IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019.DOI

Compact Message Permutation for a Fully Pipelined BLAKE-256/512 Accelerator

HOAI LUAN PHAM[1], THI HONG TRAN[2],VU TRUNG DUONG LE[1], and YASUHIKO NAKASHIMA[1]

[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 (e-mail: 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

Developing a low-cost and high-performance BLAKE accelerator has recently become an attractive research trend because the BLAKE algorithm is important in widespread applications, such as cryptocurrencies, data security, and digital signatures. Unfortunately, the existing BLAKE circuits are limited in performance and hardware efficiency. Therefore, this paper introduces the first fully pipelined BLAKE-256/512 accelerator to improve throughput and hardware efficiency. Moreover, based on the rates of changed words in consecutive message inputs, a compact message permutation scheme is proposed to reduce the area and energy consumption of the fully pipelined BLAKE-256/512 accelerator. To achieve these goals, the compact message permutation scheme includes two novel optimization techniques: register optimization, reducing the number of registers used by over 80% compared to conventional message permutation in a theoretical evaluation, and XOR optimization, decreasing the number of XOR gates by 93.8%. An ASIC-based experiment shows that the proposed compact message permutation scheme helps reduce the area and power consumption by up to 11.35% and 21.10%, respectively, for the fully pipelined BLAKE-256 accelerator and by up to 9.86% and 20.32%, respectively, for the fully pipelined BLAKE-512 accelerator. The correctness of the compact message permutation scheme is verified on a real hardware platform (an Alveo U280 FPGA).

INDEX TERMS Blockchain mining, FPGA, GPU, BLAKE, fully pipelined.

I. INTRODUCTION

THE National Institute of Standards and Technology (NIST) launched the SHA-3 competition to select one or more new hash algorithms with better efficiency and resilience to future attacks to supersede the older SHA-1 and SHA-2 algorithms. In the third round of the competition, only five algorithms were chosen from among the 51 candidates, and one of these five finalists was the BLAKE algorithm. Similar to SHA-2, BLAKE is also a family of hash functions, namely, BLAKE-224, BLAKE-256, BLAKE-384, and BLAKE-512, of which BLAKE-256 and BLAKE-512 are the most widely used. Today, the BLAKE functions are usually applied in generic security applications, such as hash-based radio frequency identification (RFID) security protocols [1], hash-based message authentication [2], [3], password encryption [4], JPEG image encryption [5], and digital signatures [6]. Beyond such generic applications, BLAKE-256 and BLAKE-512 are currently used for the blockchain mining process in many famous cryptocurrencies, such as Decred [7] and Dash [8].

In generic applications such as network security, typical client devices are sufficiently powerful only to execute relatively few hash calculations, while servers need high-performance BLAKE hardware to perform a large number of hash computations to serve requests from clients. In addition, in blockchain mining, miners need ultrahigh-performance BLAKE circuits to maintain the security of the blockchain network and gain additional profits. Therefore, developing a high-performance and hardware-efficient BLAKE circuit has recently become an attractive research trend.

Many studies have proposed various BLAKE architectures based on field-programmable gate arrays (FPGAs) and

application-specific integrated circuits (ASICs) to improve performance and power consumption. For example, to ensure suitability for power-constrained environments such as wireless sensor networks or RFID systems, the authors of [9]–[13] proposed compact BLAKE architectures to optimize area and energy consumption. Specifically, the authors of [9] explored shift-register-based compact hardware architectures for the BLAKE functions to minimize area and energy consumption. Furthermore, [10], [11] introduced a compact BLAKE implementation that used a small arithmetic and logic unit (ALU) embedded with all the required operators in parallel and distributed random access memory (RAM) to store intermediate values, message blocks, and constants. In [12], [13], an ALU with a four-stage pipeline was designed by harnessing the intrinsic parallelism of the algorithm to interleave the calculation of four instances of the G_i function, thereby significantly reducing the area of the BLAKE circuit. However, despite their advantages of low power and small areas, the compact architectures in [9]–[13] had to accept extremely high latency as a trade-off, resulting in very low throughput. In other BLAKE architectures, specific processor-oriented hardware implementations for the BLAKE functions have been proposed to accelerate performance and reduce area costs [14], [15]. Although the processors in [14], [15] were significantly improved in terms of the critical path, those processors still delivered low throughput because of the need to execute numerous instructions to perform a single hash computation. To improve throughput, the authors of [16]–[27] proposed round-transformation BLAKE architectures, which can perform several rounds to generate a hash output. However, the BLAKE architectures in [16]–[27] are still limited in throughput because of their high latency. Overall, the main problem with previous BLAKE architectures is poor performance, making them inefficient to apply in servers that must perform large amounts of hash computations or support modern high-performance applications such as blockchain mining.

