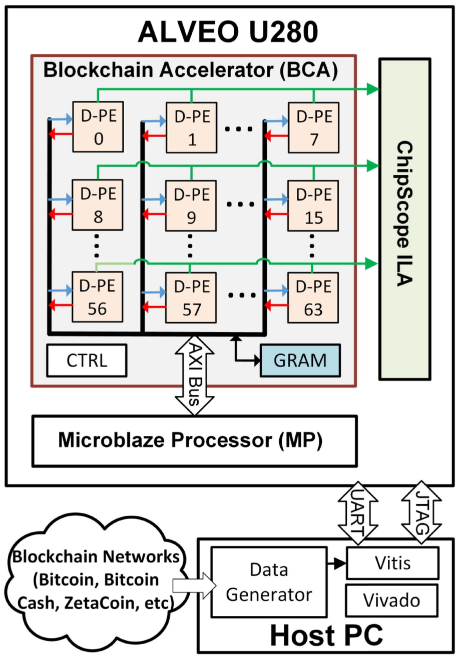


Fig. 12. Overview of the BCA verification platform on Alveo U280 FPGA.

forms the block header, and calculates SHA-256_1 of the double SHA-256. The Vitis tool runs a C program that configures the BCA and transfers input data into the BCA. The Vivado tools load the synthesized BCA circuit into an Alveo U280 FPGA card. The Alveo U280 FPGA embeds the following cores: a MicroBlaze Processor (MP), our BCA 64 D-PEs version, and

TABLE I
IMPACT OF THE PROPOSED TECHNIQUES INSIDE THE BCA ON REDUCING THE LOCAL MEMORY SIZE AND AMOUNT OF TRANSFERRED DATA

Evaluated content	[28], 2021	The proposed BCA
No. of hash functions per PE per config.	$4 \times L$	Unlimited
Size of a WM block in WRAM	$L \times 64 \times 32$	64×32
Normalized total no. of WM blocks	64×12	64×8
Size of an HM block in HRAM	$L \times 8 \times 32$	8×32
Normalized total no. of HM blocks	64×12	4
Amount of transferred data (bytes)	384×2^{32}	200

a ChipScope Integrated Logic Analyzer (ChipScope ILA). The MP transfers data from the host PC to the BCA and reads the valid hash and the equivalent *nonce* value from the BCA to the host PC. The ChipScope ILA is a set of tools offered by Xilinx supporting the verification of the BCA circuit. All three cores run at 100 MHz. During our experiment, we used Xilinx Vivado version 2019.2.

We experimented with two test cases: D-PE verification and mining tasks. In the D-PE verification, the D-PEs of BCA are programmed to calculate a predefined range of *nonce* fields of the same block header. All hash values computed by the D-PEs (including nonvalid hash values) are captured by the ChipScope ILA. The ChipScope ILA then compares the D-PE outputs with the hash values computed by an Intel Xeon Gold 6144 CPU. Our experiment proves that all D-PEs of the BCA work 100% correctly. In the mining task, we programmed the BCA to perform the mining task for the following BC networks: Bitcoin, Bitcoin Cash, Bitcoin Atom, BitcoinV, BitcoinSV, FreiCoin, Zetacoin, DeVault, Deutsche eMark, EmbargoCoin, Susucoin, FreeCash, FreeCash, and Kryptofranc. The experiment demonstrates that our BCA always generates the same valid hash and *nonce* values as those available in these BC networks. In other words, the BCA performs the mining task correctly.

B. Evaluating the Impact of the Proposed Techniques Inside the BCA

This work is the improved version of our previous work published in [28]. To clarify the impact of the proposed techniques, such as multimem local memory, D-PE, and NAU, on the BCA performance, we focus on comparing the performance of the proposed BCA and the accelerator in [28] in this section. The comparison is shown in Table I. Notably, the accelerator in [28] has been designed to compute the SHA-256 function; therefore, hardware resources to compute SHA-256_1, SHA-256_2, and SHA-256_3 of double SHA-256 are supposed to be increased three-fold to ensure that the calculation rate is not degraded. The normalized total numbers of memory blocks HM and WM in [28] are obtained by considering the difference between SHA-256 and double SHA-256. We evaluate the performance in terms of the number of hash functions that can be computed per configuration time, the size of local memories, and the amount of data transfer between the accelerator and external DDRAM.

1) Multimem Architecture: The multimem idea was originally proposed in our previous work [28] to increase the calculation rate of the accelerator. However, the addition of multiple local memory blocks resulted in an increase in hardware cost and power consumption. Selecting the size of local memory blocks involved a trade-off between the system processing rate and hardware cost. In this work, we utilized the characteristics of double SHA-256 for blockchain mining to modify the multimem architecture so that the size of the local memory block is minimized and no longer requires a trade-off between the system processing rate and hardware cost. Numerically, each block of the WRAM inside a PE of the accelerator in [28] must have a size of at least $L \times 64 \times 32 = 2048 \times L$ bits so that a PE can compute up to $4 \times L$ hash functions per configuration time. To compute the double SHA-256 function, the normalized number of blocks of WRAM per PE should be $4 \times 3 = 12$ blocks. Thus, an accelerator with 64 PEs requires a WRAM size of $64 \times 12 \times 2048 \times L$ bits. Each block of the HRAM inside a PE of the accelerator in [28] must have a size of at least $L \times 8 \times 32 = 256 \times L$ bits. To compute the double SHA-256 function, the normalized number of blocks of HRAM per PE should be $4 \times 3 = 12$ blocks. Thus, the accelerator with 64 PEs requires an HRAM size of $64 \times 12 \times 256 \times L$ bits. In contrast, the proposed BCA with 64 D-PEs requires only $64 \times 8 \times 64 \times 32$ bits (or 128 KB) for the total size of the WRAM and $4 \times 8 \times 32$ bits (or 128 bytes) for the total size of the HRAM. This means that the total sizes of the WRAM and HRAM are reduced by $1.5 \times L$ and $192 \times L$ times, respectively. Basically, L should be at least 1000 so that the accelerator in [28] can achieve an acceptable processing rate. In this case, the total sizes of the WRAM and HRAM can be reduced by 1500 and 192000 times, respectively.

2) D-PE Architecture: The PE architecture in [28] was developed to compute a single SHA-256 algorithm for multiple purposes. If it is used to compute double SHA-256 in blockchain mining, the intermediate values (such as the results of SHA-256_1 and SHA-256_2) must be read out of the accelerator and then written into the accelerator again to compute the final output (the result of SHA-256_3). As a result, the total processing time of the accelerator at the SoC level is remarkably increased because the data transfer time occupies a large time fraction, i.e., $3 \times 128 = 384$ bytes per double SHA-256 calculation. In this work, the D-PE architecture was proposed to compute all SHA-256 processes of double SHA-256 in pipelined pipes without transferring the intermediate values of a single SHA-256. The data transfer time is thus eliminated, and the double SHA-256 calculation rate is improved by $3\times$. The cost to achieve this performance is that the circuit area of computation units inside each D-PE of the BCA is approximately $2\times$ greater than that inside each PE in [28].

3) NAU and VHC: The baseline design in [28] computes single SHA-256 functions independently. The computation of 2^{32} values of double SHA-256 corresponding to 2^{32} values of *nonce* requires transferring an amount of 384×2^{32} bytes of data. The data transfer time occupies a large time portion of the total processing time, which may limit the total processing

TABLE II

EXPERIMENTAL RESULTS OF THE BCA AND THE STATE-OF-THE-ART CPU AND GPU PLATFORMS

Platform		Tech.	Chip Size (mm^2)	Power (W)		Frequency (MHz)	Exec. Time (second)	Throughput (Mhps)	Power Eff. (Mhps/W)	Area Eff. (Mhps/mm^2)
				Measured	TDP					
CPU Intel i9-10940X		ASIC 14 nm	-	164	165	3,300	394.77	11	0.67	-
GPU GTX 1080 Ti, 11 GB		ASIC 16 nm	471	188	250	1,481	23.26	185	0.98	0.393
GPU RTX 3070, 8 GB		ASIC 8 nm	392	197	220	1,500	20.78	207	1.05	0.528
GPU RTX 3090, 24 GB		ASIC 8 nm	628	346	350	1,395	10.45	411	1.19	0.654
GPU Tesla V100, 16 GB		ASIC 12 nm	815	163	250	1,245	13.04	329	2.02	0.404
Prop. BCA	FPGA ALVEO U280	FPGA 16 nm	-	**8.38**	-	400	11.95	360	**43**	-
	Layout Chip	ASIC 65 nm	25	**0.53**	-	100	-	90	**170** 1,381**	**3.6** 238*

*: Normalized area efficiency = Area Eff. × (65 nm/8 nm)^2
**: Normalized power efficiency = Power Eff. × (65 nm/8 nm)

time because the bottleneck issue on the maximum data transfer rate can be provided by the embedded board. In this work, we proposed a combination of the NAU and VHC that allows a BCA chip to continuously compute an unlimited number of double SHA-256 values until finding a valid hash. The computation requires transferring only 50 words (200 bytes) of data (see Fig. 7), which helps to significantly improve the total processing time.

We ran the baseline design in [28] on Xilinx FPGA ZCU-102 and the proposed BCA on Xilinx Alveo U280 to compute 2^{32} values of double SHA-256. We were not able to run both accelerators on the same hardware platform for a fair evaluation for the following reasons. First, we implemented the baseline design [28] on ZCU-102 because this board has an ARM processor that we can use to frequently send commands related to data transfer, while Alveo U280 does not have a processor to do so. Second, we implemented the BCA on Alveo U280 because this board has enough FPGA resources to embed the entire BCA chip, while ZCU-102 does not have enough FPGA resources to do so. On Alveo U280, we developed a simple microprocessor to perform the data transfer in and out of the BCA. The performance of this microprocessor is too low but is still acceptable because the amount of transfer data is only 50 words (200 bytes). In contrast, this microprocessor is not suitable for the task of transferring the large amount of data required by the baseline design in [28]. The experimental results showed that the total processing times of the baseline and the BCA are 6569 seconds and 11.95 seconds, respectively. While the computation time of the baseline design is longer than that of the BCA by only 3 times, the total processing time is increased by $6569/11.95 = 550$ times because of the bottleneck issue on the data transfer rate provided by the ZCU-102 FPGA board. Research efforts to improve the data transfer rate of the ZCU-102 FPGA board may help to reduce the total processing time. On the other hand, the implementation of D-PE architecture has resulted in the increment of circuit power consumption. Our experimental results showed that the power consumption of the baseline design and the BCA are 3.215 W and 8.38 W, respectively. Despite of this increment of power consumption, the energy efficiency of the BCA is significantly improved. Numerically, the energy consumption for computing 2^{32} values of double SHA-256 of the baseline design and the BCA are $3.125 \times 6569 = 20,528\ J$ and $8.38 \times 11.95 = 100J$, respectively. We can see that the proposed techniques, that is, multimem, D-PE, NAU and VHC, have been integrated in the same architecture of the BCA to significantly improve the processing rate (550 times) and energy efficiency $(20,528/100 = 205$ times) compared to those of the baseline design in [28] in the case of performing the blockchain mining task.

C. Performance Evaluation: BCA vs. State-of-the-Art CPUs and GPUs

This section describes our experimental results obtained from evaluating the performance of the BCA and state-of-the-art CPU and GPU platforms when computing double SHA-256 in Bitcoin mining.

Our evaluation results are shown in Table II. We focused on evaluating the processing rate throughput R, power efficiency, and area efficiency. During our experiment, we focused on three main tasks that evaluate the performance of the BCA on FPGA, the performance of the BCA on ASIC, and the performance of the state-of-the-art CPU and GPU platforms. For comparison, all hardware platforms perform the same task of computing double SHA-256 2^{32} times, corresponding to 2^{32} different values of the *nonce* field.

1) BCA on FPGA Experiment: We embedded the proposed BCA 64 D-PEs version in a Xilinx Alveo U280 FPGA card and programmed the BCA to compute double SHA-256 2^{32} times. The embedded system is the same as that depicted in Fig. 12 but without the ChipScope ILA, which was used for debugging and verification purposes. Our experiment shows that the BCA core occupies 18% of the LUT resources and 20% of the memory resources of the Alveo U280 card, consumes 8.38 W of dynamic power when operating at 400 MHz and takes 11.95 seconds to compute double SHA-256 2^{32} times.

2) BCA on ASIC Experiment: During our experiment, we used "Design Compiler version N-2017.09-SP1" and "IC Compiler version Q-2019.12-SP4" for ASIC synthesis

TABLE III

OPTIMAL NUMBER OF THREADS, CORES, AND BLOCKS OF THE CPUs AND GPUs FROM OUR EXPERIMENT

Platform	Cores/ Blocks	Threads per Core/Block	Total Parallel Thread Execution (N)
CPU Intel i9-10940X	14	2	28
GPU GTX 1080Ti, 11 GB	256	128	32,768
GPU RTX 3070, 8 GB	2,048	16	32,768
GPU RTX 3090, 24 GB	256	256	65,536
GPU Tesla V100, 16 GB	256	256	65,536

and layout. We successfully laid out the BCA 64 D-PE version in ASIC using library *Ptt_V0p75_T25* of Renesas SOTB 65 nm technology. We used the *compile_ultra* command with the clock gating option. The exact optimization command was *compile_ultra −incremental −gate_clock −timing_high_effort_script*. Because of licensing issues, we were unable to use dedicated two-port RAM blocks provided by our technology. All RAM blocks of the BCA were thus synthesized and laid out from flip-flops, which consume more power and area than dedicated RAM blocks. The layout chip with a size of 25 mm^2 consumes 530 mW of power when running at a 100 Mhz frequency and 0.75 V applied voltage. It is worth noting that the BCA embedded on FPGA Alveo U280 can operate at a higher frequency than the layout chip of the BCA on ASIC 65 nm because of the difference in technologies. The FPGA Alveo U280 was fabricated using Xilinx 16 nm UltraScale+ modern technology, while the ASIC layout chip of the BCA was fabricated from Renesas 65 nm old technology. The frequency of 400 MHz on the FPGA was the maximum operating frequency reported by the Vivado synthesis tool, while 100 MHz on the ASIC was obtained from our constraint setting with consideration of the trade-off between operating frequency and power consumption versus hardware cost and layout time.

3) State-of-the-Art CPU and GPU Experiment: We developed C and CUDA codes to run the task (double SHA-256 calculation 2^{32} times) on an Intel multicore CPU (Intel i9-10940X) and Nvidia GPUs (GTX 1080 Ti, RTX 3070, RTX 3090, and Tesla V100-PCIE). These platforms were selected because they are currently the fastest and most popular accelerators for performing Bitcoin mining. To ensure a fair comparison, we ran several trials while changing the number of parallel threads to ensure that the CPU and GPUs achieved their maximum processing speed. Table III shows the optimal number of threads, cores, and blocks of the CPU and GPUs that result in the fastest processing speed that we obtained in the experiment. We ran double SHA-256 on the CPU and GPUs using these optimal values for comparison with our proposed BCA. We recorded the execution time T and the measured power consumption P during execution of the CPU and GPUs. In addition, we obtained information such as the fabrication technology, thermal design power (TDP), and chip size of these platforms from their data sheets provided by the manufacturers (Intel for the CPU and NVidia for the GPUs).

4) Data Processing: From the experimental execution time T and the measured power consumption P, we computed the processing rate throughput R in terms of mega hash per second (Mhps), power efficiency in terms of Mhps per unit power W (Mhps/W), and area efficiency in terms of Mhps per unit area mm^2 ($Mhps/mm^2$) following eqs. (14), (15), and (16), respectively. Furthermore, we roughly calculated the power efficiency and area efficiency normalized to the same technology ASIC 8 nm using the scaling rules in eq. (17) and eq. (18), respectively. These scaling rules have been applied in several publications, such as [29] and [30].

$$R = \frac{2^{32}}{10^6 \times P} \tag{14}$$

$$Power\_eff = R/P \tag{15}$$

$$Area\_eff = R/A \tag{16}$$

$$Normalized\_power\_eff = \frac{1}{s} \times Power\_eff \tag{17}$$

$$Normalized\_area\_eff = \frac{1}{s^2} \times Area\_eff \tag{18}$$

where P and A are the measured power consumption and area of the hardware platform, respectively. s is the scaling factor of the transistors of the two different technologies. The normalization of our 65 nm technology to 8 nm technology involves a factor of $s = 65/8 = 8.125$.

5) Performance Evaluation: Table I summarizes our experimental results. We discuss the following evaluation results.

a) Performance of the BCA on Alveo U280: The processing speed of the BCA is **32.7 times** (360 vs. 11 Mhps) faster than that of the Intel i9-10940X CPU. The processing speed of the BCA is **1.95 times** (360 vs. 185), **1.74 times** (360 vs. 207), and **1.1 times** (360 vs. 329) faster than those of GTX 1080Ti, RTX 3070, and Tesla V100 GPUs, respectively, even though the BCA is embedded on FPGA 28 nm, while GTX 1080Ti, RTX 2070, and RTX 3090 are fabricated on ASIC 16 nm, ASIC 8 nm, and ASIC 12 nm, respectively. The power efficiency of the proposed BCA on Alveo U280 is improved by **642 times** (43 vs. 0.067 Mhps/W), **44 times** (43 vs. 0.98), **41 times** (43 vs. 1.05), **36 times** (43 vs. 1.19) and **21 times** (43 vs. 2.02) compared with those of the Intel i9-10940X CPU, GTX 1080 Ti GPU, RTX 3070 GPU, RTX 3090 GPU, and Tesla V100 GPU, respectively. This means that using the proposed BCA on Alveo U280 instead of using the above CPU and GPUs to perform the mining task could help reduce electricity consumption by the same amount (642, 44, 41, 36, and 21 times).

b) Performance of the BCA layout chip: The power efficiency of the layout chip of the BCA on ASIC 65 nm technology is improved by **2428 times** (170 vs. 0.067 Mhps/W), **173 times** (170 vs. 0.98), **162 times** (170 vs. 1.05), **143 times** (170 vs. 1.19), and **84 times** (170 vs. 2.02) compared with those of the Intel i9-10940X CPU, GTX 1080 Ti GPU, RTX 3070 GPU, RTX 3090 GPU, and Tesla V100 GPU, respectively. The area efficiency of the BCA layout chip is improved **9.2 times** (3.6 vs. 0.393 $Mhps/mm^2$), **6.8 times** (3.6 vs. 0.528), **5.5 times** (3.6 vs. 0.654), and **8.9 times** (3.6 vs. 0.404) compared with those of the GTX 1080 Ti, RTX 3070, RTX 3090, and Tesla V100 GPUs, respectively.

TABLE IV

EXPERIMENTAL RESULTS OF THE BCA AND THE STATE-OF-THE-ART FPGA-BASED DESIGNS

Design level		Standalone				SoC		
Research		[31], 2019	[20], 2019	[32], 2020	**This paper**	[33], 2020	[28], 2021	**This paper**
FPGA Platform		KCU116	KCU116	KCU116	**Alveo U280**	ZC702	ZCU102	**Alveo U280**
Technology (nm)		16	16	16	**16**	28	16	**16**
Algorithm		Single SHA-256			**Double SHA-256**	Single SHA-256		**Double SHA-256**
Area	LUT	1,819	5,314	9,316	**86,025**	6,367	205,007	**228,666**
	FF	1,327	4,302	4,188	**267,706**	25,190	346,391	**512,770**
Frequency (MHz)		264	364	111	**400**	181	320	400
Throughput (Mhps)	Single SHA-256	4.06	5.6	6.53	-	1.8	320	-
	Double SHA-256	1.35*	1.87*	2.18*	**400**	0.6*	106.67*	**360**
Power (W)		0.18	0.41	0.45	**3.39**	0.1	6.88	**8.38**
Area Efficiency (Mhps/1kLUT)		0.74	0.35	0.23	**4.65**	0.09	0.52	**1.57**
Energy Efficiency (Mhps/W)		7.5	4.56	4.84	**118**	6.67	15.5	**42.96**

* Normalized throughput of double SHA-256 = $\frac{1}{3}$ × throughput of single SHA-256.

c) Expected performance of the BCA on ASIC 8 nm: We normalized the power and area efficiencies of our BCA to the fastest technology, 8 nm, used by the RTX 3070 and RTX 3090 GPUs. The normalized results should be understood as a rough estimation to predict the expected performance of our BCA if it is fabricated on ASIC 8nm. These are not experimental results; therefore, the evaluation may not be completely correct. We provide the following statement for reference purposes simply to demonstrate our expectations. If our proposed BCAs were fabricated with the same fabrication technology as that used to fabricate the modern RTX 3070 and RTX 3090 GPUs (ASIC 8 nm), the power efficiency of the BCA would be expected to improve by **1315 times** (1381 vs. 1.05) and **1160 times** (1381 vs. 1.19), respectively, compared with these GPUs. The BCA area efficiency is expected to improve by **451 times** (238 vs. 0.528) and **394 times** (238 vs. 0.654) compared with these GPUs (RTX 3070 and RTX 3090).

D. Performance Evaluation: BCA Vs. FPGA-Based Works

This section compares the performance of the proposed BCA and that of the related works embedded in FPGAs, as shown in Table IV. We evaluated two design levels: standalone and SoC. At the standalone level, only the computation units (ALUs) and the controller (CTRL) of the BCA were synthesized and evaluated. It needs 86025 LUTs and 267706 flip-flops, consumes 3.39 W when operating at 400 MHz and provides a processing speed of 400 MHz. The area and power efficiencies of the proposed BCA are 4.65 Mhps/kLUT, and 118 Mhps/W, respectively. Compared with the values in previous studies, i.e., [20], [31], and [32], the area efficiency of the proposed BCA is improved by 6.2-, 13.3-, and 20.2-fold, respectively, and the power efficiency of the proposed BCA is improved by 15.7-, 25.9-, and 24.4-fold, respectively.

At the SoC design level, the full circuit of the BCA was synthesized and evaluated. It needs 228666 LUTs and 512770 flip-flops, consumes 8.38 W when operating at 400 MHz and provides a processing rate of 360 MHz. The numbers of LUTs and flip-flops are significantly increased compared with those at the BCA standalone level. However, the increase is reasonable because local memories such as WRAM and HRAM have been synthesized from LUTs and flip-flops. Compared with the baseline design in [28], the number of LUTs of the BCA is slightly increased due to the double number of ALU units per D-PE, which is consistent with our theoretical analysis in section IV.B. It is worth noting that while the number of ALU units per D-PE is doubled, the HRAM blocks per D-PE have been eliminated. Therefore, the numbers of LUTs are not doubled but slightly increased, as shown in Table IV. The number of flip-flops of the BCA is increased by approximately 1.5× compared with that of the baseline design in [28] for two reasons. First, the number of WRAM blocks per D-PE in the BCA is doubled. Second, the experimental results of [28] are obtained in case $L = 3$, which is much smaller than the requirement of real applications. Overall, the hardware and power efficiency of the BCA are expected to be much better than those of [28] because the processing rate for computing double SHA-256 of the BCA is much higher than that of [28]. Numerically, compared with previous studies, i.e., [33] and [28], the BCA hardware efficiency is improved 17.4- and 3-fold, respectively, and the power efficiency is improved 6.4- and 2.77-fold, respectively. In particular, the hardware resources and power consumption of [28] were obtained in the case of $L = 3$, which is too small for real applications. The processing rate of the accelerator in [28] was calculated by ignoring the processing rate degradation due to the data transfer bottleneck to/from the accelerator. Therefore, we believe that the improvement in area and power efficiencies of the BCA vs. the accelerator in [28] will be much higher than the aforementioned values if both accelerators are utilized for mining blockchain networks in reality.

V. CONCLUSION

The computation of the double SHA-256 function is responsible for nearly all of the energy consumption required to secure a BC network. In this paper, we have proposed an ultralow power and high-throughput BCA architecture for accelerating double SHA-256 computation. Unlike the related studies that accelerated the computation only, our BCA is focused on improving the accelerator throughput at the system level. We proposed a multiple local memory structure of WRAM and GRAM to both reduce the data transfer time between the BCA and external DDR memory and improve the hardware efficiency of the BCA calculation units. The D-PE architecture was originally introduced such that the entire double SHA-256 function is computed inside a single D-PE and all D-PEs work as independent processing cores to compute a large number of double SHA-256 functions simultaneously. No wire connection among the D-PEs is required, which results in lower power consumption and less chip area. The NAU and VHC mechanisms were employed such that the BCA computes up to 2^{32} double SHA-256 functions until it finds a valid hash value without interruption for external data transfer. Furthermore, a cascaded multiple BCA chip model was recommended for endlessly increasing the total processing throughput to meet real system requirements. Our experiments on FPGA Alveo U280 and ASIC Renesas SOTB 65 nm technology have proven that the BCA successfully performs mining tasks for multiple BC networks with much lower power consumption and hardware costs than the state-of-the-art CPU and GPU platforms. Notably, the power efficiency of the BCA layout chip on 65 nm technology is 2428, 173, 162, 143, and 84 times better than that of the Intel i9-10940X CPU, GTX 1080 Ti GPU, RTX 3070 GPU, RTX 3090 GPU, and Tesla V100 GPU, respectively. We even expect to improve the power efficiency up to 1160 times relative to the currently fastest Nvidia GPU RTX 3090 if the BCA is fabricated with the same RTX 3090 technology (ASIC 8 nm).

Currently, we are investigating how to apply approximate computing and neuromorphic computing techniques to reduce the power consumption of the current BCA to approximately 1 mW. At the level of 1 mW, the BCA would be able to operate sustainably using green energy harvested from solar, movement, and thermal sources. We are looking forward to a bright future in which not only power-constrained devices, such as autonomous cars, smartphones, and smart watches but also small sensors are able to employ our BCA to participate in securing BC networks. The network would then become truly decentralized such that no one and nothing is able to destroy it.

Appendix

The source codes and the synthesized and laid out results of our experiments on FPGA and ASIC can be found at https://github.com/archlab-naist/530mW-BCA/

Acknowledgment

The BCA architecture was synthesized and laid out on ASIC Renesas SOTB 65 nm technology under the Japan VLSI Design and Education Center (VDEC) license.

References

[1] Japan Cabinet Office. *Society 5.0*. Accessed: Mar. 26, 2021. [Online]. Available: https://www8.cao.go.jp/cstp/english/society5_0/index.html

[2] H. L. Pham, T. H. Tran, and Y. Nakashima, "A secure remote healthcare system for hospital using blockchain smart contract," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6, doi: 10.1109/GLOCOMW.2018.8644164.

[3] H. L. Pham, T. H. Tran, and Y. Nakashima, "Practical anti-counterfeit medicine management system based on blockchain technology," in *Proc. 4th Technol. Innov. Manage. Eng. Sci. Int. Conf. (TIMES-iCON)*, Bangkok, Thailand, Dec. 2019, pp. 1–5, doi: 10.1109/TIMES-iCON47539.2019.9024674.

[4] V.-C. Nguyen, H.-L. Pham, T.-H. Tran, H.-T. Huynh, and Y. Nakashima, "Digitizing invoice and managing VAT payment using blockchain smart contract," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, Seoul, South Korea, May 2019, pp. 74–77, doi: 10.1109/BLOC.2019.8751256.

[5] S. Fukao and C. Ballinder, *Mobility Economics*. Tokyo, Japan: Nikkei, 2020, pp. 85–96.

[6] M. Rahouti, K. Xiong, and N. Ghani, "Bitcoin concepts, threats, and machine-learning security solutions," *IEEE Access*, vol. 6, pp. 67189–67205, 2018.

[7] H. Vranken, "Sustainability of Bitcoin and blockchains," *Current Opinion Environ. Sustain.*, vol. 28, pp. 1–9, Oct. 2017.

[8] J. Taskinsoy, "Bitcoin and Turkey: A good match or a perfect storm?" *SSRN Electron. J.*, pp. 1–23, Oct. 2019, doi: 10.2139/ssrn.3477849.

[9] R. García, I. Algredo-Badillo, M. Morales-Sandoval, C. Feregrino-Uribe, and R. Cumplido, "A compact FPGA-based processor for the secure hash algorithm SHA-256," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 194–202, Jan. 2014.

[10] M. D. Rote, N. Vijendran, and D. Selvakumar, "High performance SHA-2 core using the round pipelined technique," in *Proc. IEEE Int. Conf. Electron., Comput. Commun. Technol. (CONECCT)*, Jul. 2015, pp. 1–6.

[11] L. Dadda, M. Macchetti, and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Feb. 2004, pp. 70–75.

[12] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," in *Proc. 7th Int. Symp. Embedded Comput. Syst. Design (ISED)*, Dec. 2017, pp. 1–4.

[13] X. Zhang, R. Wu, M. Wang, and L. Wang, "A high-performance parallel computation hardware architecture in ASIC of SHA-256 hash," in *Proc. 21st Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2019, pp. 52–55.

[14] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Cost-efficient SHA hardware accelerators," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 8, pp. 999–1008, Aug. 2008.

[15] R. P. McEvoy, F. M. Crowe, C. C. Murphy, and W. P. Marnane, "Optimisation of the SHA-2 family of hash functions on FPGAs," in *Proc. IEEE Comput. Soc. Annu. Symp. Emerg. VLSI Technol. Architectures (ISVLSI)*, Mar. 2006, pp. 317–322.

[16] S. B. Suhaili and T. Watanabe, "Design of high-throughput SHA-256 hash function based on FPGA," in *Proc. 6th Int. Conf. Electr. Eng. Informat. (ICEEI)*, Nov. 2017, pp. 1–6.

[17] H. E. Michail, G. S. Athanasiou, V. Kelefouras, G. Theodoridis, and C. E. Goutis, "On the exploitation of a high-throughput SHA-256 FPGA design for HMAC," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, no. 1, pp. 1–28, Mar. 2012.

[18] T. H. Tran, S. Kanagawa, D. P. Nguyen, and Y. Nakashima, "ASIC design of MUL-RED Radix-2 pipeline FFT circuit for 802.11ah system," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, Yokohama, Japan, Apr. 2016, pp. 1–3.

[19] F. Opritoiu, S. L. Jurj, and M. Vladutiu, "Technological solutions for throughput improvement of a secure hash algorithm-256 engine," in *Proc. IEEE 23rd Int. Symp. Design Technol. Electron. Packag. (SIITME)*, Oct. 2017, pp. 159–164.

[20] R. Martino and A. Cilardo, "A flexible framework for exploring, evaluating, and comparing SHA-2 designs," *IEEE Access*, vol. 7, pp. 72443–72456, 2019.

[21] R. Martino and A. Cilardo, "A configurable implementation of the SHA-256 hash function," in *Proc. Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.* Cham, Switzerland: Springer, Nov. 2019, pp. 558–567.

[22] F. Kahri, B. Bouallegue, M. Machhout, and R. Tourki, "An FPGA implementation and comparison of the SHA-256 and blake-256," in *Proc. 14th Int. Conf. Sci. Techn. Autom. Control Comput. Eng. (STA)*, Dec. 2013, pp. 152–157, doi: 10.1109/STA.2013.6783122.

[23] Y. Chen and S. Li, "A high-throughput hardware implementation of SHA-256 algorithm," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–4.
[24] V. B. Suresh, S. K. Satpathy, and S. K. Mathew, "Optimized SHA-256 datapath for energy-efficient high-performance Bitcoin mining," U.S. Patent 10 755 242 B2, Aug. 25, 2020.
[25] V. B. Suresh, S. K. Satpathy, and S. K. Mathew, "Bitcoin mining hardware accelerator with optimized message digest and message scheduler datapath," U.S. Patent 10 142 098 B2, Nov. 27, 2018.
[26] H. L. Pham, T. H. Tran, T. D. Phan, V. T. D. Le, D. K. Lam, and Y. Nakashima, "Double SHA-256 hardware architecture with compact message expander for Bitcoin mining," *IEEE Access*, vol. 8, pp. 139634–139646, 2020, doi: 10.1109/ACCESS.2020.3012581.
[27] Y. Zhang *et al.*, "A new message expansion structure for full pipeline SHA-2," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 4, pp. 1553–1566, Apr. 2021, doi: 10.1109/TCSI.2021.3054758.
[28] T. H. Tran, H. L. Pham, and Y. Nakashima, "A high-performance multimem SHA-256 accelerator for society 5.0," *IEEE Access*, vol. 9, pp. 39182–39192, 2021, doi: 10.1109/ACCESS.2021.3063485.
[29] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation," *IEEE J. Solid-State Circuits*, vol. 46, no. 7, pp. 1754–1765, Jul. 2011, doi: 10.1109/JSSC.2011.2144470.
[30] N. Prasad, I. Chakrabarti, and S. Chattopadhyay, "An energy-efficient network-on-chip-based reconfigurable Viterbi decoder architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 10, pp. 3543–3554, Oct. 2018, doi: 10.1109/TCSI.2018.2825362.
[31] D. K. N. and R. Bhakthavatchalu, "Parameterizable FPGA implementation of SHA-256 using blockchain concept," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2019, pp. 0370–0374.
[32] R. Martino and A. Cilardo, "Designing a SHA-256 processor for blockchain-based IoT applications," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100254.
[33] M. Kammoun, M. Elleuchi, M. Abid, and M. S. BenSaleh, "FPGA-based implementation of the SHA-256 hash algorithm," in *Proc. IEEE Int. Conf. Design Test Integr. Micro Nano-Syst. (DTS)*, Jun. 2020, pp. 1–6.

Thi Hong Tran (Member, IEEE) received the bachelor's degree in physics and the master's degree in microelectronics from Vietnam National University-Ho Chi Minh City University of Science (VNU-HCMUS), Vietnam, in 2008 and 2012, respectively, and the Ph.D. degree in information science from Kyushu Institute of Technology, Japan, in 2014. Since 2015, she has been with Nara Institute of Science and Technology (NAIST), Japan, as a full-time Assistant Professor. Her research interests include digital hardware circuit design and algorithms related to wireless communication, communication security, blockchain technologies, SHA-2, SHA-3, and cryptography. She is a Regular Member of IEEE, IEICE, and REV-JEC.

Hoai Luan Pham (Member, IEEE) received the bachelor's degree in computer engineering from Vietnam National University Ho Chi Minh City-University of Information Technology (VNUHCM-UIT), Vietnam, in 2018, and the master's degree in information science from Nara Institute of Science and Technology (NAIST), Japan, in 2020, where he is currently pursuing the Ph.D. degree. His research interests include blockchain technology and cryptography.

Tri Dung Phan received the bachelor's degree in computer engineering (hardware design) from Vietnam National University Ho Chi Minh City-University of Information Technology (VNUHCM-UIT) in 2019. He is currently pursuing the master's degree in computer architecture. His research interest includes secure hash algorithm (SHA) in hardware design, including FPGA and ASIC design.

Yasuhiko Nakashima (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer engineering from Kyoto University in 1986, 1988, and 1998, respectively. He was a Computer Architect with the Department of Computer and System Architecture, FUJITSU Ltd., from 1988 to 1999. From 1999 to 2005, he was an Associate Professor with the Graduate School of Economics, Kyoto University. Since 2006, he has been a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include computer architecture, emulation, circuit design, and accelerators. He is a fellow of IEICE, a Senior Member of IPSJ, and a Member of IEEE CS and ACM.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 22, 2021, accepted January 15, 2022, date of publication January 25, 2022, date of current version February 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3146148

A High-Efficiency FPGA-Based Multimode SHA-2 Accelerator

HOAI LUAN PHAM[1], (Graduate Student Member, IEEE),
THI HONG TRAN[2], (Member, IEEE),
VU TRUNG DUONG LE[1], (Graduate Student Member, IEEE),
AND YASUHIKO NAKASHIMA[1], (Senior Member, IEEE)
[1]Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Nara 630-0192, Japan
[2]Graduate School of Engineering, Osaka City University, Osaka 558-8585, Japan

Corresponding author: Thi Hong Tran (hong@osaka-cu.ac.jp)

This work was supported by the Japan Science and Technology Agency (JST) under a Strategic Basic Research Programs Precursory Research for Embryonic Science and Technology (PRESTO) under Grant JPMJPR20M6.

ABSTRACT The secure hash algorithm 2 (SHA-2) family, including the SHA-224/256/384/512 hash functions, is widely adopted in many modern domains, ranging from Internet of Things devices to cryptocurrency. SHA-2 functions are often implemented on hardware to optimize performance and power. In addition to the high-performance and low-cost requirements, the hardware for SHA-2 must be highly flexible for many applications. This paper proposes an SHA-2 hardware architecture named the multimode SHA-2 accelerator (MSA), which has high performance and flexibility at the system-on-chip level. To achieve high performance and flexibility, our accelerator applies three optimal techniques. First, a multimode processing element architecture is proposed to enable the accelerator to compute various SHA-2 functions for many applications. Second, a three-stage arithmetic logic unit pipeline architecture is proposed to reduce the critical paths and hardware resources. Finally, nonce generator and nonce validator architectures are proposed to reduce memory access and maximize the performance of the proposed MSA for blockchain mining applications. The MSA accuracy is tested on a real hardware platform (the Xilinx Alveo U280 FPGA). The experimental results on the field programmable gate array (FPGA) prove that the proposed MSA achieves significantly better performance, hardware efficiency, and flexibility than previous works. The evaluation results for energy efficiency show that the proposed MSA achieves up to 38.05 Mhps/W, which is 543.6 and 29 times better than the state-of-the-art Intel i9-10940X CPU and RTX 3090 GPU, respectively.

INDEX TERMS SHA-2, blockchain mining, FPGA, multimode, Bitcoin, accelerator.

I. INTRODUCTION

The Secure Hash Algorithm (SHA) published by the National Institute of Standard and Technology (NIST) [1] has three families of cryptographic hash functions, including SHA-1, SHA-2, and SHA-3. Currently, SHA-1 is deprecated due to its found vulnerabilities [2]. SHA-2 was firstly introduced in 2001 as an inevitable alternative to SHA-1. SHA-3 is the newest generation published by NIST in 2015 [3]. However, SHA-3 has not yet reached widespread diffusion because of two main reasons. First, there was no significant vulnerability to SHA-2 has been found yet. Second, the hardware architecture of SHA-3 is completely different from that of SHA-2, while most of the systems nowadays have been secured by SHA-2. Replacing SHA-2 by SHA-3 will require a huge investment in new hardware infrastructure to support SHA-3. For these reasons, SHA-2 and SHA-3 become two independent research themes that are conducted in parallel. Systems relied on old infrastructures intend to use SHA-2, while the completely new system may consider applying SHA-3. Therefore, SHA-2 is still one of the most reliable hash functions for long-term collision resistance and is widely used today. In particular, SHA-224, SHA-256, SHA-384, and SHA-512 are the most famous hash functions of the SHA-2 family and are widely used in many generic security applications, such as hash-based message authentication codes [4]–[6], error detection and correction (EDAC) [7], digital signature algorithms (DSAs) [8],

The associate editor coordinating the review of this manuscript and approving it for publication was Vincenzo Conti.

pseudorandom number generators (PRNGs) [9], RFID [10] and trusted computing [11]. Beyond generic applications, SHA-2, especially SHA-256, is chosen as the underlying hash function in blockchain, the modern technology behind well-known cryptocurrencies such as Bitcoin [12].

A. GENERIC APPLICATIONS

In network security, client devices may be sufficiently powerful to execute a limited number of hash computations, while servers often perform many hash computation tasks with various SHA-2 functions to serve authentication requests from clients. Thus, the server side needs SHA-2 hardware that has high performance and flexibility to perform a large number of hash computations with various hash functions [13]. In addition, with the development of modern technology such as the Internet of Things (IoT), data security for millions of devices increases the processing requirements for central servers. To reduce the processing pressure on servers, edge computing has recently been used to share hash computing requirements from IoT devices. Thus, edge computing also needs SHA-2 hardware with high performance and flexibility to execute a large number of hash computations. For the above reasons, developing a high-performance and flexible SHA-2 hardware accelerator has become a current research trend.

B. BLOCKCHAIN APPLICATIONS

SHA-2 functions play a crucial role in blockchain, an emerging technology used in many famous cryptocurrencies, such as Bitcoin, Litecoin, and Ethereum [14]. Among the hash functions of the SHA-2 family, SHA-256 is commonly used in many blockchains [15]. For example, SHA-256 is used to build Merkle trees that help the blockchain network maintain the integrity of transactions [16]. The most prominent use of SHA-256, particularly double SHA-256, is the hash computation in the mining process for blockchain networks, the most well-known of which is Bitcoin. Accordingly, the blockchain mining process adds a new valid block to the chain of blocks by hashing a block header, which includes values such as the previous block hash, Merkle root hash, timestamp, target, and *nonce*. For a new block to be considered valid, miners must find a valid *nonce* to make the hashing output value less than the target [17]. To quickly determine the valid *nonce* and win the reward, miners often use an ultrahigh-performance double SHA-256 circuit to speed up the hash computation of the block header. The double SHA-256 circuit must be fast enough to compete favorably in a blockchain network and be power efficient so that the energy costs do not exceed the mining revenue [18], [19]. Therefore, developing high-processing-rate double SHA-256 hardware with high hardware efficiency has recently become an attractive research area.

Conventional works have applied many techniques or proposed new architectures to optimize the performance of SHA-2 hardware. For example, the authors of [20]–[22] applied the pipeline technique to shorten the critical path in the SHA-256 and SHA-512 hardware. The authors of [23]

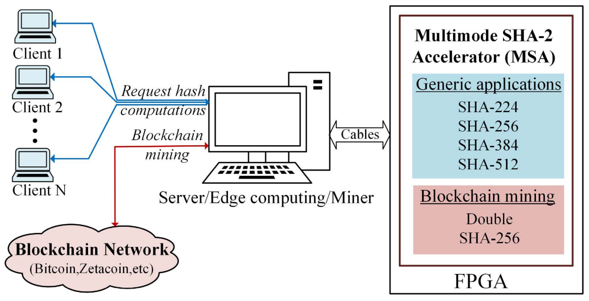


FIGURE 1. High-level diagram of the proposed system.

proposed the reordering computation method to reduce the critical path of the SHA-256 circuit. An unrolling technique with multiple factors was proposed in [24] and [25] to reduce the delay of the SHA-2 loop, thereby increasing the throughput. In [26]–[30], several hardware techniques, such as CSA, unrolling, and pipelining, were applied to SHA-2 accelerators to increase throughput. Although the performance of the accelerators in [20]–[30] was effectively optimized, these accelerators still deliver poor performance and are not compatible with high-speed SHA-2 applications. To address speed-demanding applications, the authors of [31]–[38] proposed several new hardware architectures to achieve high performance for SHA-2 computations. For instance, the authors of [31]–[36] proposed a full pipeline architecture to accelerate SHA-256 computation for blockchain mining. In addition, a multicore architecture was proposed in [37], [38] to perform multiple SHA-256 processes simultaneously, thereby achieving high performance. Despite the advantage of a high processing rate, the accelerators in [31]–[38] have no flexibility because they can only execute a single hash function, such as the SHA-256 function. Overall, the accelerators in [20]–[38] need to improve performance and flexibility to be compatible with multiple SHA-2 applications, ranging from generic applications to blockchain mining.

This work proposes a multimode SHA-2 accelerator (MSA) that achieves a high processing rate and flexibility for generic applications and blockchain mining. The high-level diagram of the proposed system is shown in Fig. 1, where the proposed MSA is applied to support servers, edge computing nodes, or miners to perform high-speed computations with high flexibility. Concretely, the server or edge computing node can employ the proposed MSA to perform a large number of hash computations with a variety of hash functions, including SHA-224, SHA-256, SHA-384, and SHA-512. In addition, miners can adopt our accelerator to accelerate the double SHA-256 calculation for blockchain mining process.

To achieve the high processing rate and flexibility for multiple applications, the proposed MSA employs several optimization techniques, such as multiple multimode processing elements (M-PE), dual arithmetic logic unit (ALU) architecture inside each M-PE, a nonce generator (NOG), and a nonce detector (NOD). The impact of those

TABLE 1. **Parameters of the SHA-2 functions.**

Parameter	SHA-224	SHA-256	SHA-384	SHA-512
Block size (S) (bit)	512	512	1024	1024
Max message length (bit)	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
# of selected words (nH)	7	8	6	8
Word size (D) (bit)	32	32	64	64
Output size (D*nH) (bit)	224	256	384	512
# of rounds (R)	64	64	80	80

optimization techniques is analyzed and evaluated in this paper. The implementation and verification of the proposed MSA on the Xilinx Alveo U280 field programmable gate array (FPGA) for general applications and blockchain mining are explicitly presented. The experimental results on the FPGA show that the proposed MSA is better than state-of-the-art works in terms of performance, hardware efficiency, and flexibility. Compared to the current most powerful CPU and GPU, the FPGA-based MSA is better than the Intel i9-10940X CPU and RTX 3090 GPU in terms of power efficiency.

The remainder of this paper is organized as follows: Section II presents the background. Section III describes our proposed multimode SHA-2 accelerator in detail. Section IV presents the implementation, verification, and evaluation of the proposed MSA on the FPGA. Finally, Section V concludes the paper.

II. BACKGROUND

This section briefly describes basic information about the SHA-2 functions for generic applications and blockchain mining. Additionally, the preliminary ideas for the proposed MSA are clearly analyzed.

A. SHA-2 FUNCTIONS FOR GENERIC APPLICATIONS

SHA-2 is a set of one-way and collision-resistant cryptographic hash functions. The SHA-2 family consists of six hash functions, namely, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. Because the SHA-512/224 and SHA-512/256 functions are truncated versions of SHA-512 and are not widely used, we focus on only the first four hash functions, SHA-224, SHA-256, SHA-384, and SHA-512. These four hash functions are essentially the same in terms of operational process, but they have differences in parameters, which are shown in Table 1. Based on the similarities of the parameters and operational processes, the SHA-2 hashing algorithms are divided into two main groups: SHA-224/256 (SHA-224 or SHA-256) and SHA-384/512 (SHA-384 or SHA-512). Algorithm 1 shows the

```
Algorithm 1 Hash = SHA-2(Message)
1:  SHA224/256:
2:    R = 64, S = 512, Llen = 64, D = 32, nH = 7/8
3:    M consists of N 512-bit padded blocks.
4:    $H^0_{[0:7]}$ : 32-bit square root of the first 8 primes.
5:    $K_{[0:63]}$: 32-bit square root of the first 64 primes.
6:  SHA384/512:
7:    R = 80, S = 1024, Llen = 128, D = 64, nH = 6/8
8:    M consists of N 1024-bit padded blocks.
9:    $H^0_{[0:7]}$ : 64-bit square root of the first 8 primes.
10: $K_{[0:79]}$: 64-bit square root of the first 80 primes.
11: $L_{[0:Llen-1]}$ = length_in_bit(message)
12: N = (L ÷ S) + 1
13: ($M^{[0:N-1]}$, N) = Padding (message)
    Padding:
14: k = S − (1 + D + (L  mod S))
15: Pad = {1, zeros(1,k), L}
16: $M^{[0:N-2]} = message[0 : ((N-2) * S) - 1]$
17: $M^{N-1} = \{message[(N-2) * S : L-1], Pad\}$
18: for t ← 0 to (N-1) do
      $W_{[0:R-1]}$ = Message_Expansion($M^t$)
      Message Expansion:
19:   for i ← 0 to (R-1) do
20:     if i < 16 then
21:       $W_i = M^t_{[i*D:(i+1)*D]}$
22:     else
23:       $W_i = W_{i-16} + \sigma_0(W_{i-15}) + W_{i-7} + \sigma_1(W_{i-2})$
24:     end if
25:   end for
      $H^{t+1}$ = MessageCompression($H^t$, K, W)
      Message Compression:
26:   $a = H^t_0, b = H^t_1, c = H^t_2, d = H^t_3$
      $e = H^t_4, f = H^t_5, g = H^t_6, h = H^t_7$
27:   for i ← 0 to (R-1) do
28:     $T_1 = h + \Sigma_1(e) + Ch(e,f,g) + K_i + W_i$
29:     $T_2 = h + \Sigma_0(a) + Maj(a,b,c)$
30:     $h = g, g = f, f = e, e = d + T_1,$
        $d = c, c = b, b = a, a = T_1 + T_2$
31:   end for
32:   $H^{t+1}_0 = H^t_0 + a, \ldots H^{t+1}_7 = H^t_7 + h$
33: end for
34: return Hash = $\{H^N_0, \ldots, H^N_{nH-1}\}$
```

SHA-2 algorithm pseudocode. It includes three main steps: padding, message expansion, and message compression.

1) PADDING

The padding process is performed to make the last block have the same size as the other blocks. Concretely, the original message has L bits, and then the bit "1" is appended at the beginning bit and k zero bits at the remaining bits. The appended bits must satisfy the equation $L+1+k \equiv 448$ mod 512 for SHA-224/256 functions or the equation $L+1+k \equiv$ 896 mod 1024 for SHA-384/512 functions. Then, the padded message is divided into N blocks ($M^{[1:N]}$) of S-bit size, where S is 512 for SHA-224/256 and 1024 for SHA-384/512.

2) MESSAGE EXPANSION

After padding, all blocks ($M^{[1:N]}$) have a fixed length of S bits. Each block is compressed through two processes: message expansion (ME) and message compression (MC). Both ME and MC processes include R loops, where R is 64 for SHA-224/256 and 80 for SHA-384/512. Sixteen chunks of the 32/64-bit word (denoted as $W_i, 0 \leq i \leq 15$) parsed from the t^{th} block (denoted as M^t) are compressed in the first 16 loops of the MC process. The ME process expands the message input (M^t) to the R-16 chunks of the 32/64-bit W_i ($16 \leq i \leq R-1$) required in the last R-16 loops of the MC process.

3) MESSAGE COMPRESSION

Basically, the MC process compresses the R chunks of the 32/64-bit W_i ($0 \leq i \leq R-1$) from the ME process into a 224/256/384/512-bit hash output. The MC process involves three main steps: *initialization*, *compression*, and *final adding*. In the *initialization* step, eight internal hash values (denoted as a, b, c, d, e, f, g, h) are assigned to the eight hash inputs $H_0^t, H_1^t, \ldots, H_7^t$. Note that in the MC process for the first block (M^0, t = 0), the eight hash inputs are the eight H constants ($H^0_{[0:7]}$: eight 32- or 64-bit decimal places of the square roots of the first eight primes). In the *compression* step, the eight internal hash values $a, b, \ldots, h$ are computed and updated through R loops. In the *final adding* step, the hash output (H^{t+1}) is updated by adding the eight internal hash values $a, b, \ldots, h$ to the eight hash inputs $H_0^t, H_1^t, \ldots, H_7^t$. After finishing the ME and MC process for block M^{t+1}, the H^{t+1} value is used as the hash input in the MC process for the next block (M^{t+1}). Finally, the concatenation of the hash output H^N updated by compressing the last block (M^{N-1}) is the final hash output of the hash algorithm.

The details of the logical functions $\sigma_0(x)$, $\sigma_1(x)$, $\Sigma_0(x)$, $\Sigma_1(x)$, $Ch(x, y, z)$, and $Maj(x, y, z)$ in the ME and MC processes can be found at [39]. Note that the logical functions $\sigma_0(x)$, $\sigma_1(x)$, $\Sigma_0(x)$, and $\Sigma_1(x)$ are different between SHA-224/SHA-256 and SHA-384/SHA-512. Algorithm 1 uses parameters to distinguish the hash functions of the SHA-2 family. The most typical parameters, such as S, nH, D, and R, are presented in Table 1. In addition, the parameter $Llen$ is used to determine the length of the L string, where the L string is a bit string representing the length of the input message in bits padded to the last block.

In practice, the storage of the R-16 chunks ($W_{[16:R-1]}$) in the last R-16 loops of the ME process will occupy a large amount of memory. To reduce hardware resources, most previous works, such as [27], [29], [40], employed a shift-register method for the message expansion calculation, which uses only sixteen 32/64-bit registers to store the last 16 chunks, and the sixteen 32/64-bit registers must shift continuously during the loop calculation. Therefore, this paper also applies the shift-register method to reduce hardware resources but does not consider it a contribution.

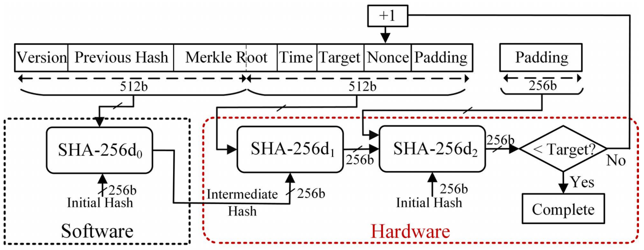


FIGURE 2. Double SHA-256 architecture for blockchain mining.

B. DOUBLE SHA-256 FOR BLOCKCHAIN MINING

The most famous application of SHA-2 is Bitcoin cryptocurrency. Essentially, Bitcoin operates based on blockchain technology, which uses the double SHA-256 (SHA-256d) to validate transactions. Concretely, blockchain technology stores transactions in a block, and then blocks are linked together to become a chain of blocks known as ledgers [41]. To add the new block to the ledger, miners in the blockchain network compete for the SHA-256d computation of block headers as a proof of work (PoW) to find a valid block and receive a decent reward, commonly called blockchain mining. SHA-256d is not a variant hash function of the SHA-2 family but calculates SHA-256 twice. For example, SHA-256d(x) is equivalent to SHA-256(SHA-256(x)). In blockchain mining, SHA-256d is used to prevent length extension attacks [42].

Fig. 2 illustrates the overview architecture of SHA-256d for blockchain mining. Specifically, the message input to the SHA-256d computation is the 1024-bit block header, including a 32-bit version, a 256-bit hash of the previous block, a 256-bit hash of the Merkle root, a 32-bit timestamp, a 32-bit target, a 32-bit *nonce*, and 384-bit padding. The 1024-bit message is divided into two 512-bit messages. Then, SHA-256d$_0$ computes the first 512-bit message, and SHA-256d$_1$ calculates the final 512-bit message. Due to the double SHA-256 requirement, SHA-256d$_2$ compresses the 256-bit hash output from SHA-256d$_1$. In blockchain mining, the final hash output from SHA-256d$_2$ is compared with the target hash to determine the valid *nonce*. If the final hash output is smaller than the target hash, the valid *nonce* will be determined, and a new block will be added to the ledger. Otherwise, the *nonce* is increased by one to create the new 1024-bit message for the SHA-256d computation again. Because of the infrequent change of the first 512-bit message, SHA-256d$_0$ is regularly computed at the software level. Meanwhile, the *nonce* value has to be tried billions of times to find a valid *nonce*, causing the final 512-bit message to change continuously. Thus, the computation of SHA-256d$_1$ and SHA-256d$_2$ should often target hardware design for performance optimization.

C. PRELIMINARY IDEA FOR THE MSA

There are three characteristics of SHA-2 functions that should be noted. First, SHA-2 functions use only low-cost arithmetic logic operators, such as adders, rotations, shifts, and XORs. There are no complex operators, such as multipliers, dividers, and exponents. Second, the number of operators per loop calculation is quite large, specifically, approximately 50 operators. Third, the data among loops have high dependencies. For example, the $(i+1)^{th}$ loop calculation needs the results of the i^{th} loop calculation. Because of these three characteristics, high-performance hardware platforms such as CPUs and GPUs do not efficiently execute the SHA-2 computation. On the other hand, the memory blocks of the CPUs and GPUs, such as double data rate (DDR) memory and caches, are located far away from the computational units. Thus, the data transfer time between memories and computational units can constitute a large amount of the total processing time, which reduces the processing rate. Although CPUs and GPUs have multiple cores to perform a large number of hash computations in parallel to achieve high performance, they often suffer from large power consumption, resulting in limited energy efficiency.