To address the problems with related works, this paper introduces the first fully pipelined BLAKE-256/512 accelerator, which can generate one hash value per clock cycle. Although this fully pipelined BLAKE-256/512 accelerator shows outstanding advantages in terms of performance and hardware efficiency, it still suffers from a large area cost and high energy consumption. Therefore, optimizing the area and power consumption is necessary to compensate for the disadvantages of the fully pipelined BLAKE-256/512 accelerator.

In this study, we propose a new approach to reducing the hardware cost and power consumption of the fully pipelined BLAKE-256/512 accelerator. In particular, we classify the sixteen message words of consecutive message inputs into three groups in terms of the word change rate: frequently changed words (CW-2), infrequently changed words (CW-1), and unchanged words (CW-0). Based on the characteristics of CW-1 and CW-0 words, we propose a compact message permutation scheme that significantly reduces the hardware cost required for the message permutation computations of the BLAKE-256/512 functions. Accordingly, the proposed compact message permutation scheme includes two new optimization techniques, namely, register optimization and XOR optimization, which greatly reduce the numbers of registers and XOR gates needed as the number of CW-1 or CW-0 words increases. Experimental results on an ASIC prove that this compact message permutation scheme helps significantly reduce the area and power consumption of the fully pipelined BLAKE-256/512 accelerator as the number of CW-1 or CW-0 words increases, thereby considerably improving both area efficiency and energy efficiency. We have verified the correctness of the compact message permutation scheme on a Xilinx Alveo U280 FPGA. Moreover, experiments on several FPGA boards show that the fully pipelined BLAKE-256/512 accelerator with the proposed compact message permutation scheme is far superior to related works in terms of throughput and area efficiency.

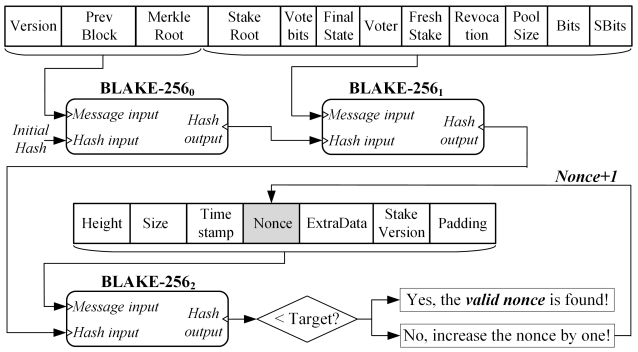

FIGURE 1. High-level diagram of the proposed system.

The remainder of this paper is organized as follows. Section II presents the research background. Section III describes our proposed compact message permutation scheme in detail. Section IV reports our evaluations on the basis of theory as well as ASIC and FPGA experiments. Finally, Section V concludes the paper.

II. BACKGROUND

A. IMPORTANCE OF A HIGH-PERFORMANCE BLAKE ACCELERATOR

To clarify the importance of a high-performance BLAKE accelerator, we analyze a real-world BLAKE application, namely, the cryptocurrency Decred. Concretely, miners in the Decred network use BLAKE-256 to perform hash computations for block headers as a proof of work (PoW) to find a valid block and receive a reward. This process is commonly called blockchain mining. Fig. 1 illustrates the BLAKE-256 architecture for Decred mining, which includes three BLAKE-256 blocks named BLAKE-256$_0$, BLAKE-256$_1$, and BLAKE-256$_2$. Specifically, the message inputs to BLAKE-256$_0$, BLAKE-256$_1$, and BLAKE-256$_2$ are three chunks of 512-bit data, including 1,440 bits of block header information and 96 bits of padding [28]. In Decred mining, the two 512-bit message pieces provided as input to

TABLE 1. Parameters of the BLAKE functions. All sizes are given in bits.