In another approach, state-of-the-art FPGA-based SHA-2 accelerators are developed to be compatible with the three characteristics of SHA-2 functions, thus significantly improving the area and energy efficiency. However, these accelerators can only execute either SHA-256 or SHA-512 and lack flexibility. The reason is that the calculations in the SHA-2 functions use different word sizes (32-bit or 64-bit words), and it is challenging for these accelerators to calculate both 32-bit and 64-bit words. Moreover, most FPGA-based accelerators focus only on improving a single computational block and overlook developing an architecture for a large amount of hash computation. Thus, these accelerators often have poor performance when performing multiple hash calculations.

To be applicable for generic applications and blockchain mining, the SHA-2 hardware architecture should be high-performance and flexible (supporting various SHA-2 functions) with high hardware efficiency. However, there has been no high-performance and flexible SHA-2 hardware until now.

In this study, we develop an MSA that achieves high performance and flexibility with high hardware efficiency by eliminating the weaknesses of CPUs, GPUs, and state-of-the-art FPGA-based accelerators. There are three ideas in the proposed MSA to achieve this purpose. **Idea 1: A multimode processing element with dual ALUs**. Since the smallest word size in the SHA-2 functions is 32 bits, the ALU is proposed to perform the 32-bit word calculations. In the ALU, registers (considered local memory) are located near computational units to reduce the data transfer time. There is a problem that a single ALU cannot perform the SHA-384/512 functions because the calculations in SHA-384/512 functions use 64-bit words. In addition, a single ALU is insufficient to execute double SHA-256 computation for blockchain mining. To solve these problems, we use dual ALUs that can be concatenated to create one ALU for calculations of 64-bit words. In another approach for the concatenation of dual ALUs, the output of the first ALU is transferred to the input of the second ALU to construct

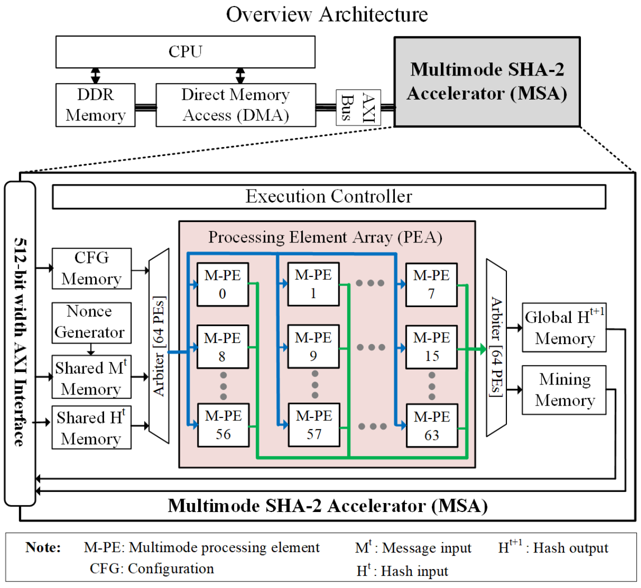


FIGURE 3. Overview architecture of the proposed multimode SHA-2 accelerator at the system-on-chip level.

a double SHA-256 circuit for blockchain mining. Moreover, dual ALUs can execute two independent SHA-224/256 functions in parallel to double the processing rate. Because dual ALUs can improve the performance and flexibility of the MSA, dual ALUs are located inside each processing element PE) of the MSA. By using dual ALUs, the PE can execute multiple SHA-2 functions (modes); thus, it is called a multimode processing element (M-PE). **Idea 2: Pipelined dual-ALU architecture**. Although only low-cost arithmetic logic operators are employed, the dual ALUs must use a large number of operators for the loop calculation, approximately 50 operators. This means that the dual ALUs suffer from a very long critical path, resulting in a low processing rate. To shorten the critical path, we employ the pipeline technique for the dual ALUs. Accordingly, the dual ALUs have three-stage pipelines, and the computational workload is balanced for each stage. Moreover, the carry-save adder (CSA) technique is also applied for the dual ALUs to reduce the critical path and hardware resources. **Idea 3: Nonce generator (NOG) and nonce detector (NOD) mechanisms**. In blockchain mining, the MSA must scan all possible values of 2^{32} 32-bit *nonces* to find the valid *nonce*. To scan and verify one *nonce* value, the accelerator must exchange at least 1,280-bit data (the 512-bit message input to SHA-256d_1, the 256-bit hash input to SHA-256d_1, the 256-bit hash input to SHA-256d_2, and the 256-bit hash output) with DDR memory. However, the bandwidth transmission between DDR memory and the accelerator is limited, which creates a long data transfer time, thus causing the total processing time to be very large. Optimizing the accelerator performance for blockchain mining will be meaningless if the bandwidth transmission between DDR memory and MSA is bottlenecked. Therefore, NOG and NOD mechanisms are proposed to solve this problem. Concretely, the NOG can automatically generate up to

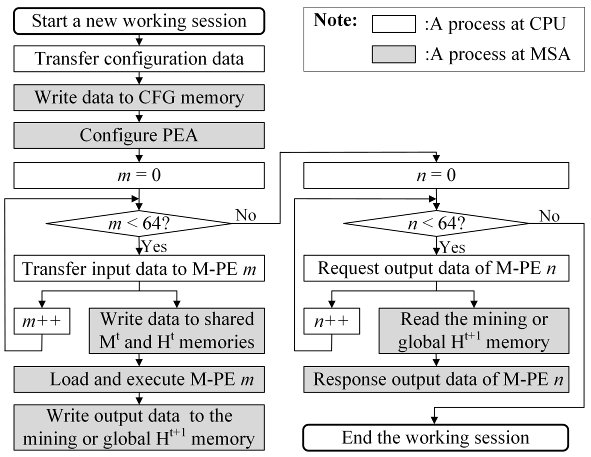

Note:
:A process at CPU
:A process at MSA
Start a new working session
Transfer configuration data
Write data to CFG memory
Configure PEA
m = 0
m < 64?
No
Yes
Transfer input data to M-PE m
m++
Write data to shared M^t and H^t memories
Load and execute M-PE m
Write output data to the mining or global H^t+1 memory
n = 0
n < 64?
Request output data of M-PE n
n++
Read the mining or global H^t+1 memory
Response output data of M-PE n
End the working session

FIGURE 4. High-level flowchart of the working session on FPGA.

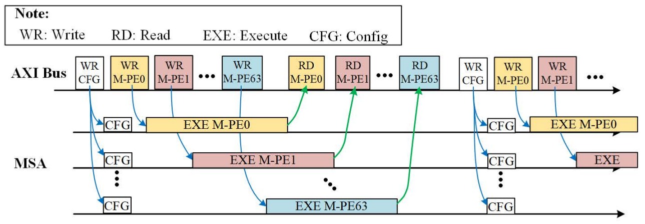


FIGURE 5. Timing chart of the multimode processing element execution.

2^{32} *nonce* values, equivalent to creating 2^{32} message inputs to the SHA-256d computation. On the other hand, the NOD can automatically verify the hashing output to find a valid *nonce* value inside each M-PE. Thanks to the NOG and NOD mechanisms, the MSA performance for blockchain mining is not reliant on the transmission bandwidth between the DDR memory and the accelerator, thus achieving 100% hardware efficiency.

III. PROPOSED MULTIMODE SHA-2 ACCELERATOR

A. OVERVIEW ARCHITECTURE

Fig. 3 shows the overview architecture of the proposed MSA at the system-on-chip (SoC) level. The CPU is responsible for controlling the operations of the entire system. In the task of controlling the proposed accelerator, the CPU sends a request to direct memory access (DMA) to transfer data from the DDR memory to the MSA, where the MSA connects to DMA via the advanced extensible interface (AXI) bus. The communication between the CPU and the proposed MSA is separated into many working sessions. Each working session of the proposed system is shown in Fig. 4. Concretely, at the start of a new session, the CPU transfers configuration data to the proposed MSA. The configuration data are written to CFG memory and then are used to configure the hash function mode of the processing element array (PEA). Afterward, the CPU sends the input data to the proposed accelerator, including the message and hash inputs. Notably, the input data transfer is executed in parallel with the hash computation of the

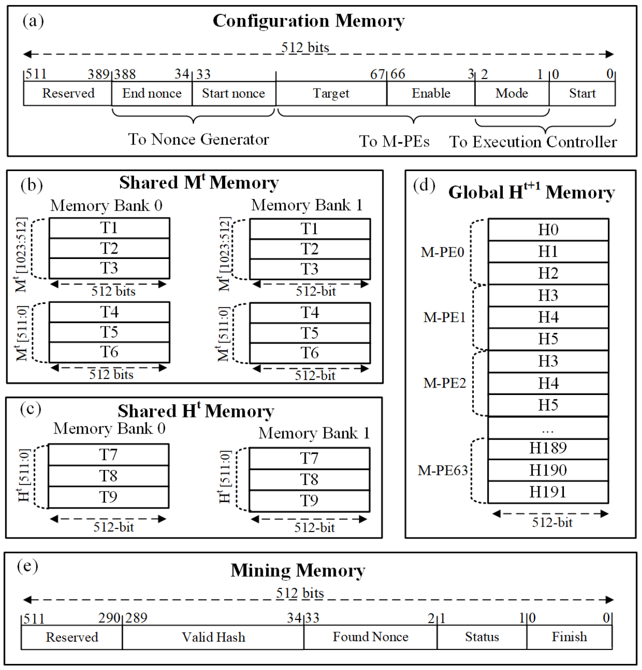


FIGURE 6. The memory organization of the MSA: (a) config memory; (b) shared W^t memory; (c) shared H^t memory; (d) global H^{t+1} memory; (e) mining memory.

proposed MSA to accelerate the total processing rate. After the completion of the hash computations, the hash outputs cannot be immediately transferred to DDR memory but must wait for a request from the CPU. Therefore, we develop a global hash output memory to store the hash outputs to reduce the number of DDR memory requests and increase the processing rate. In addition, mining memory is developed to store the valid nonce and hash output for blockchain mining. After the CPU finishes reading output data from global hash output or mining memories, the working session is completed.

The proposed MSA consists of four main components: the processing element array (PEA), memory, NOG, and execution controller. The four components are presented as follows: **First**, the PEA is the key component of the proposed MSA that accelerates the hash computation with various hash functions. The PEA includes sixty-four M-PEs, which are designed to perform hash computations in pipeline and parallel, as shown in Fig. 5. When the AXI bus is writing and reading data to and from an M-PE, the other M-PEs of the MSA are still executing the hash computation. Accordingly, the data transfer time between the DDR memory and the proposed MSA will not affect the total processing rate of the system if the AXI bus in the system is fast enough. In our system, we use an AXI bus with a 512-bit data width to improve the transfer data time between the DDR memory and the accelerator. **Second**, there are five types of memory, including configuration memory, shared message (M^t) memory, shared hash input (H^t) memory, global hash output (H^{t+1}) memory, and mining memory, to store the configuration data, message, hash input, hash output, and mining results, respectively. Fig. 6 presents the organization of the five types of

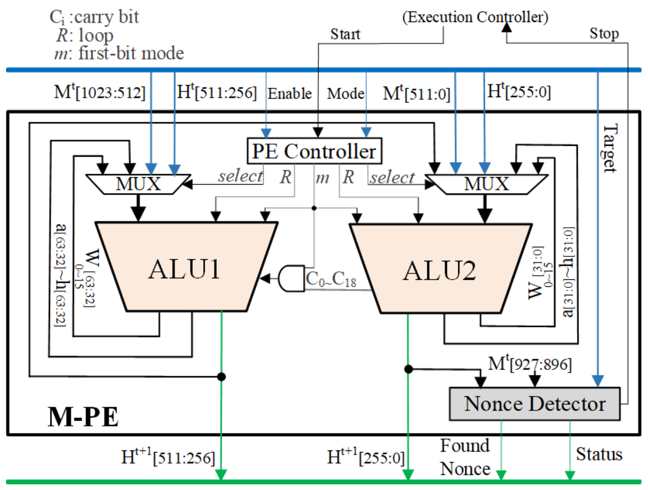


FIGURE 7. Multimode processing element (M-PE) architecture.

TABLE 2. Operating mode of dual ALUs.

No.	Mode	Operating mode	
		ALU1	ALU2
0	00	SHA224/256	SHA224/256
1	01	SHA384/512	
2	10	SHA256d$_1$	SHA256d$_2$

memory. As shown in Fig. 6 (a), the 512-bit configuration memory stores configuration information for the execution controller, NOG, and M-PEs. In Fig. 6 (b) and (c), we present the organization of the shared M^t and H^t memories. Because M-PEs operate in parallel and pipeline, only one M-PE receives the message and hash input data at a time. Thus, the shared H^t and H^t memories need to store only enough message and hash input data for one M-PE to minimize the hardware resources. To continuously write data from the AXI bus and read data to load to the M-PEs without collision, the shared M^t and H^t memories are designed with two memory banks according to the ping-pong memory mechanism [43]. In particular, while memory bank 0 writes data from the AXI bus, memory bank 1 reads data to load to the M-PEs, and vice versa. Since the dual ALUs inside the M-PE are developed in the three-stage pipeline to execute three hash computations in parallel, the shared M^t and H^t memories must be designed to store sufficient messages and hash inputs. Specifically, the six 512-bit transactions (denoted T1 to T6) stored in the shared M^t memory are three 1024-bit message inputs, and the three 512-bit transactions (denoted T7 to T9) stored in the shared H^t are three 512-bit hash inputs. In Fig. 6 (d), we present the global H^{t+1} memory used to store 192 512-bit hash outputs (denoted H0 to H191) from sixty-four M-PEs. As shown in Fig. 6 (e), the mining memory is used to store the valid hash output, found nonce value, status flag (equal to 0 if no valid nonce is found and equal to 1 if the valid nonce is found), and finish flag when the proposed MSA performs the blockchain mining task. **Third**, the NOG block is used to automatically generate up to 2^{32} nonce values, which are employed to update the 2^{32} messages to the SHA-256d computation for blockchain mining. The details of the NOG are described in Section III-D. **Fourth**, the execution controller controls the operations of the PEA, memories, and NOG.

B. MULTIMODE PROCESSING ELEMENT (M-PE) ARCHITECTURE

In the PEA, the processing elements are named multimode processing elements because they are designed to perform multiple SHA-2 functions for generic applications and blockchain mining. In this section, the M-PE architecture is clarified.

Fig. 7 illustrates the multimode processing element architecture with dual ALUs. Basically, each ALU executes the 32-bit word calculations in the message expansion and compression processes of the SHA-224/256 functions. However, one ALU cannot perform the SHA-384/512 computations because the SHA-384/-512 functions require 64-bit word calculations. Therefore, it is proposed that each M-PE uses dual ALUs that can be concatenated to perform 64-bit word calculations. The dual ALUs are ALU1 and ALU2, where ALU1 and ALU2 obtain the 32 most significant bits (MSBs) and the 32 least significant bits (LSBs) in the 64-bit word calculations, respectively. Additionally, ALU1 and ALU2 can perform two independent 32-bit word calculations in parallel to double the processing rate of the SHA-224/256 functions. For ALU1 and ALU2 to correctly perform both 32-bit and 64-bit word calculations, the 32-bit and 64-bit arithmetic logic operators for the calculations are processed as follows: The two 32-bit bitwise logic operators in ALU1 and ALU2 can be concatenated to create one 64-bit bitwise logic operator because the bitwise logical operators, such as AND, OR, and XOR, examine one bit at a time. In the shift and rotation logic operators, two 32-bit operators and one 64-bit operator execute in parallel, and the results are then selected by a multiplexer gate. In the arithmetic operator, the two 32-bit adders in ALU1 and ALU2 can be concatenated to form one 64-bit adder by turning the 32^{nd} carry bit of the adder in ALU2 on or off. Overall, using dual ALUs, the M-PE can execute two SHA-224/256 functions in parallel or perform one SHA-384/512 function with no wasted hardware resources.

In each M-PE, the PE controller controls the concatenation of the arithmetic logic operators in ALU1 and ALU2 by the first bit of the two-bit mode (denoted as m) received from the configuration memory. In addition to concatenating the arithmetic logic operators, ALU1 and ALU2 can be concatenated to create a double SHA-256 (SHA-256d) circuit for blockchain mining. Accordingly, the hash output of the SHA-256d$_1$ computation in ALU1 is transferred to the message input to the SHA-256d$_2$ computation in ALU2. The M-PE uses the second bit of the two-bit mode to configure ALU1 and ALU2 as the SHA-256d circuit. As a result, ALU1 and ALU2 can execute the various hash functions for generic applications and blockchain mining, configured by a two-bit

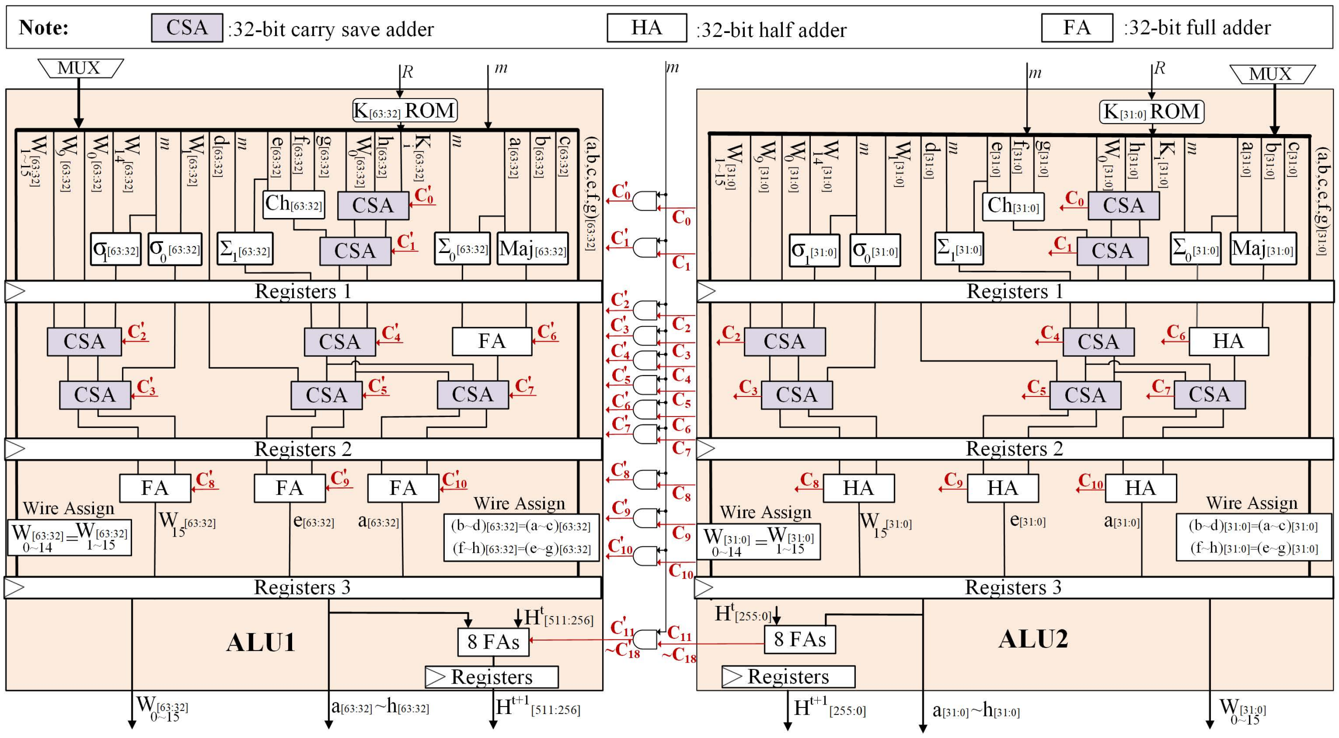


FIGURE 8. The three-stage pipelined dual ALU architecture.

mode received from the configuration memory, as shown in Table 2. On the other hand, each M-PE can be activated or deactivated by an *enable* signal from the configuration memory to reduce the redundant power consumption. The power overhead for the unused M-PEs is diminished by the clock gating technique.

To optimize this system for blockchain mining, we propose a NOD in each M-PE to find the hash output of the SHA-256d$_2$ computation less than the target threshold, which is used to determine the *valid nonce*. The detailed presentation of the NOD is described in Section III-D.

C. PIPELINED DUAL-ALU ARCHITECTURE

The dual-ALU architecture is an iteration structure, requiring 64 or 80 loops to generate the hash output. Consequently, the dual ALUs must contain all operators for one loop calculation of the ME and MC processes. However, a large number of operators in the dual ALUs can cause a long critical path, resulting in a significantly limited processing rate. Therefore, we propose using the pipeline technique for the dual ALU architecture to reduce the critical path and improve the processing rate.

Fig. 8 shows a three-stage pipelined dual ALU architecture. According to this architecture, both the ME and MC processes in the dual ALUs are divided into three-stage pipelines, where the computational workload of each stage is balanced to achieve the lowest critical path. Since the adders have the highest computational cost, the path through the adders is the critical path in each stage. Therefore, this architecture

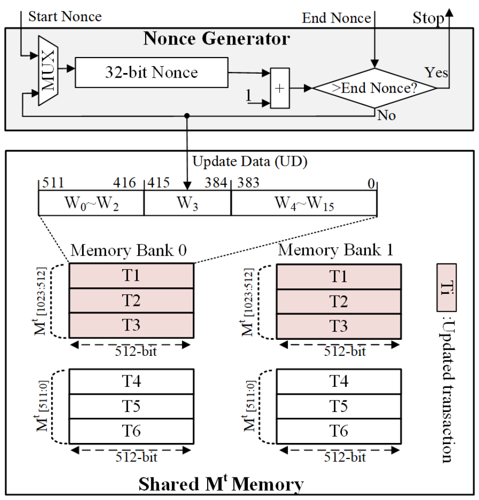


FIGURE 9. Nonce generator architecture.

replaces several full adders FAs) and half adders (HAs) with CSAs to reduce the critical path and hardware resources. Accordingly, the hardware can be improved to be at least 14% faster [44] when applying two CSAs to construct an adder of four operands.

With this architecture, the i^{th} loop calculation is executed through the three stages. The results of the i^{th} loop calculation are outputted from the third stage and then fed back to the first stage to perform the $(i+1)^{th}$ loop calculation. Thus, all 64 (at SHA-224/256) or 80 (at SHA-384/512) loops of

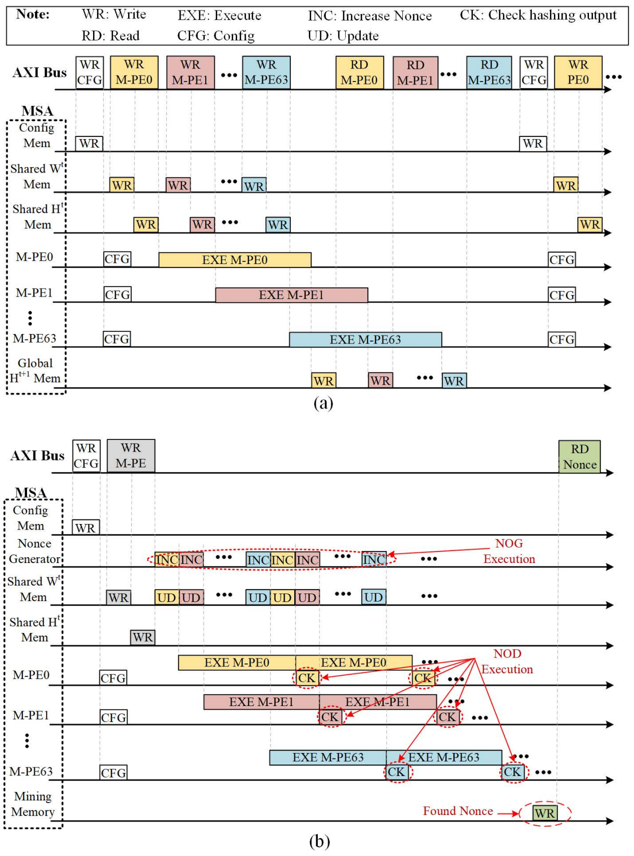


FIGURE 10. Detailed timing chart of the proposed MSA in (a) generic application and (b) blockchain mining.

the ME and MC processes can be completed in the dual ALUs. Note that the shift-register method is applied to the ME process, so we use only sixteen variables W_0, $W_1, \ldots$, and W_{15} to compute and update the W_i of the last 48 or 60 loops. To efficiently use 100% of the hardware resources of the dual ALUs, three data flows, including messages and hash inputs, from the shared M^t and H^t memories should be used as input data to the three-stage pipelined dual ALUs. The registers at three stages (denoted registers 1, 2, and 3) are used to store enough variables that the three stages can execute three data flows in parallel. Since all stages are always busy, the dual ALU architecture achieves 100% hardware efficiency. After completing the 64 or 80 loops, the three results of the MC process are added to the three hash inputs (H^t) to generate three hash outputs (H^{t+1}), which are then stored in the global H^{t+1} memory.

D. NONCE GENERATOR AND DETECTOR FOR BLOCKCHAIN MINING

In blockchain mining, the MSA should scan all possible instances of 2^{32} 32-bit *nonce* values, equivalent to calculating 2^{32} messages, to find a valid hash smaller than the target. Since the bandwidth between the DDR memory and the accelerator is limited, the writing time of the 2^{32} messages

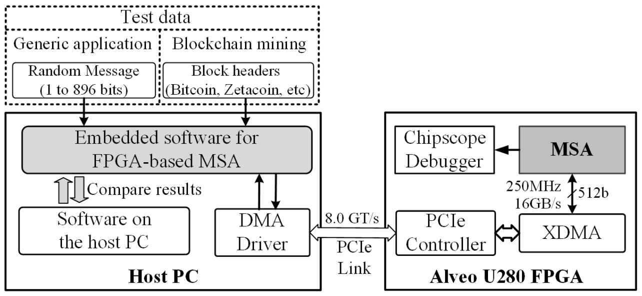


FIGURE 11. Implementation and verification of the proposed MSA on a Xilinx Alveo U280 FPGA.

and the reading time of the 2^{32} hash outputs is a bottleneck for the process of finding the nonce. Therefore, this section presents two mechanisms, NOGs and NODs, to improve the processing time.

The NOG automatically updates the *nonce* value inside the 512-bit messages in the shared M^t memory, as shown in Fig. 9. In each M-PE, the message to the SHA-256d$_1$ computation is performed in ALU1. According to our shared M^t memory organization, transactions T1-T3 are 512-bit messages to the SHA-256d$_1$ computation in ALU1. Based on our investigation, the *nonce* value is located at position W$_3$ of the messages to SHA-256d$_1$ in blockchain networks. Therefore, the NOG repeatedly updates the W$_3$ value, where W$_3$ is in bits 384 to 415 inside transactions T1-T3. So that it is user oriented, the NOG only generates *nonce* values between the *start nonce* and *end nonce* thresholds. The NOG will send the *stop* signal to the execution controller to stop the MSA operation if the generated nonce exceeds the *end nonce* threshold. At that time, the *finish* flag in the mining memory is valid for the CPU to check.

The NOD is used to compare the hash output of the SHA-256d$_2$ computation from ALU2 with the target value, as shown in Fig. 7. If the hash output is less than the target, the *status* flag, 32-bit found *nonce* and 256-bit hash output will be written to the mining memory. After that, the NOD will send the *stop* signal to the execution controller to stop the MSA operation and turn on the *finish* flag in the mining memory for the CPU to check.

To clarify the impact of the NOG and NOD, we present a detailed timing chart of the proposed MSA in generic applications and blockchain mining, as shown in Fig. 10. In generic applications, the accelerator performance is highly dependent on the AXI bus bandwidth, as shown in Fig. 10 (a). Specifically, the accelerator performance is low since the M-PEs have a long idle time to wait for writing and reading data. Thanks to the NOG and NOD mechanisms, writing and reading data between the DDR memory and the MSA are only performed once during the process of finding the nonce. Therefore, the M-PEs execute continuously with no idle time, thereby maximizing the performance for blockchain mining, as shown in Fig. 10 (b).

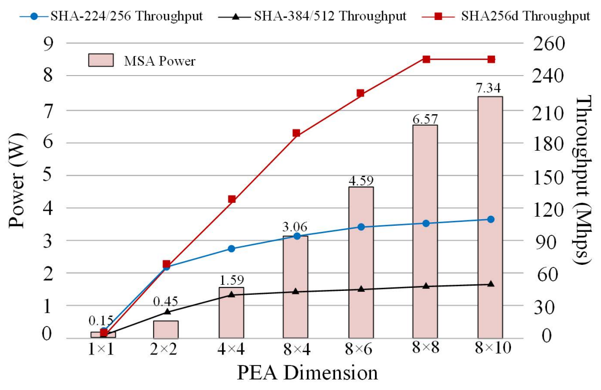


FIGURE 12. Power and throughput of the seven MSA versions with seven different processing element array (PEA) dimensions.

IV. VERIFICATION AND EVALUATION

In this part, the proposed architecture is verified, implemented, and evaluated on a Xilinx FPGA Alveo U280 Data Center Accelerator Card (Alveo U280 FPGA), which is a 16nm FPGA featuring more than 1,300k Look-Up Tables (LUTs) and 2,600k Flip-flops (FFs). Thanks to the huge resource of Alveo U280 FPGA, we can evaluate various MSA versions with different PEA dimensions to find the most suitable PEA size. Besides, the FPGA Alveo U280 board has PCIe Express 3.0, which can speed up the data transfer rate performance between the CPU and the FPGA-based MSA to 8.0 GT/s (equivalent to 32 GB/s).

A. FPGA-BASED MSA VERIFICATION

In this section, the proposed MSA is implemented and verified on the Xilinx FPGA Alveo U280 Data Center Accelerator Card, as shown in Fig. 11. The experimental devices are an Alveo U280 FPGA and a host PC with an Intel Xeon CPU E5-2620v2@2.10 GHz with 94 GB RAM. The proposed MSA is developed on the Alveo U280 FPGA (denoted as the FPGA-based MSA) and exchanges data with the host PC via a Xilinx PCI Express DMA (XDMA). To maximize the transmission bandwidth between the host PC's DDR memory and the FPGA, we use the XDMA with a performance of 8.0 gigatransfers per second (GT/s), which connects to the MSA via the 512-bit data width AXI bus. In the host PC, we design embedded software for the FPGA-based MSA to transmit test data and read the hash outputs. Regarding debugging, Chipscope ILA is added to the Alveo U280 FPGA to monitor the MSA signals. After the system-on-chip development, the FPGA-based MSA is verified for both generic applications and blockchain mining.

1) FPGA-BASED MSA VERIFICATION IN GENERIC APPLICATIONS

This section verifies the accuracy of the proposed MSA for the SHA-224, SHA-256, SHA-384, and SHA-512 computations, which are frequently performed in generic applications. Since the messages in generic applications are usually of an unknown length and value, the proposed MSA should be verified for hash computation with different bit sizes and message values. Therefore, the messages are randomly generated with various bit sizes and values for the FPGA-based MSA to compute in the SHA-224, SHA-256, SHA-384, and SHA-512 modes. The experiment is conducted with 100,000 random messages for each mode. For verification, the hashing output from the global H^{t+1} memory of the MSA is compared to the hashing results computed from the software in the host PC. The experimental results show that FPGA-based MSA works 100% correctly for the SHA-224, SHA-256, SHA-384, and SHA-512 computations.

TABLE 3. Performance comparison between two MSA architectures: MSA without the NOG and MSA with the NOG.

Design	Algorithm	Throughput (Mhps)	Power (W)	Energy Efficiency (Mhps/W)
MSA w/o NOG	SHA256d	62.5	6.53	9.57
MSA with NOG		**250**	**6.57**	**38.05**

2) FPGA-BASED MSA VERIFICATION IN BLOCKCHAIN MINING

This section verifies the correctness of the proposed MSA for the SHA-256d computation, which is used in blockchain mining. The 1024-bit message to the SHA-256d computation is obtained from the block headers in the blockchain network. For the sake of saving hardware resources, the 1024-bit message is computed in both the host PC (considered software) and FPGA-based MSA (considered hardware). The first 512-bit message to SHA-256d$_0$ is computed in the host PC. Then, the hashing output of SHA-256d$_0$ and the final 512-bit message to SHA-256d$_1$ are loaded to the FPGA-based MSA to find a valid 32-bit nonce. In the blockchain mining mode, the FPGA-based MSA executes the SHA-256d$_1$ and SHA-256d$_2$ computations until the valid *nonce* is found. For verification, the found valid *nonce* and hash output from the mining memory of the proposed MSA are compared with the available results on the website of the blockchain network. The experiment uses the 1024-bit message of block headers from various blockchain networks, such as Bitcoin, BitcoinCash, Bitcoin Atom, Bitcoin V, BitcoinSV, FreiCoin, ZetaCoin, DeVault, Deutsche eMark, EmbargoCoin, Susucoin, FreeCash, and Kryptofranc. The experimental results show that the FPGA-based MSA operates 100% correctly in the mining process on many different blockchain networks.

B. EVALUATING THE IMPACT OF THE PROPOSED TECHNIQUES INSIDE THE MSA

This section presents the suitable PEA dimension for the proposed MSA and the impact of the nonce generator for blockchain mining. Throughout this section, we use the two quantities of power and throughput for evaluation. The power consumption is obtained using the Xilinx Power Estimator

TABLE 4. Comparison between the proposed dual ALU architecture and related works based on FPGA synthesis results.

FPGA Platform	Reference	Configurable	Algorithm	Frequency (MHz)	Area (Slice)	#Cycle	#Hash	Throughput (Mhps)	Area Eff. (Khps/Slice)
Virtex XCV200	Glabb et al [27]	Yes	SHA256	50	2,951	64	2	1.563	0.53
			SHA512			80	1	0.625	0.21
	Zeghid et al [28]	Yes	SHA256	53	2,530	32	1	1.656	0.65
			SHA512			64	1	0.828	0.33
	Proposed Architecture	**Yes**	**SHA256**	**101**	**3,449**	**192**	**6**	**3.156**	**0.92**
			SHA512			**240**	**3**	**1.263**	**0.37**
			SHA256d			**192**	**3**	**1.578**	**0.46**
Virtex 2 XC2VP20	Garcia et al [11]	No	SHA256	35	431	280	1	0.125	0.29
	Kim et al [45]	No	SHA256	85	1,210	355	1	0.239	0.20
	George et al [46]	No	SHA512	73	2,169	81	1	0.901	0.42
	Proposed Architecture	**Yes**	**SHA256**	**165**	**3,695**	**195**	**6**	**5.077**	**1.37**
			SHA512			**243**	**3**	**2.037**	**0.55**
			SHA256d			**195**	**3**	**2.538**	**0.69**

tool in Vivado version 2019.2. The throughput, measured in megahashes per second (Mhps), is calculated by eq. (1), where #Hash is the number of generated hashes, T_{WR_MSA} is the time to write data from the DDR memory to the MSA, T_{MSA_EXE} is the execution time of the MSA, and $S(T_{WR\_MSA}, T_{MSA\_EXE})$ is the total time of the data writing and MSA execution.

$$\text{Throughput} = \frac{\#\text{Hash}}{S(T_{WR\_MSA}, T_{MSA\_EXE})} \quad (1)$$

Note that the throughput estimation does not consider the time for reading data from the accelerator to DDR memory because the data reading process can be performed in parallel with the execution of the M-PEs, as shown in Fig. 10 (a).

1) SUITABLE PEA DIMENSION FOR THE PROPOSED MSA

The proposed MSA uses multiple M-PEs to accelerate the SHA-2 computations. Theoretically, increasing the number of M-PEs (the PEA dimension) may improve the performance of the MSA. However, increasing the PEA dimension will greatly increase the power consumption of the MSA. Meanwhile, the MSA processing rate will not increase much because of the bandwidth bottleneck between the DDR memory and the accelerator. In addition, the large PEA dimension can make the arbiters and the global H^{t+1} memory more complex, which increases the critical path. In contrast, if a PEA dimension is excessively small, the MSA will have a low performance. To find a suitable PEA dimension, this section evaluates the throughput and power of MSAs with different PEA dimensions.

In Fig. 12, we present the throughput and power of the seven MSA versions with seven PEA dimensions: 1×1, 2×2, 4×4, 8×4, 8×6, 8×8, and 8×10. Overall, the throughput and power of the MSA increase with increasing PEA dimensions. Specifically, the MSAs with 1×1 to 8×10 PEA dimensions consume 0.15 W to 7.34 W, respectively. For the SHA-224/256 computations, the MSAs with 1×1 to 8×10 PEA dimensions deliver 7.21 Mhps to 103.81 Mhps, respectively. For the SHA-384/512 computations, the performance of the MSAs with 1×1 to 8×10 PEA dimensions is 2.93 Mhps to 49.83 Mhps, respectively. On the other hand, the MSA performance for the SHA256d computation only increases when the PEA dimension increases from 1×1 to 8×8, reaching 3.91 Mhps to 250 Mhps, respectively. Because ALU1 and ALU2 of each M-PE compute 64 loops in SHA-256d mode, using 64 M-PEs (8×8 dimensions) will enable the M-PEs to execute continuously with no idle status. If the PEA dimension exceeds 64 M-PEs, some of the M-PEs will stop after executing 64 loops, leading to wasted execution time. The proof is that the throughput of SHA-256d reaches the saturation threshold of 250 Mhps with 8×10 PEA dimensions.

Based on the above analysis, the 8×8 PEA dimension is the most suitable for the MSA to maximize the SHA-256d throughput and improve the SHA-224/256/384/512 throughput while maintaining reasonable power consumption. Therefore, the 8×8 PEA dimension is selected for the proposed MSA.

2) THE IMPACT OF THE NONCE GENERATOR (NOG) FOR BLOCKCHAIN MINING

The above analysis shows that the SHA-256d throughput is superior to the SHA224/256/384/512 throughput and peaks at 250 Mhps. The main reason for the excellent SHA256d throughput is that the NOG and NOD help to reduce the bandwidth pressure between the DDR memory and the MSA. Because this evaluation does not consider the data reading from the MSA to DDR memory, we only evaluate the NOG. To clarify the impact of the NOG, this section analyzes the

TABLE 5. Comparison of hardware efficiency between the proposed MSA and FPGA-based works.

Design level		Standalone core							System on chip					
Reference		Martino et al [20]		Zhang et al [36]		**Proposed Architecture**			Tran et al [38]	Kammoun et al [47]	Khas et al [48]	**Proposed Architecture**		
Configurable		No		No		**Yes**			No	No	No	**Yes**		
FPGA Platform		KCU116		KCU105		**Alveo U280**			ZCU102	XC7Z020	XC7A35T	**Alveo U280**		
Technology (nm)		16		20		**16**			16	28	28	**16**		
Algorithm		SHA256	SHA512	SHA256	SHA512	**SHA256**	**SHA512**	**SHA256d**	SHA256	SHA256	SHA512	**SHA256**	**SHA512**	**SHA256d**
Area	LUT	5,385	10,424	23,270	75,011	**210,880**			205,007	1,036	6,115	**285,754**		
	FF	4,314	8,540	45,312	115,200	**366,208**			346,391	1,322	5,641	**522,944**		
	Slice	-	-	4,870	16,098	**44,769**			33,852	339	1,529	**55,369**		
Frequency (MHz)		328	320	200	125	**650**			320	222	90	**250**		
Power (W)		0.43	0.77	2.56	-	**13.63**			6.14	0.05	0.26	**6.57**		
Throughput (Mhps)		5.05	3.95	200	125	**1,280**	**513.58**	**650**	12.8	0.19	0.80	**99.7**	**47.5**	**250**
Area Efficiency (Mhps/1kLUT)		0.94	0.38	8.59	1.67	**6.07**	**2.44**	**3.08**	0.06	0.18	0.13	**0.35**	**0.17**	**0.87**
Energy Efficiency (Mhps/W)		11.74	5.13	78.13	-	**93.91**	**37.67**	**47.69**	2.08	3.8	3.08	**15.18**	**7.23**	**38.05**

throughput and power of the MSA with and without the proposed NOG.

To demonstrate that the NOG can maximize the SHA256d throughput, we evaluate two versions of the MSA architecture: the MSA without the NOG and the MSA with the NOG. In this experiment, the two architectures try 2^{32} nonce values by performing the SHA256d computation of 2^{32} messages. In the MSA without the NOG, the 2^{32} messages are transmitted from the DDR memory 2^{32} times. However, the MSA with the NOG receives only one message from the DDR memory, and the NOG will update the 2^{32} 32-bit nonce values to generate 2^{32} messages.

Table 3 describes the performance comparison between the two MSA architectures when performing the SHA256d computation for 2^{32} messages. Specifically, the MSA with the NOG is **4.17 times** (250 vs. 62.5) better than the MSA without the NOG in terms of throughput. Additionally, the MSA with the NOG is not much better than the MSA without the NOG in terms of power consumption. Therefore, the MSA with the NOG is approximately **4.17 times** (38.05 vs. 9.57) higher than the MSA without the NOG in terms of energy efficiency.

Using the NOG, the MSA can achieve the maximum performance for the SHA256d computation. Therefore, the NOG is integrated into the proposed MSA to achieve 250 Mhps for blockchain mining.

C. PERFORMANCE EVALUATION

1) EVALUATING THE PROPOSED DUAL ALU ARCHITECTURE

In the proposed MSA, the dual ALUs are the most important component to accelerate the computational performance of SHA-2 functions. On the other hand, most previous SHA-2 works only focus on optimizing the SHA-2 ALU. Therefore, this section presents a performance evaluation between the proposed dual ALU architecture and related ALU architectures.

For a fair comparison with the existing SHA-2 ALU architectures such as [11], [27], [28], [45], [46], we have synthesized the proposed dual ALU circuits on two Xilinx Virtex FPGA boards, including Virtex XCV200-2 FF324 and Virtex 2 XC2VP20-7 FG676. Note that the proposed dual ALU architecture is discarded the final adders, used for hashing completion after message expansion and compression processes, to be similar to the ALU architectures in [27], [28]. In contrast, the proposed dual ALU architecture is kept intact for comparison with the related ALU architectures in [11], [45], [46]. Comparative factors include throughput, area efficiency, and flexibility. During our experiment, we used an Xilinx ISE version 10.1.

The throughput, measured in megahashes per second (Mhps), is calculated by eq. (2), where #Hash is the number of generated hashes per working session, frequency is the maximum operating frequency obtained from ISE synthesis results, and #Cycle is the number of clock cycles to generate #Hash.

$$\text{Throughput} = \frac{\#\text{Hash} \times \text{Frequency}}{\#\text{Cycle}} \tag{2}$$

Then, hardware efficiency is calculated by eq. (3).

$$\text{Area Efficiency} = \frac{\text{Throughput}}{\text{Area}} \tag{3}$$

Table 4 shows the throughput and area efficiency comparisons between the proposed dual ALU architecture and previous ALU architectures on the Virtex XCV200 and Virtex 2 XC2VP20 boards.

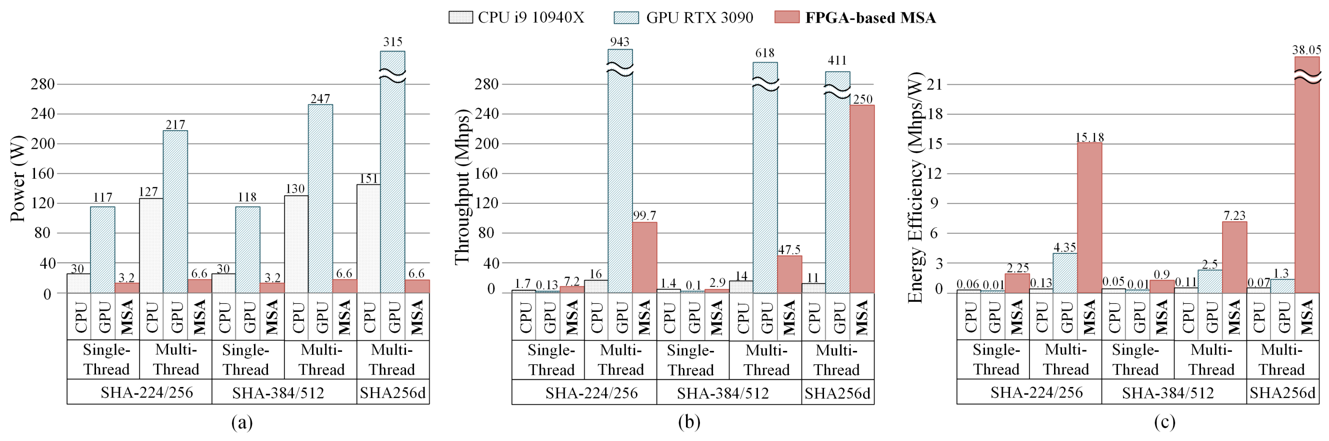


FIGURE 13. Comparison of the proposed MSA with a state-of-the-art CPU and GPU: (a) Power, (b) throughput, and (c) energy efficiency comparisons.

On the Virtex XCV200 board, the proposed dual ALU architecture occupies 3,449 slices, operates at a maximum frequency of 101 MHz, and reaches 3.156, 1.263, and 1.578 Mhps for SHA256, SHA512, and SHA256d computations, respectively. In SHA256 mode, the proposed dual ALU architecture is **2 times** (3.156 vs. 1.563) and **1.9 times** (3.156 vs. 1.656) higher than [27] and [28] in throughput, respectively, and **1.7 times** (0.92 vs. 0.53) and **1.4 times** (0.92 vs 0.65) better than [27] and [28] in area efficiency, respectively. In SHA512 mode, the proposed dual ALU architecture is **2 times** (1.263 vs. 0.625) and **1.5 times** (1.263 vs. 0.828) greater than [27] and [28] in throughput, respectively, and **1.8 times** (0.37 vs. 0.21) and **1.1 times** (0.37 vs 0.33) better than [27] and [28] in area efficiency, respectively.

On the Virtex 2 XC2VP20 board, the proposed dual ALU architecture utilizes 3,695 slices, operates at a maximum frequency of 165 MHz, and delivers 5.077, 2.037, and 2.538 Mhps for SHA256, SHA512, and SHA256d computations, respectively. In SHA256 mode, the proposed dual ALU architecture is **40.6 times** (5.077 vs. 0.125) and **21.2 times** (5.077 vs. 0.239) higher than [11] and [45] in throughput, respectively, and **4.7 times** (1.37 vs. 0.29) and **6.9 times** (1.37 vs 0.20) better than [11] and [45] in area efficiency, respectively. In SHA512 mode, the proposed dual ALU architecture is **2.3 times** (2.037 vs. 0.901) greater than [46] in throughput, and **1.3 times** (0.55 vs. 0.42) higher than [46] in area efficiency.

In addition to comparing throughput and area efficiency, we evaluate the flexibility between the proposed dual ALU architecture and the previous ALU architectures in [11], [27], [28], [45], [46]. Particularly, the proposed dual ALU architecture can be configured by embedded software to change between many SHA-2 functions (modes) immediately. Although the ALU architectures in [27], [28] are configurable, those ALUs can only perform SHA-256 and SHA-512 but SHA256d. Meanwhile, the ALU architectures in [11], [45], [46] are not configurable and can only execute a single hash function. Therefore, the proposed dual ALU architecture has more flexibility than previous ALU architectures.

2) MSA VS. FPGA-BASED WORKS

This section presents a comparison of the throughput, area, and energy efficiencies between the proposed MSA and state-of-the-art designs based on the results of the FPGA evaluation, as shown in Table 5. We evaluate them at two levels: the standalone core and SoC.

At the standalone core level, only the dual ALUs (ALU1 and ALU2) and PE controller of the proposed MSA are synthesized and evaluated on the Xilinx Alveo U280 FGPA. The MSA needs 210,880 LUTs and 366,208 flip-flops (FFs), operates at a maximum frequency of 650 MHz, and consumes 13.63 W. The throughput of the proposed MSA reaches 1,280 Mhps, 580 Mhps, and 650 Mhps for the SHA256, SHA512, and SHA256d computations, respectively. Note that the throughput of the accelerators in [20] and [36] is calculated based only on the number of generated hashes over the execution time inside the computational unit (ALU) without considering the transmission time between the DDR memory and the accelerator. For a fair comparison, the throughput of the proposed MSA is also calculated similarly to that of the designs in [20] and [36]. In SHA256 mode, the proposed MSA is **6.46 times** (6.07 vs. 0.94) and **8 times** (93.91 vs. 11.74) better than [20] in area and power efficiencies, respectively. In SHA512 mode, the proposed MSA is **6.42 times** (2.44 vs. 0.38) and **1.46 times** (2.44 vs. 1.67) greater than [20] and [36] in area efficiency, respectively, and is **7.34 times** (37.67 vs. 5.13) better [20] in energy efficiency.

At the SoC level, the full circuit of the proposed MSA is implemented and evaluated on the Xilinx Alveo U280 FGPA. The proposed MSA occupies 285,754 LUTs and 522,944 FFs, operates at 250 MHz, and consumes 6.57 W. The throughput of the proposed MSA reaches 99.7 Mhps, 47.5 Mhps, and 250 Mhps for SHA256, SHA512, and SHA256d computations, respectively. Note that the throughput of the proposed MSA is calculated by eq. (1). Compared with the state-of-the-art works in SHA256 mode, the proposed MSA is **5.83 times** (0.35 vs. 0.06) and **1.94 times** (0.35 vs. 0.18) higher than [38] and [47] in area efficiency, respectively, and is **7.3 times** (15.18 vs. 2.08) and **3.99 times** (15.18 vs. 3.8) greater than [38] and [47] in energy efficiency, respectively. In SHA512 mode, the proposed MSA is **1.31 times** (0.17 vs. 0.13) and **1.63 times** (7.23 vs. 3.08) better than [48] in area and energy efficiencies, respectively.

In addition to comparing area and energy efficiencies, we evaluate the flexibility between the proposed MSA and the accelerators in [20], [36], [38], [47], and [48]. Specifically, the proposed MSA can be configured by embedded software to switch between many SHA-2 functions (modes) instantly. Additionally, the accelerators in [20], [36], [38], [47], and [48] are fixed hardware for performing only a single hash function. Therefore, the proposed MSA has higher flexibility than state-of-art FPGA-based works.

3) MSA VS. A STATE-OF-THE-ART CPU AND GPU

Since state-of-the-art FPGA-based accelerators have poor performance and low flexibility, the proposed MSA needs to be evaluated with other high-performance and flexible hardware platforms that can execute a large number of hash computations with various SHA-2 modes. Therefore, this section evaluates the proposed MSA in comparison with high-performance hardware platforms, such as CPUs and GPUs. Concretely, this section compares the power, throughput, and energy efficiency of the proposed MSA with the most powerful CPU and GPU when executing SHA-224/256, SHA-384/512, and SHA-256d in two scenarios: single-thread (or one activated M-PE of the proposed MSA) and multithread (or the full sixty-four activated M-PEs of the proposed MSA).

Fig. 13 (a)-(c) compares the power, throughput, and energy efficiency of three hardware platforms: the proposed MSA on the Xilinx Alveo U280 FGPA (FPGA-based MSA), the Intel i9 10940X CPU, and the RTX 3090 GPU. It should be noted that each hardware platform consumes a different amount of static power even without running SHA-2 programs. Specifically, the static power of the CPU, GPU, and FPGA-based MSA is 13, 31, and 10.9 W. However, the power for SHA-2 execution is known as dynamic power. For a fair comparison, the power consumption considered in this section is only dynamic power. Fig. 13 (a) shows that the GPU consumes the most power regardless of the experimental scenario. Additionally, the CPU consumes approximately half as much power as the GPU. Regarding the most energy-efficient platform, the FPGA-based MSA power is at least **9.4 times** (30 vs. 3.2) and **36.6 times** (117 vs. 3.2) less than the CPU and GPU power, respectively. In the performance comparison, Fig. 13 (b) presents the throughput of SHA224/256, SHA384/512, and SHA256d performed on the CPU, GPU, and FPGA-based MSA. When performing the SHA-2 computations in a single thread, the CPU and GPU platforms exhibit poor performance, less than 1.7 Mhps. In the single thread experiment, the FPGA-based MSA delivers at least 2.9 Mhps, which is significantly better than the CPU and GPU. For multithread execution, the GPU outperforms the CPU and MSA since the GPU has a large number of cores and threads. Specifically, the GPU performance peaks at 943 Mhps for SHA-224/256, which is 58.9 times (943 vs. 16) and 9.5 times (943 vs. 99.7) higher than that of the CPU and FPGA-based MSA, respectively. Note that the FPGA-based MSA is less than 1.6 times (250 vs. 411) less than the GPU in SHA-256d throughput thanks to the proposed NOG and NOD mechanisms. Despite being inferior in performance to the GPU, the FPGA-based MSA's energy efficiency is still better than that of the GPU since the FPGA-based MSA power is very low compared to the GPU power. As shown in Fig. 13 (c), the energy efficiency of the FPGA-based MSA reaches 38.05 Mhps/W for the SHA-256d computation, which is **543.6 times** (38.05 vs. 0.07) and **29 times** (38.05 vs. 1.3) higher than that of the CPU and GPU, respectively.

V. CONCLUSION

The SHA-2 cryptographic functions play an important role in many applications, from ensuring data security and integrity in network security to maintaining the distribution of blockchain networks. Developing hardware architectures with high performance and flexibility for a wide range of SHA-2 applications has thus become an attractive research trend. Unfortunately, it is difficult to achieve state-of-the-art SHA-2 architectures with high performance and flexibility with high hardware efficiency. In this study, we solve the above problems by developing a multimode SHA-2 accelerator (MSA) at the system-on-chip level. Specifically, the proposed MSA applies several optimization techniques, including multiple multimode processing elements, dual pipeline ALUs, nonce generators, and nonce detectors, to achieve this purpose. The proposed MSA is implemented and verified on the Xilinx Alveo U280 FPGA. With FPGA Xilinx 16 nm FinFET technology, the proposed MSA reaches a maximum processing rate of 250 Mhps in SHA-256d mode. The experimental results on the FPGA show that the MSA not only achieves high performance and hardware efficiency but also has superior flexibility compared to previous works. Comparing general hardware platforms such as CPUs and GPUs, the proposed MSA is significantly better than the Intel i9-10940X CPU and RTX 3090 GPU in energy efficiency.

Overall, our accelerator supports only the hash functions of the SHA-2 family. However, data security applications and blockchain mining require the use of various cryptographic hash algorithms, such as SHA-3, BLAKE, and MD-5. Therefore, developing high-performance and power-efficient hardware that can support more hash functions will be our research direction in the near future.

APPENDIX

The synthesized results on the FPGA, the C code for the CPU, and the Cuda code for the GPU can be found at https://github.com/archlab-naist/MSA_Luan/.