Parameter	BLAKE-224	BLAKE-256	BLAKE-384	BLAKE-512
Digest size (D)	224	256	384	512
Block size (B)	512	512	1024	1024
Message size (M)	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Word size (W)	32	32	64	64
Salt size (S)	128	128	256	256
Round (R)	10/14	8/10/14	14/16	14/16

BLAKE-256_0 and BLAKE-256_1 are not frequently changed because they do not include the 32-bit *nonce* field. Accordingly, BLAKE-256_0 and BLAKE-256_1 need to be executed only once per mining task in the software implementation. Conversely, the 512-bit message input to BLAKE-256_2 is updated frequently because miners must scan all 2^{32} possible *nonce* values to find a hashing output smaller than the target. It has been reported for the Decred network that the BLAKE-256_2 computation must be performed up to 4×10^{17} times on average to successfully discover a valid *nonce*. Therefore, a high-performance accelerator for the BLAKE-256_2 computation is necessary to quickly find a valid *nonce* value.

In addition to BLAKE-256, the BLAKE-512 function is also used for the blockchain mining process in many current cryptocurrencies, such as Dash. Accordingly, a BLAKE-512 circuit with a high processing rate is also needed to speed up blockchain mining for miners. Overall, the development of high-performance BLAKE accelerators has become a research trend in recent years.

B. BLAKE ALGORITHM

Before developing a high-performance BLAKE accelerator, we first investigate the details of the BLAKE algorithm. Specifically, the BLAKE algorithm is built based on a combination of three previously analyzed and reliable components selected by Aumasson et al. [30], including the HAsh Iterative FrAmework (HAIFA) of Biham and Dunkelman [29], the internal structure of the LAKE hash function [31], and the modified version of the ChaCha function presented by Bernstein [32]. The BLAKE algorithm is a family of four hash functions, namely, BLAKE-224, BLAKE-256, BLAKE-384, and BLAKE-512, as shown in Table 1. Since BLAKE-256 and BLAKE-512 are the most widely used of these functions, this study focuses only on BLAKE-256 and BLAKE-512. To distinguish the variants of the BLAKE-256 and BLAKE-512 functions with different numbers of rounds, we denote the BLAKE-256 function with 8, 10, and 14 rounds by BLAKE-256r8, BLAKE-256r10, and BLAKE-256r14, respectively, and the BLAKE-512 function with 14 and 16 rounds by BLAKE-512r14 and BLAKE-256r16, respectively.

Algorithm 1 shows the pseudocode for the BLAKE-256/512 functions, where the values of B, M, W, S, and R are given in Table 1. The calculations of BLAKE-256/512 for a given message input include three steps: padding, message permutation, and message compression.

Algorithm 1 Hash = BLAKE-256/512(Message)

BLAKE-256:
  M consists of N 512-bit padded blocks.
  $H^0_{[0:7]}$: 32-bit square root of the first 8 primes.
  $C_{[0:15]}$: sixteen 32-bit constants.
BLAKE-512:
  M consists of N 1024-bit padded blocks.
  $H^0_{[0:7]}$: 64-bit square root of the first 8 primes.
  $C_{[0:15]}$: sixteen 64-bit constants.
$T_{[0:1]}$ = t_0, t_1 (counter [29])
L = Length_in_bit(Message)
Padding:
  k = B - (1 + W + (L $\mod$ B))
  Pad = {1, zeros(1,k-1), 1, L}
  $M^{[0:N-2]}$ = message[0:((N-2)×B)-1]
  M^{N-1} = {message[(N-2)×B:L-1], Pad}
for t ← 0 to (N-1) **do**
  $V_{[0:15]}$ = **Initialization**(H^t, S, T, $C_{[0:7]}$)
  $W_{[0:15]}$ = M^t
  for r ← 0 to (R-1) **do**
    Permutation:
      $r' \leftarrow (r \equiv 10)$
    for i ← 0 to 7 **do**
      $W'_i = W_{\sigma_{r'}(2i)} \oplus C_{\sigma_{r'}(2i+1)}$
      $W'_{i+1} = W_{\sigma_{r'}(2i+1)} \oplus C_{\sigma_{r'}(2i)}$
    end for
    Compression:
      $G_0(V_0, V_4, V_8, V_{12}, W'_0, W'_1)$
      $G_1(V_1, V_5, V_9, V_{13}, W'_2, W'_3)$
      $G_2(V_2, V_6, V_{10}, V_{14}, W'_4, W'_5)$
      $G_3(V_3, V_7, V_{11}, V_{15}, W'_6, W'_7)$
      $G_4(V_0, V_5, V_{10}, V_{15}, W'_8, W'_9)$
      $G_5(V_1, V_6, V_{11}, V_{12}, W'_{10}, W'_{11})$
      $G_6(V_2, V_7, V_8, V_{13}, W'_{12}, W'_{13})$
      $G_7(V_3, V_4, V_9, V_{14}, W'_{14}, W'_{15})$
  end for
  H^{t+1} = **Finalization**($V_{[0:15]}$, H^t, S)
end for
return Hash = $H^{(N-1)}$

Padding: Padding is performed to construct the last block such that it has the same size as the other blocks. Specifically, if the original message contains L bits, then a "1" bit is appended first, the following k bits are "0" bits, and another "1" bit and the length L are appended as the last bits. The padded message is then divided into N blocks ($M^{[0:N-1]}$) of B bits.

Message permutation: After padding, each of the N blocks is subjected to message permutation processing. First, each block is separated into sixteen chunks of 32/64-bit words (denoted by W_i, $0\leq i\leq 15$). In each round, the sixteen W_i are permuted and then subjected to XOR computations with sixteen constants. The permutations of the sixteen W_i and the sixteen constants are parameterized by the round in-

TABLE 2. Permutations used by the BLAKE functions (reprinted from []).

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

dex $\sigma_{r'}$, as shown in Table II. The sixteen XOR computations between the W_i and the constants return sixteen 32/64-bit permuted words (denoted by W'_i, 0≤i≤15).

Message compression: Essentially, the message compression process compresses the R chunks of sixteen W'_i obtained in the message permutation step into a 256/512-bit hash output. First, sixteen internal states (denoted by V_i, 0≤i≤15) are initialized by the initialization() function. Afterward, the sixteen internal states V_0, ..., V_{15} are computed and updated based on eight G-functions (denoted by G_j, 0≤j≤7) through R rounds. Then, the hash output (H^{t+1}) is updated by the finalization() function. Once the message permutation and compression processes have been completed for block M^t, the H^{t+1} value is used as the hash input to the initialization() function for the computation of the next block (M^{t+1}). Finally, the hash output H^N updated by compressing the last block (M^{N-1}) is the final hash output of the BLAKE-256/512 function.

The details of the G_j(), initialization(), and finalization() functions can be found in [29].

C. NORMAL FULLY PIPELINED BLAKE-256/512 ACCELERATOR

To improve the processing rate of BLAKE functions, the loop computations of the message permutation and compression processes need to be performed fully in parallel. Therefore, this section introduces the first normal fully pipelined BLAKE-256/512 accelerator, which performs the loop computations in parallel.

According to our investigation, no fully pipelined BLAKE architecture has previously been proposed, although the possibility has been mentioned in several related works. On the other hand, many works have proposed fully pipelined SHA-2 architectures to optimize performance and hardware efficiency [33]–[37]. Essentially, a fully pipelined BLAKE architecture is similar to a fully pipelined SHA-2 architecture, which is unfolded into R (where R is the number of rounds) pipeline stages.