REFERENCES

[1] Q. H. Dang, "Secure hash standard," Federal Inf. Process. Standards Publication, 2015, pp. 180–184.
[2] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Proc. 25th Annu. Int. Conf. Adv. Cryptol.* Berlin, Heidelberg: Springer-Verlag, 2005, pp. 17–36.
[3] M. J. Dworkin, "Sha-3 standard: Permutation-based hash and extendable-output functions," 2015.
[4] H. E. Michail, G. S. Athanasiou, V. Kelefouras, G. Theodoridis, and C. E. Goutis, "On the exploitation of a high-throughput SHA-256 FPGA design for HMAC," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, no. 1, pp. 1–28, Mar. 2012.
[5] M. Juliato and C. Gebotys, "A quantitative analysis of a novel SEU-resistant SHA-2 and HMAC architecture for space missions security," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, no. 3, pp. 1536–1554, Jul. 2013.
[6] H. Choi and S. C. Seo, "Optimization of PBKDF2 using HMAC-SHA2 and HMAC-LSH families in CPU environment," *IEEE Access*, vol. 9, pp. 40165–40177, 2021.
[7] W. Shan, W. Dai, C. Zhang, H. Cai, P. Liu, J. Yang, and L. Shi, "TG-SPP: A one-transmission-gate short-path padding for wide-voltage-range resilient circuits in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 55, no. 5, pp. 1422–1436, May 2020.
[8] P. Gallagher, "Digital signature standard (DSS)," Federal Inf. Process. Standards Publications, 2013, pp. 186–193.
[9] A. Coughlin, G. Cusack, J. Wampler, E. Keller, and E. Wustrow, "Breaking the trust dependence on third party processes for reconfigurable secure hardware," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, New York, NY, USA, 2019, pp. 282–291.
[10] M. Feldhofer and C. Rechberger, "A case against currently used hash functions in RFID protocols," in *Proc. Int. Conf. Move Meaningful Internet Syst., AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET.* Berlin, Heidelberg: Springer-Verlag, 2006, pp. 372–381.
[11] R. García, I. Algredo-Badillo, M. Morales-Sandoval, C. Feregrino-Uribe, and R. Cumplido, "A compact FPGA-based processor for the secure hash algorithm SHA-256," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 194–202, Jan. 2014.
[12] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.
[13] H. Michail, A. Kakarountas, A. Milidonis, and C. Goutis, "A top-down design methodology for ultrahigh-performance hashing cores," *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 4, pp. 255–268, Oct. 2009.
[14] F. Wang, Y. Chen, R. Wang, A. O. Francis, B. Emmanuel, W. Zheng, and J. Chen, "An experimental investigation into the hash functions used in blockchains," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1404–1424, Nov. 2020.
[15] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117134–117151, 2019.
[16] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.
[17] T. H. Tran, H. L. Pham, T. D. Phan, and Y. Nakashima, "BCA: A 530-mW multicore blockchain accelerator for power-constrained devices in securing decentralized networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 10, pp. 1–14, Oct. 2021.
[18] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," in *Proc. 25th IET Irish Signals Syst. Conf. China-Ireland Int. Conf. Inf. Communities Technol. (ISSC/CIICT)*, 2014, pp. 280–285.
[19] S. Valfells and J. H. Egilsson, "Minting money with megawatts [point of view]," *Proc. IEEE*, vol. 104, no. 9, pp. 1674–1678, Sep. 2016.
[20] R. Martino and A. Cilardo, "A flexible framework for exploring, evaluating, and comparing SHA-2 designs," *IEEE Access*, vol. 7, pp. 72443–72456, 2019.
[21] R. Martino and A. Cilardo, "Designing a SHA-256 processor for blockchain-based IoT applications," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100254.
[22] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Cost-efficient SHA hardware accelerators," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 8, pp. 999–1008, Aug. 2008.
[23] I. Algredo-Badillo, C. Feregrino-Uribe, R. Cumplido, and M. Morales-Sandoval, "FPGA-based implementation alternatives for the inner loop of the secure hash algorithm SHA-256," *Microprocess. Microsyst.*, vol. 37, no. 6, pp. 750–757, Aug. 2013.
[24] R. P. McEvoy, F. M. Crowe, C. C. Murphy, and W. P. Marnane, "Optimisation of the SHA-2 family of hash functions on FPGAs," in *Proc. IEEE Comput. Soc. Annu. Symp. Emerg. VLSI Technol. Arch. (ISVLSI)*, 2006, pp. 317–322.
[25] H. E. Michail, G. S. Athanasiou, G. Theodoridis, and C. E. Goutis, "On the development of high-throughput and area-efficient multi-mode cryptographic hash designs in FPGAs," *Integration*, vol. 47, no. 4, pp. 387–407, Sep. 2014.
[26] R. Ramanarayanan, S. Mathew, F. Sheikh, S. Srinivasan, A. Agarwal, S. Hsu, H. Kaul, M. Anders, V. Erraguntla, and R. Krishnamurthy, "18 Gbps, 50 mW reconfigurable multi-mode SHA hashing accelerator in 45 nm CMOS," in *Proc. ESSCIRC*, Sep. 2010, pp. 210–213.
[27] R. Glabb, L. Imbert, G. Jullien, A. Tisserand, and N. Veyrat-Charvillon, "Multi-mode operator for SHA-2 hash functions," *J. Syst. Archit.*, vol. 53, nos. 2–3, pp. 127–138, Feb. 2007.
[28] A. Hodjat, P. Schaumont, and I. Verbauwhede, "Architectural design features of a programmable high throughput AES coprocessor," in *Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC)*, 2004, pp. 498–502.
[29] N. Sklavos and O. Koufopavlou, "Implementation of the SHA-2 hash family standard using FPGAs," *J. Supercomput.*, vol. 31, no. 3, pp. 227–248, Mar. 2005.
[30] W. Sun, H. Guo, H. He, and Z. Dai, "Design and optimized implementation of the SHA-2(256, 384, 512) hash algorithms," in *Proc. 7th Int. Conf. ASIC*, Oct. 2007, pp. 858–861.
[31] L. V. T. Duong, N. T. T. Thuy, and L. D. Khai, "A fast approach for bitcoin blockchain cryptocurrency mining system," *Integration*, vol. 74, pp. 107–114, Sep. 2020.
[32] V. Suresh, S. Satpathy, and S. Mathew, "Bitcoin mining hardware accelerator with optimized message digest and message scheduler datapath," U.S. Patent 15 274 200, Mar. 29, 2018.
[33] V. B. Suresh, S. K. Satpathy, and S. K. Mathew, "Optimized SHA-256 datapath for energy-efficient high-performance Bitcoin mining," U.S. Patent 10 142 098, Nov. 27, 2018.
[34] V. B. Suresh, S. K. Satpathy, and S. K. Mathew, "Energy-efficient bitcoin mining hardware accelerators," U.S. Patent 10 313 108, Jun. 4, 2019.
[35] H. L. Pham, T. H. Tran, T. D. Phan, V. T. Duong Le, D. K. Lam, and Y. Nakashima, "Double SHA-256 hardware architecture with compact message expander for bitcoin mining," *IEEE Access*, vol. 8, pp. 139634–139646, 2020.
[36] Y. Zhang, Z. He, M. Wan, M. Zhan, M. Zhang, K. Peng, M. Song, and H. Gu, "A new message expansion structure for full pipeline SHA-2," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 4, pp. 1553–1566, Apr. 2021.
[37] V. D. Phan, H. L. Pham, T. H. Tran, and Y. Nakashima, "High performance multicore SHA-256 accelerator using fully parallel computation and local memory," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, Apr. 2021, pp. 1–3.
[38] T. H. Tran, H. L. Pham, and Y. Nakashima, "A high-performance multimem SHA-256 accelerator for society 5.0," *IEEE Access*, vol. 9, pp. 39182–39192, 2021.
[39] R. Martino and A. Cilardo, "SHA-2 acceleration meeting the needs of emerging applications: A comparative survey," *IEEE Access*, vol. 8, pp. 28415–28436, 2020.
[40] I. Ahmad and A. Shoba Das, "Hardware implementation analysis of SHA-256 and SHA-512 algorithms on FPGAs," *Comput. Electr. Eng.*, vol. 31, no. 6, pp. 345–360, Sep. 2005.
[41] M. Rahouti, K. Xiong, and N. Ghani, "Bitcoin concepts, threats, and machine-learning security solutions," *IEEE Access*, vol. 6, pp. 67189–67205, 2018.
[42] J. Taskinsoy, "Bitcoin and Turkey: A good match or a perfect storm," *SSRN Electron. J.*, Oct. 2019, doi: 10.2139/ssrn.3477849.
[43] Y. Joo and N. McKeown, "Doubling memory bandwidth for network buffers," in *Proc. IEEE INFOCOM*, vol. 2, 1998, pp. 808–815.
[44] T. Kim, W. Jao, and S. Tjiang, "Circuit optimization using carry-save-adder cells," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 10, pp. 974–984, Oct. 1998.
[45] M. Kim, D. G. Lee, and J. Ryou, "Compact and unified hardware architecture for SHA-1 and SHA-256 of trusted mobile computing," *Pers. Ubiquitous Comput.*, vol. 17, no. 5, pp. 921–932, Jun. 2013.
[46] G. S. Athanasiou, C. E. Goutis, G. Theodoridis, and H. E. Michail, "Optimising the SHA-512 cryptographic hash function on FPGAs," *IET Comput. Digit. Techn.*, vol. 8, no. 2, pp. 70–82, 2014.

[47] M. Kammoun, M. Elleuchi, M. Abid, and A. M. Obeid, ''HW/SW architecture exploration for an efficient implementation of the secure hash algorithm SHA-256,'' *J. Commun. Softw. Syst.*, vol. 17, no. 2, pp. 87–96, 2021.

[48] A. Al Khas and I. Cicek, ''SHA-512 based wireless authentication scheme for smart battery management systems,'' in *Proc. 8th Int. Conf. Renew. Energy Res. Appl. (ICRERA)*, Nov. 2019, pp. 968–972.

HOAI LUAN PHAM (Graduate Student Member, IEEE) received the bachelor's degree in computer engineering from Vietnam National University Ho Chi Minh City (VNUHCM)—University of Information Technology, Vietnam, in 2018, and the M.S. degree in information science from the Nara Institute of Science and Technology (NAIST), Japan, in 2020, where he is currently pursuing the Ph.D. degree. His research interests include blockchain technology and cryptography.

THI HONG TRAN (Member, IEEE) received the bachelor's degree in physics and the master's degree in microelectronics from Vietnam National University Ho Chi Minh City (VNU-HCM)—University of Science, Vietnam, in 2008 and 2012, respectively, and the Ph.D. degree in information science from the Kyushu Institute of Technology, Japan, in 2014. From January 2015 to September 2021, she has been with the Nara Institute of Science and Technology (NAIST), Japan, as a full-time Assistant Professor. Since October 2021, she has been with Osaka City University, Japan, as a full-time Lecturer, and NAIST as a Visiting Associate Professor. Her research interests include digital hardware circuit design, algorithms related to wireless communication, communication security, blockchain technology, SHA-2, SHA-3, and cryptography. She is a Regular Member of IEEE, IEICE, and REV-JEC.

VU TRUNG DUONG LE (Graduate Student Member, IEEE) received the B.E. degree in IC and hardware design from Vietnam National University Ho Chi Minh City (VNUHCM)—University of Information Technology, in 2020. He is currently pursuing the M.S. degree with the Nara Institute of Science and Technology (NAIST), Japan. His research interests include blockchain technology and cryptography.

YASUHIKO NAKASHIMA (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer engineering from Kyoto University, in 1986, 1988, and 1998, respectively. He was a Computer Architect with the Computer and System Architecture Department, Fujitsu Ltd., from 1988 to 1999. From 1999 to 2005, he was an Associate Professor with the Graduate School of Economics, Kyoto University. Since 2006, he has been a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include computer architecture, emulation, circuit design, and accelerators. He is a fellow of IEICE, a Senior Member of IPSJ, and a member of the IEEE CS and ACM.

Received November 11, 2021, accepted November 24, 2021, date of publication November 30, 2021,
date of current version December 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3131558

MRSA: A High-Efficiency Multi ROMix Scrypt Accelerator for Cryptocurrency Mining and Data Security

VU TRUNG DUONG LE[1], (Graduate Student Member, IEEE),
THI HONG TRAN[2], (Member, IEEE),
HOAI LUAN PHAM[1], (Graduate Student Member, IEEE),
DUC KHAI LAM[3], AND YASUHIKO NAKASHIMA[1], (Senior Member, IEEE)
[1]Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma, Nara 630-0192, Japan
[2]Graduate School of Engineering, Osaka City University, Osaka 558-8585, Japan
[3]Computer Engineering Department, University of Information Technology, Vietnam National University, Ho Chi Minh City 700000, Vietnam

Corresponding author: Thi Hong Tran (hong@osaka-cu.ac.jp)

This work was supported by the Japan Science and Technology Agency (JST) under a Strategic Basic Research Programs PRESTO (Precursory Research for Embryonic Science and Technology), Grant number JPMJPR20M6.

ABSTRACT The development of low-energy, high-performance hardware for cryptocurrency mining is gaining widespread attention. The mining process for proof-of-work (PoW) in conventional cryptocurrencies' blockchains is increasingly being replaced by application-specific integrated circuits (ASICs). This leads to many security threats for the blockchain network because it decreases security and increases power consumption for mining. Therefore, Scrypt, the most representative ASIC-resistant algorithm, was developed to solve this problem. However, there are still some problems and challenges with the current Scrypt hardware. This article presents a new hardware architecture for the Scrypt algorithm intended for a PoW-based cryptocurrency mining system. The proposed Multi ROMix Scrypt Accelerator (MRSA) hardware architecture applies several optimization techniques: configuration, local-memory computing with high-performance pipelined Multi ROMix and rescheduling resources to significantly increase processing speed, flexibility, and energy efficiency. For evaluation, the MRSA is implemented on field-programmable gate arrays (FPGAs) to examine its actual performance, consumption, and correctness. Evaluation results on a Xilinx system-on-chip (SoC) with the ALVEO U280 Data Center Accelerator Card FPGA show that the MRSA is much more power-efficient than some of the most powerful commercial CPUs, GPUs, and other FPGA implementations. On the ALVEO U280, the MRSA achieves a maximum hash rate of 296.76 kHash/s, a throughput of 304.9 Mbps when reaching a maximum frequency of 259.94 MHz, and a power consumption of 18.12 W. The energy efficiency of the MRSA on the ALVEO U280 SoC is 52.83 and 867.88 times higher than those on an RTX 3090 GPU and an i9-10940X CPU, respectively.

INDEX TERMS Blockchain, Scrypt, accelerator, FPGA, SoC, ASIC, cryptocurrency, ASIC-resistant, cryptography hash function, proof-of-work, Litecoin.

I. INTRODUCTION

Recently, cryptocurrency has been a topic of interest. A cryptocurrency is a monetary network that uses blockchain technology as a consensus mechanism among users [1], [2]. In a blockchain network, transactions are grouped into lists contained within blocks. The blocks are linked together through the hash of the previous block, thus forming a blockchain. The blockchain is synchronized among the nodes in the network, ensuring that no data in the blockchain can be changed. To ensure authenticity, transactions require digital signatures from users [3], [4]. In addition, cryptocurrencies have mechanisms to solve other security problems, such as the possible occurrence of double spending when multiple transactions are performed simultaneously [5]–[8] or a fork occurring when multiple longest blockchains

The associate editor coordinating the review of this manuscript and approving it for publication was Yue Zhang.

exist [9], [10]. The consensus mechanism is one of the most important tools that a cryptocurrency uses to ensure consistency and integrity. There are many types of consensus mechanisms, such as proof-of-work (PoW), proof-of-stake (PoS), proof-of-authority (PoA), and several other types presented in [11], [12], and [13]. Among them, PoW is the most popular and is used by the largest cryptocurrency, Bitcoin [14].

In PoW, the miners obtain input data from the last block combined with finding a valid random nonce number such that the output hash value is less than the target value specified by the system. The new block is accepted and saved to the system permanently, and all transactions inside it are executed when the nonce is valid. However, finding a new block consumes a significant amount of computational resources, which is one of the biggest problems with PoW-based blockchains. It has been reported that the total energy consumption of the Bitcoin network in 2020 reached 109.07 TWh, which is approximately equal to the total energy consumption associated with electricity use in the Netherlands. Therefore, research and development on high-performance and low-power hardware for cryptocurrency mining systems have become a research trend in recent years [15], [16].

Many studies have presented hardware architectures to improve computational efficiency and reduce power consumption for Bitcoin mining, which uses double SHA-256 encoding. The authors of [17] introduced a high-performance multimem SHA-256 accelerator to greatly increase the speed of the hardware and reported its realization and testing on a ZCU102 field-programmable gate array (FPGA). With the proposal of a compact message expander hardware architecture for the double SHA-256 core in [18], the authors reduced the demand for hardware computing resources without affecting the processing speed. In addition, the authors of [19] proposed a two-level fully pipelined SHA-256 core with a hash rate equal to the operating frequency. By eliminating the finite state machine, shortening the critical path, and balancing the pipeline stages, their design achieves very high performance and low energy consumption.

PoW systems using double SHA-256, with immutability, simple computational components, and low memory requirements, offer enormous advantages when implemented on an application-specific integrated circuit (ASIC) hardware platform. Double SHA-256 miners on ASIC platforms achieve mining speeds far superior to those on other platforms such as FPGAs, GPUs, and CPUs. However, ASIC miners consume energy, and the market price is quite high, leading to the hardware power in a network being concentrated only in ASIC mining farms. Such centralization seriously threatens the safety of the network, increases mining energy consumption, and goes against the original purpose of PoW [15]. Hence, ASIC-resistant algorithms were created to solve these problems. They have several characteristic properties: they are highly serial, memory-intensive, and parameterizable. The highly serial and memory-intensive nature of these algorithms means that they require a high number of loops, complex dependencies among loops, and considerable memory, thereby decreasing performance and increasing manufacturing costs for ASICs. Meanwhile, parameterizability allows the parameters of such an algorithm to be modified as needed to make current ASICs obsolete and unusable. ASIC-resistant algorithms eliminate the advantages of ASICs because they require hardware resources with high flexibility, significantly reducing computational performance and leading to high risk when using ASIC miners [20], [21]. On the other hand, FPGA-based miners are flexible, energy-efficient, and resource-rich computing tools with a reasonable cost. Therefore, we believe that FPGAs are truly the most suitable and efficient hardware platforms for ASIC-resistant cryptocurrencies. Scrypt is one of the most representative ASIC-resistant algorithms used in today's PoW-based cryptocurrencies, of which the most popular are Litecoin [22], Dogecoin [23], Fastcoin [24], and Megacoin [25], among many others [26]. Several real-world studies and hardware improvements to the Scrypt mining system have been reported. The authors of [27] built a hardware implementation for an Scrypt miner with a double ROMix core pipeline technique and reused resources to increase computation speed and reduce hardware cost. However, the reuse of hardware has not been completely optimized, a detailed review of its implications for power consumption is lacking, and this approach has not been implemented and verified in practice on a real FPGA system-on-chip (SoC).

In this paper, we propose a high-performance hardware architecture for Scrypt by assessing computation time, hardware cost, and power consumption. Furthermore, this is the first hardware implementation for Scrypt miners on a Xilinx SoC. This hardware architecture is called the Multi ROMix Scrypt Accelerator (MRSA). With its proposed configurability feature, the MRSA can also operate under many parameters and modes to adapt when the mining system parameters change or be applied in many other Scrypt applications. The MRSA uses multiple ROMix processing elements (ROMix PEs) in a cyclic pipeline to increase processing efficiency and minimize the hardware idle time. With near-memory computing, these pipelined ROMix PEs can access the memory separately and in parallel. This significantly reduces the time needed for data transfer between the accelerator and the external memory. Finally, we analyze the algorithm and apply rescheduling and rearranging techniques to reduce the total hardware computation power and resources.

The remainder of this paper is presented as follows. Section II provides the background for this study. Section III presents the details of the proposed research contributions. A comparative evaluation of the proposed design implemented on a Xilinx FPGA SoC with other hardware platforms and studies is presented in Section IV. Finally, Section V concludes the paper.

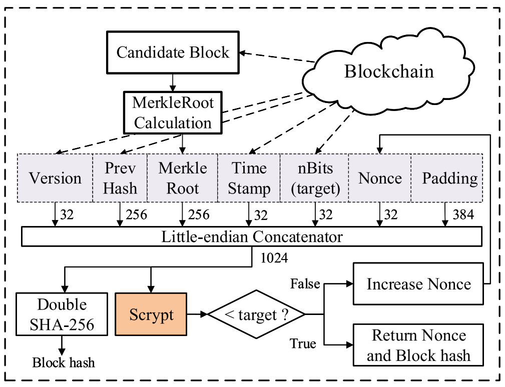


FIGURE 1. Scrypt proof-of-work (PoW) mining system architecture.

II. BACKGROUND

A. PROOF-OF-WORK

PoW is the most popular and secure consensus mechanism used in the oldest and most stable cryptocurrencies, such as Bitcoin, Ethereum, and Litecoin. It trades off hardware power to ensure the security of the blockchain network. Fig. 1 shows a diagram of a PoW mining system. In this system, miners choose pending transactions and gather them into a candidate block. Then, the miners use their computational power to find the proof necessary to add the candidate block to the blockchain network. This proof is a random nonce value such that the mining result is lower than the required target. In PoW-based cryptocurrencies, a block consists of two main fields: the block header and transactions. The transactions field is the list of executed transactions saved in the block. The remaining field is the block header, which comprises six fields, as described in Table 1, serving as the input for the mining process.

The main processing component in the mining system is the mining algorithm. The mining algorithm considered in this study is Scrypt. It returns a 256-bit hash string calculated from the block header input. Increasing the block header's nonce value allows miners to change the hash output to find a valid nonce. The comparator module compares the Scrypt hash against the target value. The new block and nonce are considered valid only if the Scrypt hash is lower than the target value. Then, they will be broadcast through the blockchain network for the other miners to verify. Subsequently, the new block is permanently added to the blockchain network. Additionally, the miner who mined that block will automatically receive a reward from the system and all fees from the transactions in the new block. However, if a valid result is not found, the miner must change the nonce and recalculate until the Scrypt result is accepted. Essentially, the current ASIC-resistant mining process is not fully effective because it is performed by general hardware platforms such as CPUs and GPUs, which generally have low performance and high energy consumption.

B. SCRYPT

Introduced by Percival and Josefsson in [28], the Scrypt algorithm is a password-based key derivation and a sequential memory-hard function created to defend against attacks from custom hardware such as ASICs. Algorithm 1 explains the details of the Scrypt algorithm. Accordingly, several parameters are used to modify the algorithm depending on its intended use. They are the block size factor (r), the CPU/memory cost parameter (N), the parallelization parameter (p), and the derived key length in bits (dklen). These parameters determine how much memory and computational power are used and how many iterations are performed in the subfunctions. In most current cryptocurrency mining systems, the parameter set (r, N, p, dklen) used in the Scrypt algorithm is (1, 1024, 1, 256) [29]. Overall, this algorithm includes two main functions, PBKDF2 and ROMix, and is divided into three steps. The first step is to process the PBKDF2 function with input parameters (message, salt, dklen) of (B_header, B_header, (1024 × r × p)). The second step is to run the ROMix function with the input parameters (Block, N, r) set to (P1, N, r). The final step is to execute the PBKDF2 function again with the input parameters (message, salt, dklen) set to (B_header, RM_out, dklen). The LittleEndian32 function converts each 32-bit segment, separately and in parallel, into the little-endian format [30]. The remainder of this subsection explains the PBKDF2 and ROMix functions in detail.

TABLE 1. Fields of the block header.

Field	Length in Bit	Description
Version	32	Block version number
Previous Block Hash	256	256-bit hash of the previous block header
Merkle Root	256	256-bit hash based on all of the transactions in the block
Time Stamp	32	Current block timestamp in seconds since 1970-01-01T00:00 UTC
nBits	32	Current target in compact format
Nonce	32	32-bit number (starts at 0)

Algorithm 1 Out = Scrypt (Blockheader)

```
Scrypt variables and parameters
for cryptocurrency mining:
    Block header (B_header) (1024 bits)
    Block size factor (r) = 1
    Parallelization parameter (p) = 1
    CPU/memory cost parameter (N) = 1024
    Length of DerivedKey in bits (dklen) = 256
Steps of the algorithm:
1: P1 = PBKDF2(B_header, B_header, 1024 × r × p)
2: P1 = LittleEndian32(P1)
3: RM_out = ROMix(P1, N, r)
4: RM_out = LittleEndian32(RM_out)
5: Scrypt_out = PBKDF2(B_header, RM_out, dklen)
6: Scrypt_out = LittleEndian32(Scrypt_out)
7: return Scrypt_out
```

1) PBKDF2

The Password-Based Key Derivation Function 2 (PBKDF2) is one of the key derivation functions used to

Algorithm 2 DK = PBKDF2(Message, Salt, Dklen)

```
1:  DK = “”
2:  for i ← 1 to (dklen/256) do
        $DK_i$ = HMAC(message, {salt, i})
        HMAC:
3:          IPAD = 36363636 ... $36_{16}$ (256 bits)
4:          OPAD = 5C5C5C5C ... $5C_{16}$ (256 bits)
5:          KHASH = SHA256(message)
6:          IXOR = {(KHASH $\oplus$ IPAD), IPAD}
7:          OXOR = {(KHASH $\oplus$ OPAD), OPAD}
8:          IHASH = SHA256({IXOR, salt, i})
9:          OHASH = SHA256({OXOR, IHASH})
10:         $DK_i$ = OHASH
11:     DK = {DK, $DK_i$}
12: end for
13: return DK
```

reduce vulnerabilities to brute force attacks with a sliding computational cost. In the Scrypt algorithm, PBKDF2 uses the Hash-based Message Authentication Code (HMAC) to input the message along with a salt value to produce a derived key [31]. HMAC is a message authentication code (MAC) that uses a cryptographic hash function and a secret cryptographic key [32], [33]. It is used to verify data integrity, to authenticate messages, and in many other cryptographic applications [34], [35]. Algorithm 2 presents more details of the PBKDF2 function, where {a, b} denotes the concatenation of a and b and $\oplus$ is the exclusive OR (Xor) operator. Accordingly, PBKDF2 includes dklen/256 loops of HMAC functions. In Scrypt with the current mining parameters (r, N, p, dklen) = (1, 1024, 1, 256), there are four HMAC loops in PBKDF2 in the first step because the input parameter dklen is 1024 $\times$ r $\times$ p. In the third step of the Scrypt algorithm, the PBKDF2 function performs HMAC only once because dklen is 256 (refer to Algorithm 1 to see the second PBKDF2 call). Finally, the output of PBKDF2 is the concatenation of the results of all HMAC loops.

HMAC uses SHA-256 as its cryptographic hash function, combined with some Xor and concatenation operations. SHA-256 is a cryptographic hash function in the Secure Hash Algorithm 2 family (SHA-2) created by the United States National Security Agency [36]. It is one of the most popular hashing algorithms and is widely used in cryptography and cybersecurity applications. Accordingly, the SHA-256 hash values create the linkages in the blockchain. This hash algorithm is used in most current cryptocurrencies and is the primitive PoW algorithm applied in Bitcoin.

SHA-256 includes three steps: padding, message expansion, and message compression. In the padding step, the message is divided into multiple 512-bit data blocks. The last block of the message is padded with a string of zeros as necessary, and the message length is expressed in bits. For each data block and the previous hash (or initial constants), message expansion and message compression are performed to compute an intermediate 256-bit hash. The hash result of the final block (final digest) is the hash value of the

Algorithm 3 RM_Out = ROMix(Block, N, r)

```
1:  for i ← 0 to (N-1) do
        Writing to memory:
2:      $Mem_i$ = Block
3:      Block = BlockMix(Block, r)
4:  end for
5:  for j ← 0 to (N-1) do
        Reading from memory:
6:      j = Block[489:480]
7:      Block = BlockMix(Block $\oplus$ $Mem_j$, r)
8:  end for
9:  return RM_out = Block
```

Algorithm 4 BM_Out = BlockMix(Block, r)

```
1:  X = Block[1023:512] (block’s 512 high bits)
2:  for i ← 0 to ((2 $\times$ r) − 1) do
3:      X = X $\oplus$ Block[511:0] (block’s 512 low bits)
4:      X = X + Salsa20/8(X)
5:      if (i == 0) then
6:          OutH = X
7:      else
8:          OutL = X
9:      end if
10: end for
11: return BM_out = OutH, OutL
```

entire message. The reader is referred to [17]–[19] for a better understanding of SHA-256. In PBKDF2, SHA-256 is the most complex process that must be considered when optimizing the hardware.

2) ROMix

ROMix is a sequential memory-hard function that Scrypt uses to interact with the (N $\times$ 128 $\times$ r)-byte memory. The details of the ROMix algorithm are presented in Algorithm 3. It consists of two main phases: the writing-to-memory phase and the reading-from-memory phase. Each phase includes N-1 loops of writing data to or reading data from memory. In current Scrypt mining systems, the number of loops in each writing and reading phase is 1024 (N = 1024, r = 1). In the writing phase, the writing values are handled by the BlockMix function and saved to memory in ascending order of address. Then, the Xor operation is performed on the stored value and the previous BlockMix calculation to decide the random order for the reading phase. More specifically, the random address to be read is determined from the 489th to 480th bits of the block data (Block[489:480]), as described in step 6 of Algorithm 3. If the parameter N is 1024 and the parameter r is 1, then the required memory for each ROMix execution is 128 kB. This is why Scrypt is a memory-intensive algorithm that is suitable for GPUs, CPUs, and FPGAs but not ASICs. Overall, the ROMix function, the second step of the Scrypt algorithm, is the most complex and hardware-demanding process. It occupies 98 percent of the total Scrypt execution time because of the many memory writing and reading loops. Therefore, we propose the Multi ROMix architecture with the main purpose of accelerating the ROMix process.

3) BlockMix

ROMix uses the BlockMix function to mix data for the writing and reading phases. Algorithm 4 shows the pseudocode for the BlockMix function. It consists of $2 \times r - 1$ processing loops. In current mining systems, the number of loops is two because the block size factor parameter (r) is one. Accordingly, each loop includes one Xor operation, one sum operation, and one Salsa20/8 process.

Salsa20/8 is the main process that the BlockMix function uses to mix the input data. It is an original cipher developed by Daniel J. Bernstein in 2005 [37]. Salsa20/8 is a hash function whose input consists of a set of sixteen 32-bit strings in little-endian format [30]. Specifically, it consists of four column rounds (CRs) and four row rounds (RRs) performed alternately. The final BlockMix result is a set of sixteen 32-bit strings, the same width as its input. Both the CRs and RRs refer to a smaller loop called a quarter round (QR). The reader is referred to [27] for more details about the Salsa20/8 algorithm.

Overall, Salsa20/8 is the most complex process in the ROMix function. It has the longest critical path when implemented in hardware, similar to the SHA-256 process in the PBKDF2 function. Hence, it is also necessary to improve the Salsa20/8 process to accelerate the entire Scrypt hardware implementation.

C. PRELIMINARY IDEA AND MOTIVATION FOR THE HIGH-PERFORMANCE MULTI ROMix SCRYPT ACCELERATOR

In general, Scrypt has several characteristics that make it suitable for implementation on FPGAs. **First**, Scrypt uses only low-computational-cost operators such as And, Xor, right shifting/rotation, and addition. There are no complex operators such as multiplication, division, or exponentiation. **Second**, the number of loops and the number of operands in each loop are both very high, mainly concentrated in the SHA-256 and Salsa20/8 calculations. **Third**, the dependency between loops in the Scrypt algorithm is very high. To be more specific, in the ROMix function, the reading order in the reading-from-memory phase is entirely dependent on the value previously written to the memory in the writing phase. On the other hand, the PBKDF2 processes in Scrypt include multiple SHA-256 calculations. These calculations also have a high dependency between loops, as analyzed in [17]. **Fourth**, the ROMix process has enormous memory requirements because of the many writing and reading loops it comprises. After the writing phase, the memory must be kept intact for the reading phase. **Fifth**, Scrypt has several parameters that the system can modify to change the number of loops and the amount of memory required for computational functions. This helps the blockchain-based PoW mechanism be more flexible to reduce the high risks posed by ASIC miners.

Scrypt is an ASIC-resistant memory-intensive algorithm with high loop dependency, as seen from its second, third, and fourth characteristics described above. This greatly reduces the advantage of ASICs over flexible hardware platforms such as CPUs, GPUs, and FPGAs. However, the performance of ASIC miners is still extremely outstanding than other hardware platforms. For example, the ASIC-based Bitmain Antminer L7 scheduled for November 2021 offers a hash rate of 9.5 GHash/s at 3425 W [38]. Despite the great advantage in performance, ASIC miners have several limitations as follows. First, ASIC miners will be at high risk of being useless and obsolete if the blockchain network changes the parameters for the mining process. This is because current commercial ASIC miners are all designed to work with fixed parameters for the best mining performance. Second, commercial ASIC miners are designed solely for blockchain mining in ultra-high performance, which throws off the balance of mining power between ASIC miners and individual user miners (e.g. CPU, GPU, and FPGA miners). Accordingly, mining farms with a concentration of many ASIC miners can easily control the entire blockchain network based on their computing power [39], [40]. Third, Scrypt was created not only for blockchain mining but also for data security applications. Meanwhile, the current commercial Scrypt ASICs are designed with fixed parameters for only blockchain mining and are unable to use for other security applications. As a result, the ASICs are low flexible and unsuitable for individual users who ensure the decentralization of the blockchain network and still have their data security demands.

Hardware platforms intended for general purposes, such as CPUs and GPUs, have considerable memory resources and numerous computation instructions. They are suitable and currently popular for implementing Scrypt in many applications. However, they tend to exhibit very poor performance because of the high loop dependency and high simple operator loop requirements, as mentioned in the first, second, and third Scrypt characteristics. Applications run on CPUs and GPUs, called software, can execute only one instruction at a time, separately and sequentially, as stipulated by their architectures and compiler mechanisms. **The greater the number of loops to be executed is, the lower the performance on CPUs and GPUs**. Furthermore, CPUs and GPUs have extremely high energy consumption because they need to operate their extremely complex computing architectures. This drawback is more evident when they need to run in multicore and multithread modes to achieve the best performance.

We believe that with their high computational and memory resources, reprogrammable hardware design, low power consumption, and high optimization for parallel pipeline processes, FPGAs are well suited for Scrypt implementation. There are several high-performance architectures that can be applied on FPGAs to reduce the memory access time, such as the systolic-array-based accelerator called EMAXVR [41], [42] used in near-memory computing. However, despite

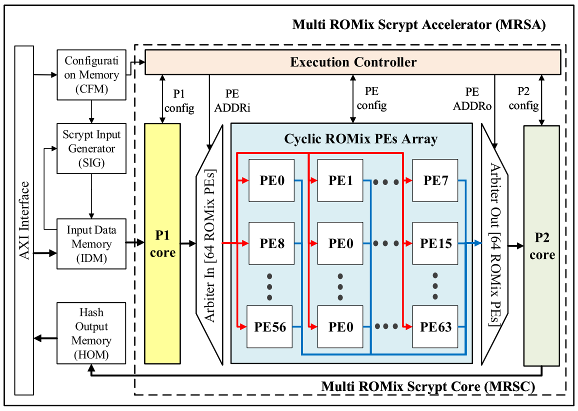


FIGURE 2. The MRSA hardware architecture.

exhibiting high performance in machine learning and image processing applications, they can achieve only poor performance when performing low-cost operator hash functions [43]. Therefore, it is necessary to develop a specific hardware architecture to optimize the performance of Scrypt on FPGAs.

Based on our understanding of Scrypt's characteristics along with the current difficulties of other hardware platforms, we propose the MRSA hardware architecture. Because Scrypt can make existing hardware useless and obsolete for performing the mining task or other security applications if the Scrypt parameters are changed, in accordance with the fifth Scrypt characteristic, we propose a configurability function for the MRSA to solve this problem. By this means, the MRSA allows its parameters to be configured to be compatible with many applications or parameterizable mining systems. In addition, Scrypt has high loop dependency and requires an enormous amount of memory for the ROMix process, in accordance with the third and fourth Scrypt characteristics. This significantly decreases the Scrypt hashing performance. Therefore, the proposed Multi ROMix architecture is applied in the MRSA to overcome this challenge. In this architecture, ROMix processes are performed in parallel by multiple ROMix PEs. With the local memory placed near the arithmetic and logic unit (ALU) in each ROMix PE, the MRSA can execute multiple Scrypt processes in parallel without conflict when using a shared ALU and without facing a bandwidth bottleneck when accessing shared memory. Scrypt also has many loops that process the same input, leading to a considerable waste of hardware computing power. Therefore, we deeply analyze the algorithm and propose a rescheduling technique for the MRSA to remove these unnecessary loops. Furthermore, large processing modules such as SHA-256 and Salsa20/8 are optimized to maximize the hardware efficiency and the hashing performance for the MRSA.

III. THE PROPOSED MULTI ROMix SCRYPT ACCELERATOR (MRSA)

A. CONFIGURABLE ARCHITECTURE

Scrypt is a parameterizable ASIC-resistant algorithm. Therefore, each Scrypt application in cryptocurrency mining or security requires specification of the input parameter set. This parameter set determines the number of loops and the width of the data passed in the subfunctions. Consequently, a configurability proposal is applied to help the MRSA adapt itself to many working modes, from cryptocurrency mining to security applications, by providing a parameter modification mechanism.

Fig. 2 shows the hardware architecture of the MRSA. It uses three memory regions managed by an Advanced eXtensible Interface (AXI). The first region is the Configuration Memory (CFM), which contains configuration data to control the MRSA. These configuration data help the CFM control the Multi ROMix Scrypt Core (MRSC) and the Scrypt Input Generator (SIG). Accordingly, the configuration data transmitted to the Execution Controller tell the MRSC when to start and which working mode and parameters are configured. The second region is the Input Data Memory (IDM), which stores the input data for the MRSA. Finally, the third region is the Hashing Output Memory (HOM). It stores the returned Scrypt hash results that the MRSA returns to the host PC.

Table 2 shows the organization of the MRSA memory in terms of byte-numbered addresses. Each register in the memory is 32 bits wide, and their addresses are separated by 4 units. The structures of the Status and Control registers are detailed in Fig. 3. These are two important registers used for the proposed configurability function. The Status register, with address 0×00, contains flags representing the status of the MRSA. The possible flags are the ready, busy, not found, and data error signals. The Control register stores the control and configuration data from the host PC. The control data include the start and reset signals. The configuration

TABLE 2. Memory organization (addresses are expressed in bytes; each location holds 32 bits).

Address	Name	Description	Region
0x00	Status	The status register	HOM
0x04	Control	The config and control	CFM
0x08	Valid Nonce	The valid nonce value	HOM
0x0C ... 0x28	Target	The target threshold value	CFM
0x2C ... 0x78	Scrypt In	The input header data	IDM
0x7C	Start Nonce	The starting nonce	CFM
0x80	Maximum Nonce	The stop nonce	CFM
0x84 ... 0xA4	Scrypt Out	The Scrypt hash output	HOM

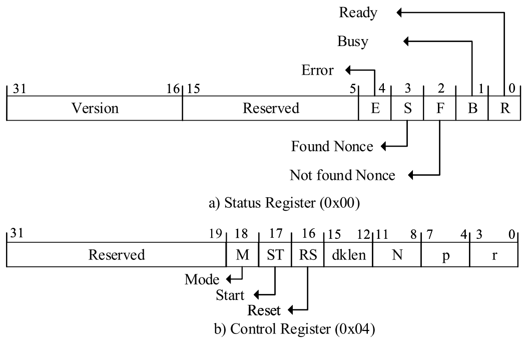


FIGURE 3. Diagrams of the status and control registers.

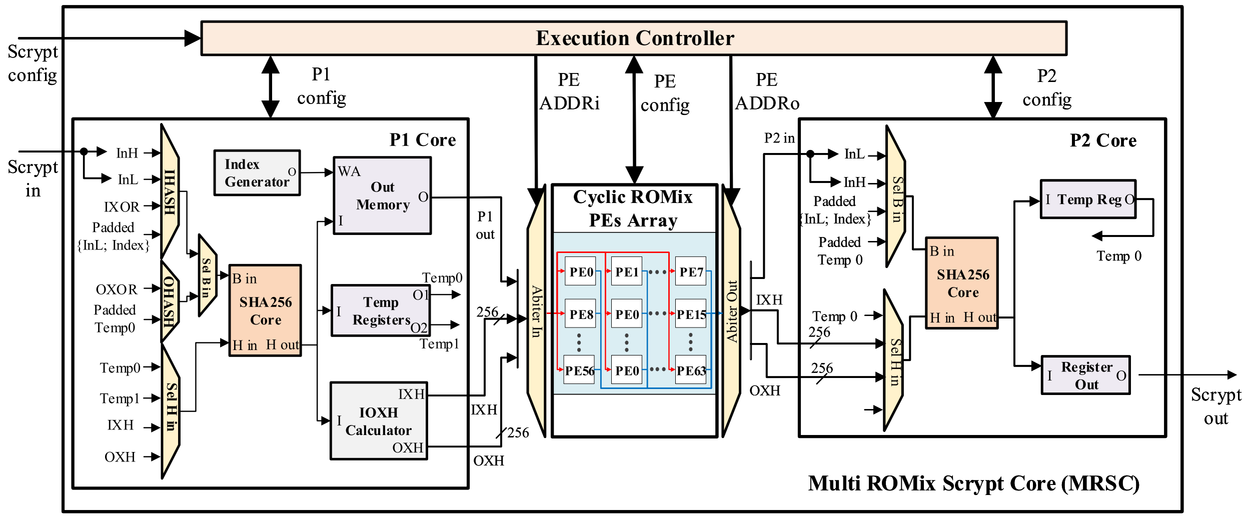


FIGURE 4. The MRSC hardware architecture.

data consist of 4-bit segments that define the configuration parameters r, p, N, and dklen, as defined in Algorithm 1 in Section II. In addition, the Control register stores some control flags for managing the MRSA and specifying its working mode. Moreover, other registers in the CFM are used to store the target threshold and the starting and stop nonces for specifying the mining task. Finally, the output registers store the returned valid nonce and the Scrypt hash output from the MRSA.

The MRSA has two working modes based on the configuration information stored in the CFM: the mining mode and the general mode. In the mining mode, the MRSA first receives the block header input from the CFM at the addresses 0 × 2C ... 0 × 78. Then, the SIG initiates the nonce value from the Start Nonce register (0 × 7C) and automatically increases the nonce if the result is invalid. A result is returned only when the value in the Scrypt Out register (0 × 84 ... 0 × A4) is lower than that in the Target register (0 × 0C ... 0 × 28) or the nonce is increased above the configured Maximum Nonce (0 × 80). The transmission and data processing times in the mining mode are significantly reduced because the MRSA can generate the increased nonce itself without obtaining new input from the host PC. In the general mode, the MRSA continuously takes inputs from the host PC and stores them in the IDM region. In this mode, the SIG is disabled, and the input is obtained directly from the IDM region. Accordingly, Scrypt results are returned one by one to the host PC for each set of input data. The general mode is suitable for high-performance applications such as edge computing nodes [44], which need to generate security keys with large arbitrary and random inputs.

B. MULTI ROMix SCRYPT CORE (MRSC)

In the Scrypt algorithm, ROMix is the most time-consuming process. It accounts for approximately 98% of the total execution time in the conventional Scrypt core (CVSC), which does not applying the pipeline technique. Therefore, we propose the MRSC hardware architecture to speed up the ROMix process, thereby drastically increasing the overall hashing performance of the MRSA.

Fig. 4 presents an overview of the hardware architecture of the MRSC. It consists of a first PBKDF2 core (P1 Core), a cyclic ROMix PE array, a second PBKDF2 core (P2 Core), and the Execution Controller. The Execution Controller includes module counters, decoders, and multiplexers. It receives external configuration signals from the CFM; manages the P1 Core, cyclic ROMix PE array, and P2 Core; and returns the status signals. It also controls the arbiters to manage the data flow for the ROMix PEs in the cyclic ROMix PE array.

With the pipeline technique, the P1 Core processes its inputs and distributes them sequentially to the ROMix PEs because the ROMix PE execution time is sixty-four times longer than that of the P1 Core. Fig. 5 shows the timing chart of the MRSC, which illustrates this more clearly. Accordingly, the numbers of execution cycles of the P1 Core, a ROMix PE, and the P2 Core are 873, 55872, and 267, respectively. Whenever a result is available, the P1 Core passes it to an idle ROMix PE. After successfully passing the output data to a ROMix PE, the P1 Core can continue receiving and processing the next input, and the next output will be transmitted to the next ROMix PE. The transmitted input proceeds in order from ROMix PE 0 to ROMix PE 63. Once the P1 Core finishes the computation for the 65th input, ROMix PE 0 has produced the result for the 1st input and is ready to process the 65th input from the P1 Core. Before processing the next input, however, ROMix PE 0 must transmit the previous output to the P2 Core to compute the final Scrypt result. Because its computation time is much shorter than that of the P1 Core, the P2 Core always completes its work in time to receive input from the next ROMix PE.

The distribution of data by the P1 Core and the reception of input by the P2 Core act as a circle. This circle is established when the P1 Core finishes processing the first

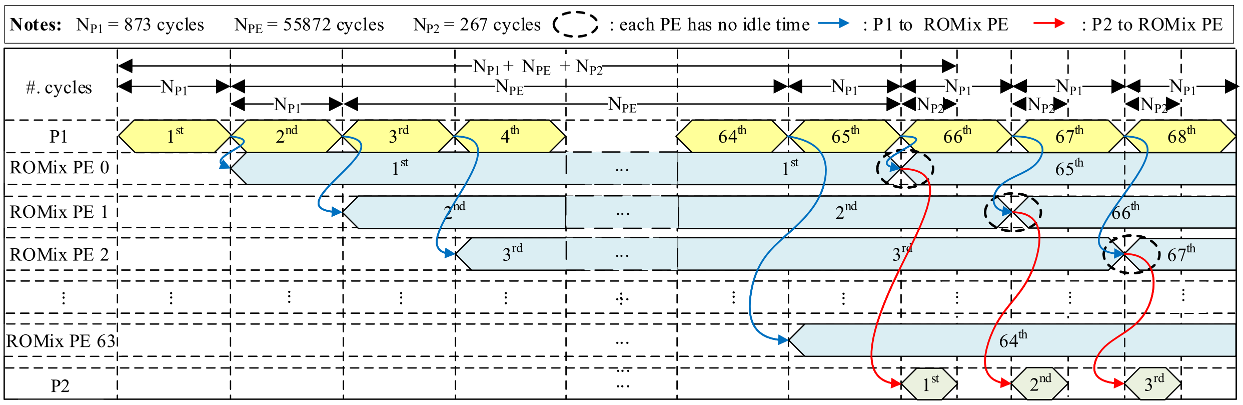

Notes: NP1 = 873 cycles NPE = 55872 cycles NP2 = 267 cycles : each PE has no idle time : P1 to ROMix PE : P2 to ROMix PE
#. cycles
NP1 + NPE + NP2
NP1
NPE
NP2
P1
ROMix PE 0
ROMix PE 1
ROMix PE 2
ROMix PE 63
P2
1st 2nd 3rd 4th 64th 65th 66th 67th 68th

FIGURE 5. The timing chart of the MRSC.

sixty-four inputs. The MRSC also reaches the highest hash rate, called the saturated hash rate, at this time. In the CVSC, the execution cycles of P1 Core, ROMix Core, and P2 Core are 873, 55872, and 256 cycles, respectively, for 1.53%, 98%, and 0.47% of the total execution time. When applying the pipeline technique to Multi ROMix Scrypt Core, P1 Core is not executed in parallel. The parallel pipeline execution includes ROMix Core (ROMix PE) and P2 Core occupied 98.47% of the total execution time. Basically, ROMix and P2 processes can be combined as a parallel process, although MRSC has only one P2 Core. According to Amdahl's law, the theoretical speedup of MRSC can be approximately sixty-six times faster than CVSC [45]. Regarding hardware resources, the MRSC saves sixty-three P1 and P2 Core pairs compared to sixty-four separate CVSCs. Hence, the MRSC is larger than the CVSC only by a factor of approximately thirty. This significantly reduces the hardware cost and increases the energy efficiency of the MRSC, as will be discussed and presented in more detail in Section IV.

If all ROMix PEs were to use one shared external memory, congestion problems would occur due to the limited memory bandwidth. When the MRSC is running, the ROMix PEs operate independently, so the memory they use for computation should preferably be separated. Therefore, in the MRSC, each ROMix PE uses its own 128 kB local memory (LMM), as shown in Fig. 6. Accordingly, the ROMix PEs can access their LMMs simultaneously. This is one of the most important features that helps the MRSA implemented on FPGAs be faster than CPU and GPU Scrypt miners. Each 128 kB LMM contains one thousand twenty-four 1024-bit memory cells. This local memory is implemented on the FPGA using block random access memory (BRAM) resources. It stores all writing-phase results and provides random addresses for the reading phase in the ROMix process. Current UltraScale FPGA lines, such as ALVEO Data Center Accelerator Cards, provide sufficient BRAM resources for implementing the MRSC, and their architecture is optimized for pipeline processing.

Overall, this proposed configurable architecture not only increases flexibility but also avoids a long reconfiguration time for programming new designs from scratch again on the FPGA because of parameter changes.

C. RESCHEDULING TECHNIQUE

In the PBKDF2 function, there are several loops of the HMAC function that produce identical results. If these results can be reused, the number of SHA-256 computations will be reduced, and the processing speed will significantly increase. Therefore, we apply a rescheduling technique in the MRSA to take advantage of this potential for optimizing both hash performance and hardware resources.

The first PBKDF2 execution includes $(N \times r \times p)/256$ HMAC loops, and the last PBKDF2 execution performs dklen/256 HMAC loops. Through analysis, we have found that the SHA-256 hash results for the first 512-bit block of data in step 8 (IXOR) and step 9 (OXOR) in Algorithm 2 are identical for all remaining HMAC loops in both the P1 and P2 Cores. As shown in the diagram of the MRSC hardware architecture presented in Fig. 4, we denote the first 512-bit block SHA-256 hashes of IXOR and OXOR by IXH and OXH, respectively. When the first HMAC loop in the P1 Core finishes, the IXH and OXH results can be stored and reused for the remaining HMAC loops. Accordingly, IXH and OXH are passed through the ROMix PEs via the pipeline flow and transmitted to the P2 Core along with the ROMix PE results. In this way, a significant number of SHA-256 calculations can be eliminated, and the processing speed for the entire MRSA is also significantly increased. This is because SHA-256 is one of the most time-consuming processes in Scrypt. Accordingly, the number of SHA-256 cores in both the P1 Core and the P2 Core is reduced to one, not three as in [27], which helps reduce the size of the entire MRSC. This is achieved by means of the Execution Controller and some intermediate temporary registers, which have the following functions: (1) controlling the multiplexers to correctly select the input for the SHA-256 core, (2) enabling the registers and function blocks for the storage of the SHA-256 core's result, and (3) generating the status signals for the entire core. The notable modules include the IOXH and Out Memory modules. The IOXH module is responsible for calculating

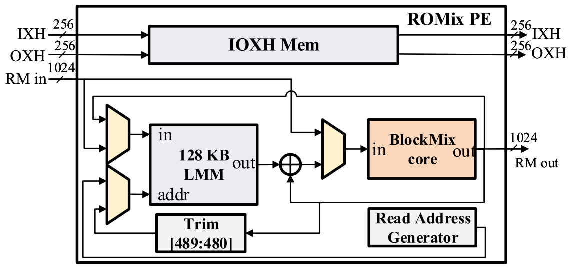


FIGURE 6. The ROMix PE hardware architecture.

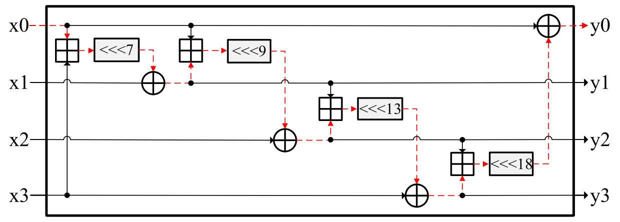


FIGURE 7. The critical path of a ROMix PE.

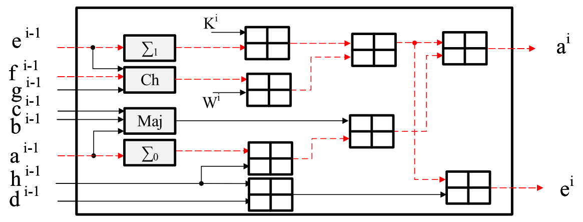


FIGURE 8. The critical path of the P1 and P2 Cores.

IXOR and OXOR and storing IXH and OXH. The Out Memory module stores and concatenates the output of the 256-bit HMAC loops to create the final 1024-bit output of the P1 Core.

As presented in Algorithm 4, BlockMix consists of 2 × r loops, and each loop performs one Xor, one addition, and one Salsa20/8 calculation. The Salsa20/8 function consists of four CRs and four RRs that are performed alternately. Each CR or RR consists of four QRs that are performed in parallel.

The red dashed arrows in Fig. 7 and 8 show the critical paths of a ROMix PE and the P1 and P2 Cores, respectively. These critical paths lie within the QR and SHA-256 processes. Although it is possible to split a QR into many stages to reduce the critical path, the total number of execution cycles will also increase by a factor of many. Consequently, the total number of execution cycles of the entire ROMix PE will similarly increase by a factor of many. This also occurs with the P1 and P2 Cores when shortening the SHA-256 critical path. After the estimation and implementation processes, we find that shortening the critical path cannot increase the MRSC processing speed because the number of execution cycles also increases.

Fig. 9(a) shows the conventional BlockMix core hardware architecture presented in [27]. It uses the CR and RR modules to perform eight alternating column rounds and row rounds. This paper presents a proposal to reduce the hardware resources consumed for the BlockMix core. The proposed BlockMix core hardware architecture is illustrated in Fig. 9.

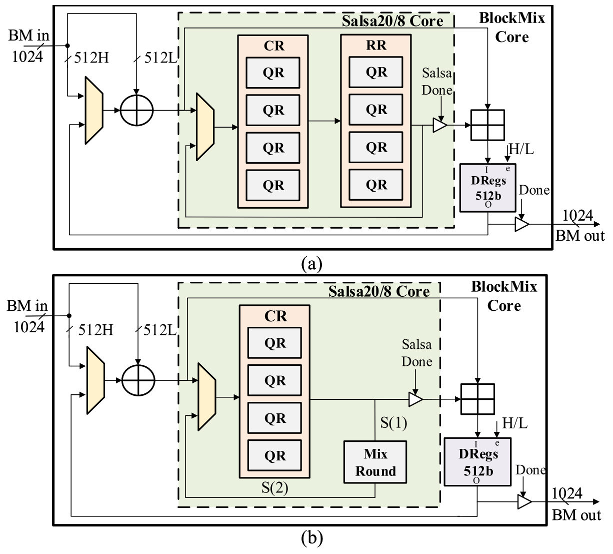


FIGURE 9. The BlockMix core hardware architecture: (a) the conventional BlockMix core; (b) the proposed BlockMix core.

The RR module is removed and replaced by the Mix Round module, while the proposed core still performs the same function as the conventional BlockMix core. In the first loop, the CR module performs a column round. Its result, referred to as the signal S(1), passes through the Mix Round module and provides feedback for the CR module, referred to as the signal S(2). In the next loop, the CR module performs a row round, and the Mix Round module generates feedback for the next column round. In this way, after eight loops, the CR and Mix Round modules have calculated eight interleaved column rounds and row rounds using fewer hardware resources. Essentially, the Mix Round module is a small and simple module for reordering the 512-bit signal S(1) into the signal S(2) as shown in the following equations, where the subscripts are the indexes of the 32-bit segments.

$$\begin{aligned}
&(S(2)_0, S(2)_1, \ldots, S(2)_{15})\\
&\quad = Mix(S(1)_0, S(1)_1, \ldots, S(1)_{15})\\
&S(2)_0 = S(1)_6;\quad S(2)_1 = S(1)_9;\quad S(2)_2 = S(1)_{12}\\
&S(2)_3 = S(1)_3;\quad S(2)_4 = S(1)_{10};\quad S(2)_5 = S(1)_{13}\\
&S(2)_6 = S(1)_0;\quad S(2)_7 = S(1)_7;\quad S(2)_8 = S(1)_4\\
&S(2)_9 = S(1)_1;\quad S(2)_{10} = S(1)_{14};\quad S(2)_{11} = S(1)_{11}\\
&S(2)_{12} = S(1)_2;\quad S(2)_{13} = S(1)_5;\quad S(2)_{14} = S(1)_8\\
&S(2)_{15} = S(1)_{15}
\end{aligned}$$

Compared with that of the conventional BlockMix core, the hardware resource consumption of the proposed core is reduced by approximately half because the Mix Round module is very simple. Moreover, reducing the hardware resources necessary for the BlockMix core significantly helps in reducing the hardware resources necessary for the entire MRSA because a BlockMix core is located inside each ROMix PE.