Fig. 2 illustrates the implementation of the normal fully pipelined BLAKE-256/512 accelerator, where the message permutation and compression processes are unfolded into R pipeline stages (where R is 8, 10, or 14 for BLAKE-256 or 14 or 16 for BLAKE-512). More precisely, the message compression part includes R compression circuit blocks (denoted by *compression* r, 0≤r≤R-1) and R groups of sixteen working variable registers for the internal states V_0, ..., V_{15}. The message permutation part includes R permutation circuit blocks (denoted by *permutation* r, 0≤r≤R-1) and R groups of sixteen working variable registers for the message words (denoted by REG_r, 0≤r≤R-1). In addition, the accelerator has initialization and finalization circuits to calculate the initialization() and finalization() functions, respectively. By virtue of the unfolding of these fully pipelined stages, the accelerator can compress a large number of adjacent message inputs and deliver one hash output per cycle, thereby accelerating its performance. However, this unfolding greatly increases the numbers of registers and computational circuits required, causing the accelerator to occupy an enormous area and incur massive power consumption. Therefore, optimizing the area and power consumption of the fully pipelined

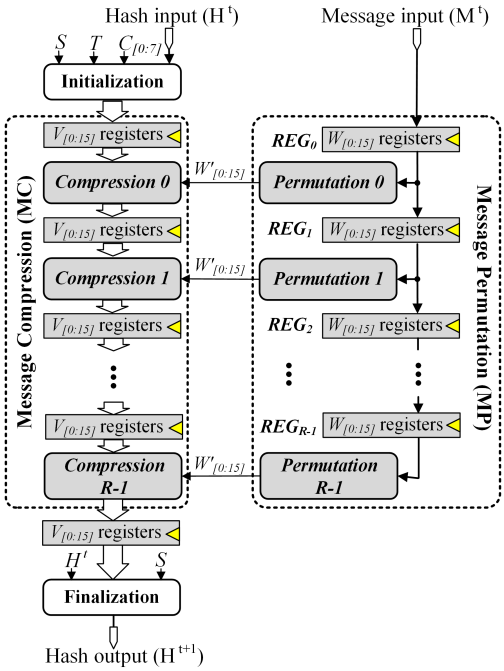

FIGURE 2. Normal fully pipelined BLAKE-256/512 accelerator.

TABLE 3. Numbers of registers and XORs used in the message permutation process and the fully pipelined BLAKE-256/512 accelerator.

Function	Number of registers (bits)		Number of XORs (bits)	
	Message permutation	Entire accelerator	Message permutation	Entire accelerator
BLAKE-256r8	4,096 (**47.1%**)	8,704	4,096 (**32.0%**)	12,800
BLAKE-256r10	5,120 (**47.6%**)	10,752	5,120 (**32.2%**)	15,872
BLAKE-256r14	7,168 (**48.3%**)	14,848	7,168 (**32.6%**)	22,016
BLAKE-512r14	14,336 (**48.3%**)	29,696	14,336 (**32.6%**)	44,032
BLAKE-512r16	16,384 (**48.5%**)	33,792	16,384 (**32.7%**)	50,176

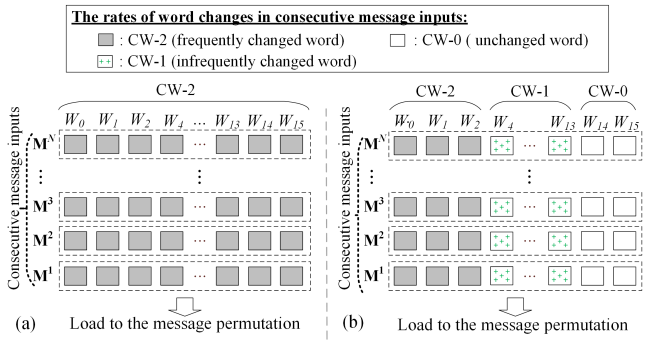

FIGURE 3. Characteristics of consecutive message inputs in the (a) normal and (b) proposed message permutation schemes.

BLAKE-256/512 accelerator is required to achieve high hardware efficiency.

Many previous works have proposed various optimization techniques to improve the message compression process because of its complexity. However, these proposed optimization techniques can only speed up the performance of the fully pipelined BLAKE-256/512 accelerator without reducing its area and energy consumption. Meanwhile, although the message permutation process is not complicated, the numbers of registers and XORs needed are significantly large. As shown in Table 3, the numbers of registers and XORs in the message permutation part account for more than 47.1% and 32%, respectively, of those in the entire accelerator. Therefore, reducing the numbers of registers and XORs needed for message permutation can significantly improve the area and power consumption of the fully pipelined BLAKE-256/512 accelerator.