In general, because ROMix is the function that takes the most computation time in Scrypt, acceleration for the P1

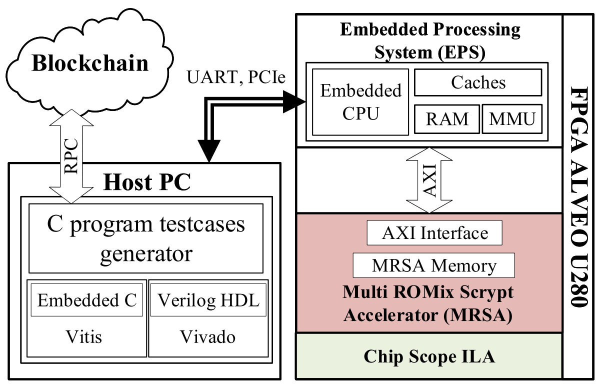


FIGURE 10. The embedded SoC on the Xilinx ALVEO U280 FPGA.

and P2 Cores is not necessary. Therefore, the proposals for the PBKDF2 cores presented in this research aim to minimize the computational resources used while still achieving the required number of execution cycles, as mentioned in Section III-B.

IV. EVALUATION AND EXPERIMENTAL RESULTS

In this section, we present MRSA implementation and verification on the ALVEO U280 FPGA. In addition, the proposed MRSA is evaluated, analyzed, and compared with CPUs, GPUs and FPGA-based designs. We do not compare our proposed work with ASIC-based designs because of the following reasons. First, to the best our knowledge, no academic research of ASIC-based designs was proposed for our comparison. Second, the current ASIC-based designs are mostly commercial ASIC miners for blockchain mining, whose specifications (chip numbers, chip architecture, single-chip area, etc.) are not published for our evaluations. Third, our proposed accelerator is aimed at multi-applications and designed towards standalone users to increase the decentralization of the blockchain network, which is unable for currently commercial ASIC miners.

A. MRSA IMPLEMENTATION AND VERIFICATION ON REAL HARDWARE

Fig. 10 shows the embedded SoC design on a Xilinx ALVEO U280 FPGA developed for the proposed MRSA to prove its correctness and efficiency on real hardware. The system consists of two main devices: a host PC and a Xilinx ALVEO U280 Data Center Accelerator Card.

The host PC includes a testcase generator, an embedded C program, and a Verilog hardware description. It exchanges data with the FPGA through UART and PCIe cables. The host PC runs the testcase generator to obtain test data from real blockchain networks through the Remote Procedure Call (RPC) protocol. Specifically, the test generator obtains a set of block header inputs as test data. This data set is used for verifying the MRSA hardware. The host PC uses the Vitis tool to embed a C code program to configure and prepare the input for the MRSA on the ALVEO U280 FPGA. Moreover, the host PC uses the Vivado tool to load the Verilog hardware description code onto the ALVEO U280 card.

The design on the ALVEO U280 FPGA includes three main intellectual property cores (IPs): an embedded processing system (EPS), the MRSA, and a ChipScope Integrated Logic Analyzer (ChipScope ILA). The EPS consists of a MicroBlaze embedded processor and storage resource components. It receives embedded C code and input data from the host PC via a Xilinx Virtual JTAG (or PCIe) cable and a UART cable, respectively. The EPS sends the configuration and input data to the MRSA IP via an AXI bus. Essentially, the EPS serves as a bridge to exchange intermediate data between the host PC and the MRSA. Finally, the MRSA IP is a version of our proposed design with 64 ROMix PEs on the ALVEO U280 FPGA. It uses the AXI interface to control the transmission and reception of data with the EPS and decides where to store the received data in the MRSA memory. Finally, the ChipScope ILA is a supported IP from Xilinx used to check the output value returned from the MRSA. We use the Xilinx Vivado Design Suite tool (version 2019.2) to implement this experimental SoC. The system operating frequency supplied for all three IPs is **100 MHz**.

In the verification process, the input data set is a set of 1,000,000 block headers taken from the Litecoin, Fastcoin, Dogecoin, and Megacoin blockchain networks. The design is considered correct if all Scrypt hashes returned by the MRSA are less than the target value. Our verification of the MRSA includes two processes: functional verification and real hardware verification. In the functional verification process, the MRSA hardware design is tested with the functional simulation system of the Vivado tool. The transmissions of all test and configuration data are controlled by testbench modules. In the real hardware verification process, the MRSA hardware design is tested in practice on the Xilix ALVEO U280 FPGA SoC. For this test, the host PC generates the test data set and controls the ALVEO U280 FPGA to help it execute the MRSA design correctly. The ChipScope ILA captures all of the input and output signals for verification. Our verification results show that in both functional and real hardware verifications, our MRSA achieves a correct rate of 100%. This experiment demonstrates that our MRSA can be applied as real mining hardware in cryptocurrency blockchain networks.

B. EFFICIENCY EVALUATION: MRSA VS. STATE-OF-THE-ART CPUs AND GPUs

To prove the high efficiency of the MRSA, we designed and implemented C and CUDA Scrypt software to run the same verification task for 1,000,000 block headers on two Nvidia GPUs (Tesla V100 and RTX 3090) and two Intel CPUs (i9-10940X and i7-3970X). These devices were selected for implementation because they are the fastest and most popular devices for performing the blockchain mining task at present. The numbers of processing threads for the best performance on the Tesla V100 GPU, the RTX 3090 GPU, the i9-10940X CPU, and the i7-3970X CPU were 16384, 16384, 28, and 12, respectively. The experimental results of these devices and our MRSA are shown in Table 3. Specifically, the energy

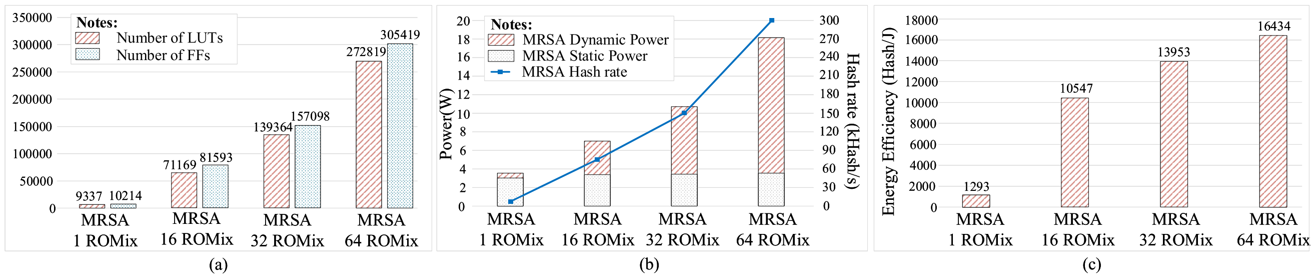


FIGURE 11. Graphs for the quantitative evaluation of the MRSA design on the Xilinx ALVEO U280 FPGA: (a) hardware resources; (b) power consumption and hash rate; (c) energy efficiency.

TABLE 3. Comparison of the results of the MRSA SoC and Scrypt software run on CPUs and GPUs.

Device	Semi. Tech. (nm)	Hash rate (kHash/s)	Power (W)	Energy efficiency (Hash/J)
Prop. MRSA 64-ROMix ALVEO U280	**16**	**114.6**	**8.8**	**13018.25**
GPU: Tesla V100	12	66.7	125	533.6
GPU: RTX 3090	8	81.3	330	246.4
CPU: i9-10940X	14	1.8	120	15
CPU: i7-3970X	32	1.5	119	12.6

efficiency of the ALVEO SoC for our MRSA design with 64 ROMix PEs is **24.4 times** (13018.25 vs. 533.6), **52.83 times** (13018.25 vs. 246.4), **867.88 times** (13018.25 vs. 15), and **1033.2 times** (13018.25 vs. 12.6) higher than those of the Tesla V100 GPU, the RTX 3090 GPU, the i9-10940X CPU, and the i7-3970X CPU, respectively. Moreover, the semiconductor technology used in the Xilinx ALVEO U280 FPGA is 16 nm, while the Tesla V100 GPU, the RTX 3090 GPU, and i9-10940X CPU use 12 nm, 8 nm, and 14 nm semiconductor technologies, respectively. Apparently, the MRSA SoC on the ALVEO U280 offers superior power efficiency and hash rate compared with the most powerful commercial CPUs and GPUs. This gap is even more pronounced when compared to current state-of-the-art CPUs.

C. EFFICIENCY EVALUATION: MRSA VS. STATE-OF-THE-ART FPGA-BASED DESIGNS

In this section, we present an efficiency evaluation of MRSA with related FPGA-based works. In addition, quantitative evaluation of MRSA versions on different FPGAs is clearly presented. As evaluation criteria, we considered the hardware resources, hash rate, throughput, power, and energy efficiency.

1) COMPARISON WITH RELATED FPGA-BASED WORKS

To prove efficiency and performance improvements, the MRSA version with 1 ROMix PE and the version with 32 ROMix PEs are compared with related works based on the Xilinx Virtex 7 FPGA synthesis results. To the best of our knowledge, there is only one related work on developing FPGA-based Scrypt hardware architecture, particularly the accelerator in [27].

The authors in [27] applied a pipeline technique for their Scrypt accelerator with dual ROMix cores. Because of utilizing more ROMix cores (dual ROMix cores), their accelerator hash rate is higher than that of the MRSA version with 1 ROMix PE by **1.95 times** (5.33 vs. 2.74 kHash/s). However, their levels of LUT, FF, and BRAM resource consumption are **5.18 times** (48626 vs. 9389), **7.49 times** (78884 vs. 10585), and **2 times** (57 vs. 28.5) higher, respectively. As a result, their energy efficiency is **6.08 times** lower than that of the MRSA version with 1 ROMix PE (900 vs. 4986 Hash/J).

On the other hand, the MRSA version with 32 ROMix PEs used the number of FFs, LUTs, and BRAMs that are **3.28 times** (159434 vs. 48626), **1.99 times** (156934 vs. 78884), and **16 times** (912 vs. 57) higher than the accelerator in [27]. In return, its hash rate is higher **16.77 times** higher (89.38 vs. 5.33 kHash/s) and its power efficiency is **14.13 times** higher (12721 vs. 900 Hash/J).

2) QUANTITATIVE EVALUATION ON DIFFERENT FPGAs AND MRSA VERSIONS

To demonstrate that the proposed MRSA hardware architecture is compatible and stable for high efficiency on FPGAs, we synthesized our MRSA design on several Xilinx FPGA devices (ALVEO U280, Virtex 7, and Kintex UltraScale). Due to the diverse hardware resources of the different FPGA devices, we built multiple MRSA versions with different numbers of ROMix PEs. Based on this evaluation, we will demonstrate that the performance and energy efficiency of the hardware is proportional to the number of ROMix PEs in each designed MRSA version.

On the Xilinx ALVEO U280 FPGA, we tested four MRSA versions: with 1 ROMix PE, 16 ROMix PEs, 32 ROMix PEs, and 64 ROMix PEs. Fig. 11 shows the quantitative evaluation of these MRSA versions based on the FPGA synthesis results for the ALVEO U280 in Table 4. Fig. 11(a) shows that the hardware resources used increase from the MRSA version with 1 ROMix PE to the version with 64 ROMix PEs. Compared to the version with 1 ROMix PE, the version with 64 ROMix PEs has **29.2 times** as many flip-flops (FFs) (272819 vs. 9337) and **29.9 times** as many lookup tables (LUTs) (305419 vs. 10214). Although the total power consumption increases by **5.12 times** (18.12 vs. 3.53 W), the version with 64 ROMix PEs also has a **65.3 times** higher hash

TABLE 4. FPGA synthesis results for the proposed MRSA in various state-of-the-art FPGA-based designs.

Device	Design	LUTs	FFs	BRAMs	Frequency (MHz)	# Cycles per hash	Hash rate (kHash/s)	Throughput (Mbps)[a]	Power [b] TP/DP (W)	EnEf [c] (Hash/J)
ALVEO U280	MRSA 1 PE	9337	10214	28.5	259.94	57012	4.56	4.67	3.53 / 0.39	1293
	MRSA 16 PEs	71169	81593	456	259.94	3492	74.44	76.22	7.06 / 3.84	10547
	MRSA 32 PEs	139364	157098	912	259.94	1746	148.88	152.45	10.67 / 7.36	13923
	MRSA 64 PEs	272819	305419	1824	259.94	873	297.76	304.9	18.12 / 14.63	16434
Virtex 7	[27]	48626	78884	57	339.10	63656	5.33	5.45	5.916 / 5.623	900
	MRSA 1 PE	9389	10535	28.5	156.05	57012	2.74	2.8	0.55 / 0.3	4986
	MRSA 32 PEs	159434	156934	912	156.05	1746	89.38	91.5	7.03 / 6.66	12721
Kintex UltraScale	MRSA 1 PE	8947	10188	28.5	174.55	57012	3.06	3.14	0.98 / 0.35	3134
	MRSA 32 PEs	142515	157128	912	174.55	1746	99.97	102.37	8.314 / 7.56	12025

[a] The throughput is calculated as (1024 × frequency) / (# cycles per hash)
[b] The power consumption (W) is shown as the total power (TP) and dynamic power (DP)
[c] The energy efficiency (Hash/J) is the ratio between the hash rate and the total power

rate (297.76 vs. 4.56 kHash/s), as shown in Fig. 11(b). Based on the graph in Fig. 11(c), the energy efficiency of the MRSA version with 64 ROMix PEs is higher than that of the version with 1 ROMix PE by **12.71 times** (16434 vs. 1293 Hash/J). The hardware cost of the MRSA version with 64 ROMix PEs is increased only by close to **30 times** (29.2× FFs and 29.9× LUTs) because the PBKDF2 modules are shared among the ROMix PEs. However, the version with 64 ROMix PEs shows improvements of **65.3 times** in hashing performance and **12.71 times** in energy efficiency compared to the MRSA design with 1 ROMix PE. Obviously, the dynamic power of the version with 1 ROMix PE is much smaller than the total power (3.53 vs. 0.39 W), significantly lowering the energy efficiency.

On the Xilinx Virtex 7 and Kintex UltraScale FPGAs, the optimal compatible version of MRSA has only 32 ROMix PEs because the resources of these FPGAs, especially in terms of BRAMs, are not sufficient for the version with 64 ROMix PEs. With the Multi ROMix architecture, the number of BRAMs used for each ROMix PE is independent, and these resources cannot be shared; therefore, this is the main criterion that determines which version is best on which FPGA.

On the Xilinx Virtex 7 FPGA, compared with the MRSA design with 1 ROMix PE, the version with 32 ROMix PEs requires **16.98 times** as many FFs (195434 vs. 9389) and **14.9 times** as many LUTs (156934 vs. 10214), respectively. However, its hash rate is **32.62 times** higher (89.34 vs. 2.47 kHash/s), and its energy efficiency is **2.6 times** higher (12721 vs. 4986 Hash/J).

On the Kintex UltraScale FPGA device, the MRSA version with 32 ROMix PEs uses numbers of FFs and LUTs that are **15.93 times** (142515 vs. 8947) and **15.42 times** (157128 vs. 10188) higher than in the version with 1 ROMix PE. In return, its hash rate increases by **32.67 times** (99.97 vs. 3.06 kHash/s), and its energy efficiency is also **3.84 times** higher (12025 vs. 3134 Hash/J) compared to the version with 1 ROMix PE.

Overall, as the number of ROMix PEs increases, the increase in the hash rate is much greater than the increase in the consumption of hardware resources. The energy efficiency also increases significantly compared to the conventional design. Therefore, the proposed MRSA design can achieve higher power efficiency when the most suitable version is chosen for each FPGA device.

D. FLEXIBILITY ADVANTAGES OF MRSA

In this subsection, we discuss the flexibility advantages of the proposed MRSA architecture in two aspects: dynamic configuration and static reconfiguration.

1) DYNAMIC CONFIGURATION

The proposed configurable architecture, described in section III-A, provides our accelerator (MRSA) the flexibility for switching operation modes of the Scrypt algorithm on runtime configuration. This architecture has an impact on both ASIC and FPGA implementation. In ASIC, it enhances the flexibility of the ASIC-based accelerator. In FPGA, it helps to avoid static reconfiguration from scratch in case of unnecessary.

2) STATIC RECONFIGURATION

This kind of flexibility is provided by the nature of FPGA platforms, which allows the accelerator to be reconfigured before runtime to meet the actual requirements by considering the tradeoff between the processing rate and power consumption/hardware cost. Our proposed MRSA allowed static reconfiguration of the number of ROMix Scrypt Cores per accelerator. For example, the ALVEO U280 FPGA implements the MRSA version with 64 ROMix PEs to maximize performance for blockchain mining or the MRSA version with 32/16/8 ROMix PEs to reduce energy costs for data authentication applications. Furthermore, the calculations inside P1, ROMix PEs, and P2 circuits may be reconfigured in the future to adopt the change of the Scrypt algorithm for accommodating security enhancement. In this context, we are not able to do so with the existing ASIC-based accelerators. In short, the static reconfiguration feature of the FPGA allows our proposed MRSA to tradeoff between accelerator processing rate and power consumption/hardware cost to meet the actual requirements, as well as enhance the circuit flexibility to adopt the future change.

V. CONCLUSION

Scrypt is an ASIC-resistant algorithm with many applications in information security, especially in PoW-based blockchain mining, because it helps avoid distributed destructive attacks from ASIC miners. Scrypt requires many iterations along with high computational memory usage. It is mostly implemented on general-purpose hardware such as CPUs and GPUs. However, CPUs and GPUs usually have very slow computation speeds and extremely high power consumption due to their sequential and complex hardware architectures. Moreover, Scrypt poses the risk that ASICs may easily become obsolete and useless because of its parameterizable nature. In this paper, we propose the Multi ROMix Scrypt Accelerator (MRSA) hardware architecture to be implemented on an FPGA hardware platform. By means of optimization techniques such as configurability parameters and working modes, the use of multiple ROMix processing elements with memory near the ALUs, and the rescheduling and reuse of computational resources, the system's energy efficiency is significantly improved. Experimental results for various versions of our MRSA design on FPGA devices have shown its compatibility and high efficiency. In particular, we have implemented the MRSA in hardware on a real ALVEO U280 SoC to verify its accuracy and performance in actual operation. The results show that the power efficiency is improved by **24.4 to 52.8 times** compared to GPUs and by **867.88 to 1033.2 times** compared to CPUs.

Thus, the proposed MRSA design for implementation on FPGAs has partially solved the problems of low performance and high power consumption on CPUs and GPUs. In particular, it helps FPGAs solve almost all of the typical problems and risks posed by Scrypt on ASICs. However, with its computational-memory-intensive nature, the proposed MRSA still has high requirements in terms of BRAM resources. This hinders the implementation of the optimal MRSA version (with 64 ROMix PEs) on moderate-resource FPGA devices such as the Xilinx Virtex 7 and Kintex UltraScale. Therefore, we believe that developing new hardware designs with new techniques and architectures to optimize memory usage for application on cheaper and smaller FPGAs is a promising research trend for the near future.

APPENDIX

The Scrypt software code for implementation on CPUs and GPUs and the synthesized results of the prototype, optimized, and proposed architectures can be found at https://github.com/archlab-naist/Multi-ROMix-Scrypt-Accelerator/.

REFERENCES

[1] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, vol. 107, pp. 841–853, Jun. 2020.

[2] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks, "A brief survey of cryptocurrency systems," in *Proc. 14th Annu. Conf. Privacy, Secur. Trust*, Dec. 2016, pp. 745–752.

[3] C. Kerry, P. Gallagher, and C. Romine, "FIPS PUB 186-4 federal information processing standards publication digital signature standard (DSS)," U.S. Dept. Commerce/Nat. Inst. Standards Technol., Jul. 2013.

[4] R. C. Merkle, "A certified digital signature," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 218–238.

[5] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proc. 19th ACM Conf. Comput. Commun. Secur. (CCS)*, Oct. 2012.

[6] G. O. Karame, E. Androulaki, and S. Capkun, "Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 248, Oct. 2012.

[7] C. Pérez-Solà, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Double-spending prevention for bitcoin zero-confirmation transactions," *Int. J. Inf. Secur.*, vol. 18, no. 4, pp. 451–463, Nov. 2018.

[8] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, Apr. 1988.

[9] P. Kubiak and M. Kutyłowski, "Preventing a fork in a blockchain—David fighting Goliath," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Dec./Jan. 2021, pp. 1044–1051.

[10] S. Kim, Y. Kwon, and S. Cho, "A survey of scalability solutions on blockchain," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1204–1207.

[11] K. Li, H. Li, H. Hou, K. Li, and Y. Chen, "Proof of vote: A high-performance consensus protocol based on vote mechanism & consortium blockchain," in *Proc. IEEE 19th Int. Conf. High Perform. Comput. Commun.; IEEE 15th Int. Conf. Smart City; IEEE 3rd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2017, pp. 466–473.

[12] F. Bravo-Marquez, S. Reeves, and M. Ugarte, "Proof-of-learning: A blockchain consensus mechanism based on machine learning competitions," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastruct. (DAPPCON)*, Apr. 2019, pp. 119–124.

[13] Y. Wang, S. Cai, C. Lin, Z. Chen, T. Wang, Z. Gao, and C. Zhou, "Study of blockchains's consensus mechanism based on credit," *IEEE Access*, vol. 7, pp. 10224–10231, 2019.

[14] S. Nakamoto, "A peer-to-peer electronic cash system," Oct. 2018. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[15] J. Li, N. Li, J. Peng, H. Cui, and Z. Wu, "Energy consumption of cryptocurrency mining: A study of electricity consumption in mining cryptocurrencies," *Energy*, vol. 168, pp. 160–168, Feb. 2019.

[16] K. O'Neal and P. Brisk, "Predictive modeling for CPU, GPU, and FPGA performance and power consumption: A survey," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2018, pp. 763–768.

[17] T. H. Tran, H. L. Pham, and Y. Nakashima, "A high-performance multimem SHA-256 accelerator for society 5.0," *IEEE Access*, vol. 9, pp. 39182–39192, 2021.

[18] H. L. Pham, T. H. Tran, T. D. Phan, V. T. D. Le, D. K. Lam, and Y. Nakashima, "Double SHA-256 hardware architecture with compact message expander for bitcoin mining," *IEEE Access*, vol. 8, pp. 139634–139646, 2020.

[19] L. V. T. Duong, N. T. T. Thuy, and L. D. Khai, "A fast approach for bitcoin blockchain cryptocurrency mining system," *Integration*, vol. 74, pp. 107–114, Sep. 2020.

[20] H. Cho, "ASIC-resistance of multi-hash proof-of-work mechanisms for blockchain consensus protocols," *IEEE Access*, vol. 6, pp. 66210–66222, 2018.

[21] M. H. Ashik, M. M. S. Maswood, A. G. Alharbi, and D. Medhi, "FPoW: An ASIC-resistant proof-of-work for blockchain applications," in *Proc. IEEE Region Symp. (TENSYMP)*, Jun. 2020, pp. 1608–1611.

[22] *Litecoin*. Accessed: May 10, 2021. [Online]. Available: https://litecoin.com

[23] *Dogecoin*. Accessed: May 10, 2021. [Online]. Available: https://dogecoin.com

[24] *Fastcoin*. Accessed: May 10, 2021. [Online]. Available: https://fastcoin.ca

[25] *Megacoin*. Accessed: May 10, 2021. [Online]. Available: https://www.megacoin.eu

[26] *List of Scrypt Crypto Currencies*. Accessed: May 10, 2021. [Online]. Available: https://en.bitcoinwiki.org/wiki/List_of_scrypt_crypto_currencies

[27] L. V. T. Duong, D. V. Hieu, P. H. Luan, T. T. Hong, and L. D. Khai, "Hardware implementation for fast block generator of Litecoin blockchain system," in *Proc. Int. Symp. Electr. Electron. Eng. (ISEE)*, Apr. 2021, pp. 9–14.

[28] C. Percival and S. Josefsson, "The scrypt password-based key derivation function," Internet Eng. Task Force, 2012.
[29] D. Watkins, "Scrypt mining with ASICS," Tech. Rep., 2017.
[30] H. Kirrmann, "Data format and bus compatibility in multiprocessors," *IEEE Micro*, vol. 3, no. 4, pp. 32–47, Aug. 1983.
[31] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, document RFC 2898, Sep. 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2898.txt
[32] D. H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, document RFC 2104, Feb. 1997. [Online]. Available: https://rfc-editor.org/rfc/rfc2104.txt
[33] M. Bellare, R. Canetti, and H. Krawczyk, "Message authentication using hash functions: The HMAC construction," *RSA Lab. CryptoBytes*, vol. 2, no. 1, pp. 12–15, 1996.
[34] S. Frankel and S. G. Kelly, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 With IPsec*, document RFC 4868, May 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4868.txt
[35] A. Visconti and F. Gorla, "Exploiting an HMAC-SHA-1 optimization to speed up PBKDF2," *IEEE Trans. Depend. Sec. Comput.*, vol. 17, no. 4, pp. 775–781, Jul. 2020.
[36] T. Hansen, *U.S. Secure Hash Algorithms (SHA and SHA-Based HMAC and HKDF)*, document RFC 6234, May 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6234.txt
[37] D. J. Bernstein, "The Salsa20 family of stream ciphers," in *New Stream Cipher Designs*. Berlin, Germany: Springer, 2008, pp. 84–97.
[38] (2021). *Bitmain Antminer L7*. Accessed: Sep. 27, 2021. [Online]. Available: https://shop.bitmain.com/product/detail?pid=00020210626153443050GW7uCVy10679
[39] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019, doi: 10.1109/ACCESS.2019.2896108.
[40] A. I. Sanka, M. Irfan, I. Huang, and R. C. C. Cheung, "A survey of breakthrough in blockchain technology: Adoptions, applications, challenges and future research," *Comput. Commun.*, vol. 169, pp. 179–201, Mar. 2021.
[41] T. Ichikura, R. Yamano, Y. Kikutani, R. Zhang, and Y. Nakashima, "EMAXVR: A programmable accelerator employing near ALU utilization to DSA," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, Apr. 2018, pp. 1–3.
[42] J. Iwamoto, Y. Kikutani, R. Zhang, and Y. Nakashima, "Daisy-chained systolic array and reconfigurable memory space for narrow memory bandwidth," *IEICE Trans. Inf. Syst.*, vol. E103.D, no. 3, pp. 578–589, Mar. 2020.
[43] D. Phan, T. H. Tran, and Y. Nakashima, "SHA-256 implementation on coarse-grained reconfigurable architecture," in *Proc. IEEE Symp. Low Power High-Speed Chips*, Japan, Apr. 2020.
[44] C. Jiang, J. Wan, and H. Abbas, "An edge computing node deployment method based on improved k-means clustering algorithm for smart manufacturing," *IEEE Syst. J.*, vol. 15, no. 2, pp. 2230–2240, Jun. 2021.
[45] J. L. Gustafson, "Reevaluating Amdahl's law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, 1988.

VU TRUNG DUONG LE (Graduate Student Member, IEEE) received the degree in IC and hardware design (engineering) from Vietnam National University Ho Chi Minh City (VNUHCM)-University of Information Technology, in 2020. He is currently pursuing the M.S. degree with the Nara Institute of Science and Technology (NAIST), Japan. His research interests include blockchain technology and cryptography.

THI HONG TRAN (Member, IEEE) received the bachelor's degree in physics and the master's degree in microelectronics from Vietnam National University Ho Chi Minh City (VNU-HCM)-University of Science, Vietnam, in 2008 and 2012, respectively, and the Ph.D. degree in information science from the Kyushu Institute of Technology, Japan, in 2014. From January 2015 to September 2021, she was with the Nara Institute of Science and Technology (NAIST), Japan, as a full-time Assistant Professor. Since October 2021, she has been with Osaka City University, Japan, as a full-time Lecturer, and NAIST as a Visiting Associate Professor. Her research interests include digital hardware circuit design, algorithms related to wireless communication, communication security, blockchain technology, SHA-2, SHA-3, and cryptography. She is a Regular Member of IEEE, IEICE, REV-JEC, and others.

HOAI LUAN PHAM (Graduate Student Member, IEEE) received the bachelor's degree in computer engineering from Vietnam National University Ho Chi Minh City (VNUHCM)-University of Information Technology, Vietnam, in 2018. He is currently pursuing the M.S. degree with the Nara Institute of Science and Technology (NAIST), Japan. His research interests include blockchain technology and cryptography.

DUC KHAI LAM received the B.E. and M.S. degrees from Vietnam National University Ho Chi Minh City (VNUHCM)-University of Science, in 2006 and 2011, respectively, and the Ph.D. degree from the Kyushu Institute of Technology, Japan, in 2016. He is currently with VNUHCM-University of Information Technology, serving as a Lecturer and a Researcher. His research interests include wireless communication systems, digital signal processing, ASICs, and VLSI design.

YASUHIKO NAKASHIMA (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in computer engineering from Kyoto University, in 1986, 1988, and 1998, respectively. He was a Computer Architect with the Computer and System Architecture Department, Fujitsu Ltd., from 1988 to 1999. From 1999 to 2005, he was an Associate Professor with the Graduate School of Economics, Kyoto University. Since 2006, he has been a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include computer architecture, emulation, circuit design, and accelerators. He is a fellow of IEICE, a Senior Member of IPSJ, and a member of the IEEE CS and ACM.

A High-Efficiency FPGA-based BLAKE-256 Accelerator for Securing Blockchain Networks

Pham Hoai Luan[1], Thi Hong Tran[1,2], Vu Trung Duong Le[1], and Yasuhiko Nakashima[1]
[1] Nara Institute of Science and Technology, Nara, Japan
[2] Osaka City University, Osaka, Japan
Email: pham.hoai_luan.ox7@is.naist.jp, hong@osaka-cu.ac.jp, le.vu_trung_duong.lp4@is.naist.jp, nakashim@is.naist.jp

***Abstract*—Developing hardware-efficient and high-speed BLAKE-256 hardware has recently been a research trend because BLAKE-256 is today an important hash function for maintaining the security of blockchain networks, such as Decred and HyperCash. However, existing BLAKE-256 circuits still have low performance and hardware efficiency. Therefore, this paper proposes the BLAKE-256 accelerator to achieve high performance and hardware efficiency for securing the blockchain networks. To achieve those goals, the proposed BLAKE-256 accelerator has three novel optimization techniques. First, a fully unrolled datapath architecture is proposed to generate one hash per clock cycle, thus improving the performance. Second, a pipelined arithmetic-logic unit (ALU) is proposed to shorten the critical path. Third, nonce generating and checking block mechanisms (NGB and NCB) are developed to reduce the data transfer time between the CPU and the accelerator, which can improve the total processing rate. Based on our experiments on a Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA at the system-on-chip level, the impact of proposed optimization techniques is clearly proven. Moreover, experimental results on several FPGAs show that our proposed accelerator has significantly better throughput and area efficiency than previous BLAKE-256 architectures.**

***Index Terms*—BLAKE-256, blockchain, security, Decred.**

I. Introduction

The National Institute of Standards and Technology (NIST) initiated an SHA-3 competition to select new hash algorithms to replace old generations such as SHA-1 and SHA-2. In the third round competition, the BLAKE algorithm was one of the final five candidates. With its outstanding security, the BLAKE algorithm, especially the BLAKE-256 function, is widely applied in many security applications, such as hased-based Radio Frequency Identification (RFID) security protocols, hash-based message authentication, password encryption, JPEG image encryption, and digital signature [1]–[3].

Beyond typical security applications, BLAKE-256 has recently been applied in several famous cryptocurrencies, such as Decred [4]. Particularly, the blockchain technology behind cryptocurrencies uses BLAKE-256 to validate transactions, called blockchain mining. For the sake of network security, miners may relentlessly perform the BLAKE-256 computation of the block header to find a valid nonce to make a hash output smaller than the target value, as shown in Fig. 1. To quickly determine a valid nonce, miners need an ultrahigh-performance BLAKE-256 circuit to accelerate the hash computation of the block header. In addition to acceleration for competing favorably in a blockchain network, the BLAKE-256 circuit should be power efficient to make the energy costs do not exceed the mining income. Therefore, developing a high-performance and hardware-efficient BLAKE-256 accelerator has thus been a research trend in recent years.

Conventional studies have proposed various BLAKE-256 architectures to improve the performance and power consumption [5]–[10]. For example, authors in [5], [8]–[10] have proposed compact BLAKE-256 architectures to optimize the area and energy consumption. Specifically, authors in [8], [9] proposed a small arithmetic-logic unit (ALU) including all required operators in parallel and distributed RAM to store enough working variables for the BLAKE-256 computation. In [5] and [10], a four-stage pipelined ALU was developed by harnessing the intrinsic parallelism of the BLAKE-256 function to interleave the calculation of four instances of the G_i function, thus significantly reducing the area of the BLAKE-256 circuit. Despite their significant improvements in the area, the BLAKE-256 circuits in [5], [8]–[10] have very low throughput since their compact ALUs must suffer a large number of cycles for one hash computation. To improve throughput, authors in [6], [7] proposed round-transformation BLAKE-256 cores to perform in a few rounds for generating a hash output. Although the proposed architectures in [6], [7] have dramatically reduced the number of cycles for the BLAKE-256 computation, their throughput was still limited because of high latency. Overall, the major problem in previous BLAKE-256 hardware is the low performance, making them inefficient to be applied for blockchain mining.

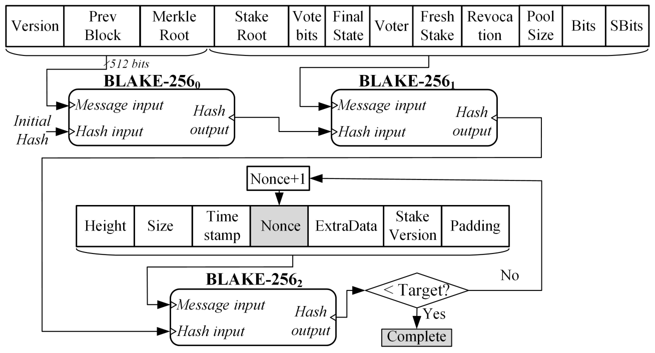


Fig. 1. The BLAKE-256 architecture for block mining.

To address the problems with previous works, this paper proposes a BLAKE-256 accelerator to achieve high performance and hardware efficiency for blockchain mining. Three new optimization techniques are proposed to achieve these goals: the fully unrolled datapath, the pipelined ALU, and nonce generating and checking block mechanisms (NGB and NCB). Besides, the experimental results on several FPGAs prove that our accelerator is significantly better than the existing BLAKE-256 architectures in performance and area efficiency. Finally, our paper is organized as follows: Section II presents the background. Section III describes our proposed BLAKE-256 accelerator in detail. Section IV presents the verification and evaluation of the proposed accelerator on the FPGA. Section V concludes the paper.

Algorithm 1 Hash_out = BLAKE-256 (Message_in, Hash_in)

```
1:  $W_{[0:15]}$ = Message_in; $H_{[0:7]}$ = Hash_in
2:  $V_{[0:15]}$ = **Initialization**($H_{[0:7]}$, $S$, $T$, $C_{[0:7]}$)
3:  **for** r ← 0 to 13 **do**
4:      **Permutation:**
5:          $r' \leftarrow (r \equiv 10)$
6:      **for** i ← 0 to 7 **do**
7:              $W'_i = W_{\sigma_{r'}(2i)} \oplus C_{\sigma_{r'}(2i+1)}$
8:              $W'_{i+1} = W_{\sigma_{r'}(2i+1)} \oplus C_{\sigma_{r'}(2i)}$
9:      **Compression:**
10:         $G_0(V_0, V_4, V_8, V_{12}, W'_0, W'_1)$
11:         $G_1(V_1, V_5, V_9, V_{13}, W'_2, W'_3)$
12:         $G_2(V_2, V_6, V_{10}, V_{14}, W'_4, W'_5)$
13:         $G_3(V_3, V_7, V_{11}, V_{15}, W'_6, W'_7)$
14:         $G_4(V_0, V_5, V_{10}, V_{15}, W'_8, W'_9)$
15:         $G_5(V_1, V_6, V_{11}, V_{12}, W'_{10}, W'_{11})$
16:         $G_6(V_2, V_7, V_8, V_{13}, W'_{12}, W'_{13})$
17:         $G_7(V_3, V_4, V_9, V_{14}, W'_{14}, W'_{15})$
18: $HO_{[0:7]}$ = **Finalization**($V_{[0:15]}$, Hash_in, $S$)
19: **return** Hash_out = $HO_{[0:7]}$
```

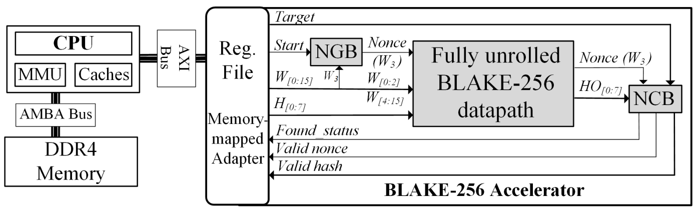


Fig. 2. Overview architecture of the proposed BLAKE-256 accelerator at the system-on-chip level.

II. Background

This section briefly presents the primary points of the BLAKE-256 algorithm that are beneficial to developing our proposed accelerator. Specifically, Algorithm 1 illustrates the BLAKE-256 computation, which consists of two main processes: message expansion and message compression.

Message permutation (MP): The 512-bit message input is spared to sixteen chunks of the 32-bit word (denoted as W_i, 0≤i≤15). In each round, sixteen message words and constants are performed the permutations, which are parameterized by the round index $\sigma_{r'}$. Then, sixteen XOR computations between permuted W_i and constants are executed to produce new sixteen 32-bit words (denoted as W'_i, 0≤i≤15).

Message compression (MC): The message compression compresses the 14 chunks of sixteen W'_i from the MC process into a 256-bit hash output in 14 rounds. Particularly, the MC process includes three steps. First, sixteen internal state values (denoted as V_i, 0≤i≤15) is initialized by a initialization() function. Second, sixteen internal state values V_0, .., V_{15} are calculated and updated based on eight G-functions (denoted as G_j, 0≤j≤7) in 14 rounds. Third, the hash output (denoted as HO_i, 0≤i≤7) is updated by a finalization() function.

The details of the permutation of the $\sigma_{r'}$, G_j(), initialization(), and finalization() functions can be found at [11].

III. Proposed BLAKE-256 Accelerator

After analyzing the BLAKE-256 algorithm, we propose the BLAKE-256 accelerator for blockchain mining. Particularly,

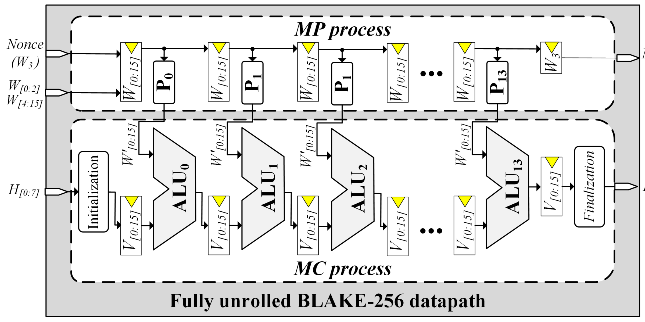


Fig. 3. Fully unrolled BLAKE-256 datapath architecture.

Fig. 2 illustrates the overview architecture of the proposed BLAKE-256 accelerator at the system-on-chip (SoC) level. To control the blockchain mining task, the CPU sends data inputs to the accelerator via AXI bus, including message parsed from the block header (denoted as W_i, 0≤i≤15), hash input (denoted as H_i, 0≤i≤7), the *target* value for comparing with the hash output to find the valid nonce, and the *start* signal for beginning the working session. After completing the mining process, the CPU reads the output data from the proposed BLAKE-256 accelerator, including the *found status* signal to inform whether the valid nonce is found and the *valid nonce* and *valid hash* values. To achieve high performance and hardware efficiency for blockchain mining, the proposed BLAKE-256 accelerator has three novel optimization techniques, which are presented in detail as follows.

A. Optimization 1: Fully Unrolled Datapath Architecture

For improving the performance of the BLAKE-256 accelerator, the loop calculation of MP and MC processes need to be performed fully parallel. Therefore, this section proposes the fully unrolled datapath architecture for the Blake-256/512 accelerator to perform the loop calculation in parallel.

Fig. 3 describes the fully unrolled BLAKE-256 datapath architecture, where the MP and MC processes are unfolded to 14 pipeline stages. Precisely, the MP process includes 14 groups of sixteen variable registers for message words W_0, .., W_{15} and 14 permutation blocks (denoted as P_r, 0≤r≤13), where each P_r block performs the permutation computation in the r^{th} loop. The MC process includes sixteen working variable registers for internal states V_0, .., V_{15} and 14 ALU blocks (denoted as ALU_r, 0≤r≤13), where each ALU_r block computes eight G-functions in the r^{th} loop. Moreover, the fully unrolled datapath has initialization and finalization circuits to calculate the initialization() and finalization() functions, respectively. By virtue of the fully unrolled datapath, the proposed accelerator can perform the hash computations for consecutive message inputs and generate one hash output per cycle, thereby accelerating its performance.

B. Optimization 2: Pipelined ALU Architecture

The critical path of each ALU_r block must go through two G-function blocks, which contain twelve 32-bit adders, eight 32-bit XORs, and eight rotation operators, resulting in low operating frequency and limited throughput. To shorten the critical path, we propose applying the pipeline technique for the ALU architecture.

Fig. 4 shows the pipelined ALU architecture. Specifically, each ALU includes eight G-functions, where the first four G-functions compute in parallel and then pass the results to the last four G-functions. To halve the critical path, we place the registers between the first and last four G-functions. On the other

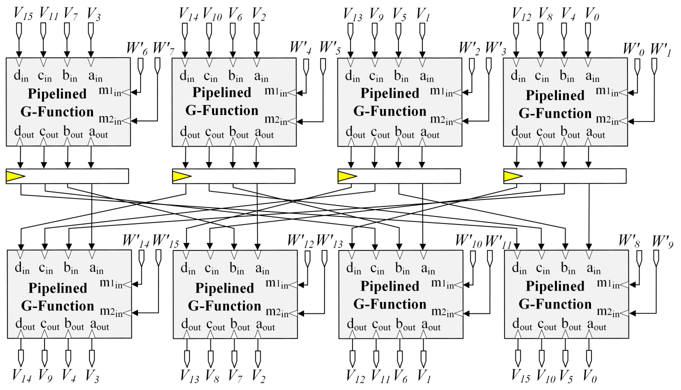


Fig. 4. Pipelined ALU architecture with eight pipelined G-functions.

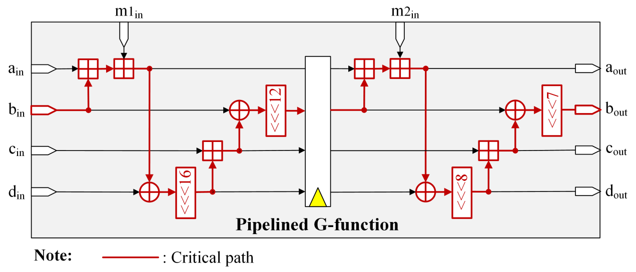


Fig. 5. Inside architecture of each pipelined G-function.

hand, we also apply the pipeline technique to eight G-functions to minimize the critical path, as shown in Fig. 5. Particularly, each G-function is divided into two pipeline stages, where the critical path of each pipeline stage only passes through three 32-bit adders, two 32-bit XOR gates, and two rotation operators. Accordingly, each ALU consists of a four-stage pipeline and its critical path is theoretically reduced by a quarter. Note that we must also place four 512-bit registers for the MP process in each round to synchronize the computational data with the four-stage pipelined ALU architecture.

C. Optimization 3: Nonce Generating and Checking Mechanisms

In blockchain mining, the BLAKE-256 accelerator should try all possible instances of 2^{32} 32-bit *nonce* values, equivalent to computing 2^{32} 512-bit messages, to find the valid hash output smaller than the target. However, transferring 2^{32} messages/hash outputs between the CPU and the accelerator via the limited AXI bus bandwidth is a bottleneck, which decreases the total processing rate significantly and makes the proposed optimization techniques 1 and 2 meaningless. Therefore, this section proposes two mechanisms, nonce generating block (NGB) and nonce checking block (NCB), to improve the processing time.

Fig. 6 illustrates two proposed mechanisms, including NGB and NCB. Concretely, bassed on our investigation in the blockchain network, such as Decred, the *nonce* value is located at position W_3 of all messages. Therefore, the NGB is developed to automatically update 2^{32} W_3 values, equivalent to creating 2^{32} messages to the BLAKE-256 computation, as shown in Fig. 6 (a). Besides, the NCB is used to compare the hash output of the BLAKE-256 computation with the target value to find a *valid nonce* value, as shown in Fig. 6 (b). If the hash output is less than the target, the *found status* flag will equal 1, and then 32-bit *found nonce* and 256-bit *hash output* values will be written to the register files (Reg. file) for the CPU to check. By virtue of the NGB and NCB

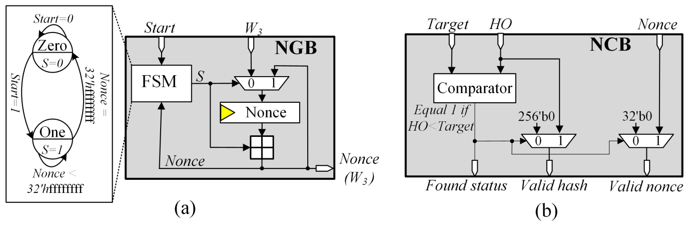


Fig. 6. Two mechanisms: (a) Nonce generating block (NGB) and (b) nonce checking block (NCB).

mechanisms, the accelerator performance for blockchain mining is not reliant on the AXI bus bandwidth, thus achieving 100% hardware efficiency.

IV. Verification and Evaluation

A. FPGA-Based Verification

To prove that the accelerator operates correctly on real hardware, we built the proposed BLAKE-256 accelerator at the system-on-chip level on a Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA, same as shown in Fig. 2. The message input is extracted from the block headers in the actual Decred blockchain network. For verification, the found *found nonce* and *hash output* values from the proposed BLAKE-256 accelerator are compared with the available results on the website of the Decred network. The experimental result shows that our accelerator operates 100% correctly for the blockchain mining process.

B. Evaluating the Impact of the Optimization Techniques

To prove the impact of the proposed optimization techniques on the Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA, we evaluate the throughput, area efficiency, and energy efficiency of designs as the addition of one optimization technique at a time to the proposed accelerator at the system-on-chip level, as shown in Fig. 7. Specifically, the throughput and area efficiency of the accelerator can be improved by adding optimization techniques one by one, as shown in Fig. 7 (a) and (b). Meanwhile, the energy efficiency of the accelerator is only superior when all three optimization techniques are added, as shown in Fig. 7 (c). Overall, the throughput, area efficiency, and energy efficiency of the BLAKE-256 accelerator are only best improved by adding all three proposed optimization techniques.

C. Performance Evaluation

This section presents a performance evaluation between the proposed BLAKE-256 accelerator and previous BLAKE-256 architectures, such as [5]–[10]. For a fair comparison, we have only synthesized the proposed BLAKE-256 accelerator on two Xilinx FPGA boards, including Virtex 5 XC5VLX20T-2 FF323 and Virtex 5 XC6VCX75T-2 FF484. During our experiment, we used a Xilinx ISE 14.7 tool for the synthesis. The comparative factors include throughput and area efficiency.

The throughput, measured in megabit per second (Mbps), is calculated by eq. (1), where *Block_Size* is 512 bits and #Cycle/Hash is the number of clock cycles to create one hash output.

$$\text{Throughput} = \frac{\textit{Block\_Size} \times \text{Frequency}}{\text{\#Cycle/Hash}} \quad (1)$$

Table I shows the throughput and area efficiency comparisons between the proposed BLAKE-256 accelerator and existing BLAKE-256 architectures on the Virtex 5 and Virtex 6 boards.

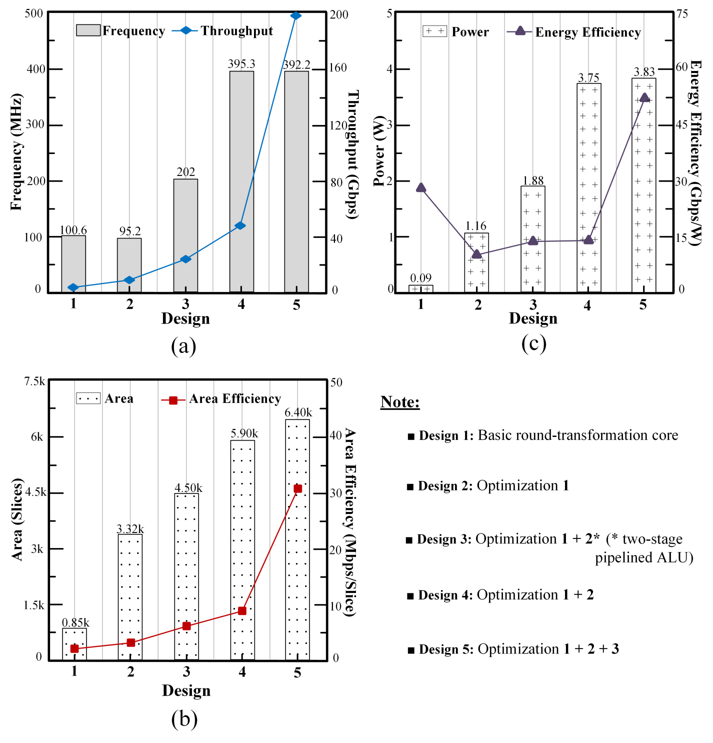


Fig. 7. The (a) throughput, (b) area efficiency, and (c) energy efficiency of designs as the addition of one optimization technique at a time to the proposed accelerator.

Note that the Block RAMs (BRAMs) used in the BLAKE-256 architectures of [5] and [8] are normalized to slices for a fair comparison with other architectures.

On the Virtex 5 board, the proposed BLAKE-256 accelerator utilizes 21,867 slices, operates at a maximum frequency of 218 MHz, and delivers the throughput of 111,616 Mbps. Compared to previous works, the proposed BLAKE-256 accelerator is **496.1 times** (111,616 vs. 225), **73.9 times** (111,616 vs. 1,510), and **41.7 times** (111,616 vs. 2,676) greater than [5], [6], and [7] in throughput, respectively, and **7.1 times** (5.10 vs. 0.72), **3.8 times** (5.10 vs 1.35), and **3.2 times** (5.10 vs 1.61) better than [5], [6], and [7] in area efficiency, respectively.

On the Virtex 6 board, the proposed BLAKE-256 accelerator occupies 125,952 slices, operates at a maximum frequency of 246 MHz, and achieves the throughput of 125,952 Mbps. Compared to the related works, the proposed BLAKE-256 accelerator is **587.5 times** (125,952 vs. 214.4), **1,061.1 times** (125,952 vs. 118.7), and **834.7 times** (125,952 vs. 150.9) better than [8], [9], and [10] in throughput, respectively, and **6.9 times** (5.53 vs. 0.80), **5.5 times** (5.53 vs 1.01), and **1.8 times** (5.53 vs 3.02) higher than [8], [9], and [10] in area efficiency, respectively.

V. Conclusion

Developing a high-performance BLAKE-256 accelerator with high hardware efficiency is an attractive research trend since the BLAKE-256 functions are widely adopted in many speed-demand modern applications, such as blockchain mining. Unfortunately, the existing BLAKE-256 architectures are limited in performance and hardware efficiency. Therefore, this paper proposes the BLAKE-256 accelerator to improve the processing rate and hardware efficiency. To achieve those goals, three optimization techniques are proposed, including the fully unrolled datapath, the four-stage piplined ALU, and nonce generating and checking block mechanisms (NGB and NCB). The accuracy of the proposed BLAKE-256 accelerator is verified on the Xilinx Zynq UltraScale+ MPSoC ZCU102 FPGA. Experimental results on several different FPGA boards prove that the proposed accelerator is significantly better throughput and area efficiency than existing BLAKE-256 architectures.

TABLE I
Performance comparison between the proposed BLAKE-256 accelerator and the existing BLAKE-256 architectures.

Device	Reference	Freq. (MHz)	Area (Slice)	#Cycle/ Hash	Throughput (Mbps)	Area Eff. (Mbps/slice)
Virtex 5	[5]	372	312*	844	225	0.72
	[6]	118	1,118	40	1,510	1.35
	[7]	115	1,660	22	2,676	1.61
	Proposed	**218**	**21,867**	**1**	**111,616**	**5.10**
Virtex 6	[8]	155	267*	370	214.4	0.80
	[9]	274	1,182	117	118.7	1.01
	[10]	349	1,184	50	150.9	3.02
	Proposed	**246**	**22,791**	**1**	**125,952**	**5.53**

*: One Block RAM (BRAM) is normalized to 128 slices, where [5] and [8] use 2 and 1 BRAM, respectively.

Acknowledgment

This work was supported by the Japan Science and Technology Agency (JST) under a Strategic Basic Research Programs Precursory Research for Embryonic Science and Technology (PRESTO) under Grant JPMJPR20M6.

References

[1] P. Li, J. Meng, and Z. Sun, “A new jpeg encryption scheme using adaptive block size,” in *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, J.-S. Pan, J. Li, O.-E. Namsrai, Z. Meng, and M. Savić, Eds. Singapore: Springer Singapore, 2021, pp. 140–147.
[2] I. H. Abdulqadder, S. Zhou, D. Zou, I. T. Aziz, and S. M. A. Akber, “Bloc-sec: Blockchain-based lightweight security architecture for 5g/b5g enabled sdn/nfv cloud of iot,” in *2020 IEEE 20th International Conference on Communication Technology (ICCT)*, 2020, pp. 499–507.
[3] M. Iavich, G. Iashvili, S. Gnatyuk, A. Tolbatov, and L. Mirtskhulava, “Efficient and secure digital signature scheme for post quantum epoch,” in *Information and Software Technologies*, Cham: Springer International Publishing, 2021, pp. 185–193.
[4] Decred-secure. adaptable. sustainable. Accessed: Feb. 18, 2022. [Online]. Available: https://www.decred.org
[5] J.-L. Beuchat, E. Okamoto, and T. Yamazaki, “Compact implementations of blake-32 and blake-64 on fpga,” in *2010 International Conference on Field-Programmable Technology*, 2010, pp. 170–177.
[6] B. Baldwin, A. Byrne, L. Lu, M. Hamilton, N. Hanley, M. O’Neill, and W. P. Marnane, “Fpga implementations of the round two sha-3 candidates,” in *2010 International Conference on Field Programmable Logic and Applications*, 2010, pp. 400–407.
[7] M. Knezevic, K. Kobayashi, J. Ikegami, S. Matsuo, A. Satoh, n. Kocabas, J. Fan, T. Katashita, T. Sugawara, K. Sakiyama, I. Verbauwhede, K. Ohta, N. Homma, and T. Aoki, “Fair and consistent hardware evaluation of fourteen round two sha-3 candidates,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 5, pp. 827–840, 2012.
[8] J.-P. Kaps, P. Yalla, K. K. Surapathi, B. Habib, S. Vadlamudi, and S. Gurung, “Lightweight implementations of sha-3 finalists on fpgas,” in *The Third SHA-3 Candidate Conference*, no. 60, 2012, pp. 1–17.
[9] S. Kerckhof, F. Durvaux, N. Veyrat-Charvillon, F. Regazzoni, G. M. de Dormale, and F.-X. Standaert, “Compact fpga implementations of the five sha-3 finalists,” in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2011, pp. 217–233.
[10] N. At, J.-L. Beuchat, E. Okamoto, s. San, and T. Yamazaki, “Compact hardware implementations of chacha, blake, threefish, and skein on fpga,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 485–498, 2014.
[11] E. Biham and O. Dunkelman, “A framework for iterative hash functions—haifa,” Computer Science Department, Technion, Tech. Rep., 2007.