D. PRELIMINARY IDEA FOR THE MSA

By virtue of the unfolding of the pipeline stages, the fully pipelined BLAKE-256/512 accelerator is able to process a long series of consecutive message inputs. By analyzing the rates of word changes in consecutive message inputs, we can classify the sixteen message words into three groups: frequently changed words (CW-2), infrequently changed words (CW-1), and unchanged words (CW-0). Because each pipeline stage contains sixteen registers and sixteen XORs, the normal message permutation scheme can process consecutive message inputs with all sixteen message words belonging to the CW-2 group, as shown in Fig. 3 (a). Fundamentally, since message words in the CW-2 group have continuously changing and arbitrary values, all registers and XORs for these message words must be retained and cannot be optimized. However, in several applications, such as blockchain mining, consecutive message inputs include words of all three change rates, as shown in Fig. 3 (b). Based on the characteristics of little or no time variation of CW-1 and CW-0, two ideas for optimizing the message permutation part of the accelerator are introduced below.

Idea 1: Register optimization. Since message words in the CW-0 group remain completely unchanged in all hash

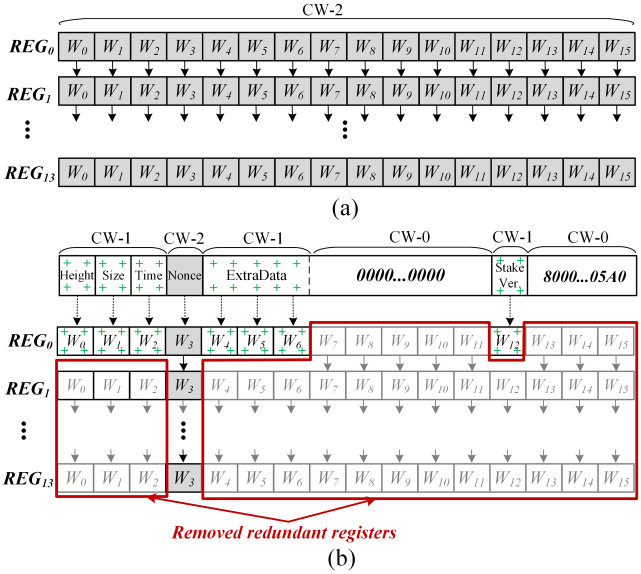

FIGURE 4. (a) Normal and (b) proposed register structures of the message permutation part of the fully pipelined BLAKE-256 accelerator for Decred mining.

computation tasks, the registers for storing these message words are unnecessary and can be eliminated. Moreover, the registers storing message words in the CW-1 group for the first round will store the same values as the registers storing those words for the remaining rounds. As a result, we can remove registers for storing message words in the CW-1 group in the remaining rounds. For example, we illustrate the normal and proposed register structures of the message permutation part of the fully pipelined BLAKE-256 accelerator for Decred mining in Fig. 4. As shown in Fig. 4 (a), each pipeline stage in the normal register structure uses sixteen 32-bit registers to store the sixteen words, meaning that 224 32-bit registers are needed for 14 pipeline stages. However, as shown in Fig. 4 (b), message words in the CW-0 group, including W_7, ..., W_{11} and W_{13}, ..., W_{15}, are constants in all mining tasks, and consequently, the corresponding registers can be removed. In addition, the registers for storing message words in the CW-1 group, including W_0, W_1, W_2, W_4, W_5, W_6, and W_{12}, for the last 13 rounds can be eliminated because the values stored in these registers will be the same as those stored in the registers for the first round during the same mining task. As a result, each pipeline stage in the proposed register structure stores only the necessary words, and the proposed message permutation scheme uses only 24 32-bit registers in 14 pipeline stages.

Idea 2: XOR optimization. In the message permutation process, each pipeline stage has a permutation circuit (denoted by *permutation r*, $0 \leq r \leq R\text{-}1$), which performs sixteen 32/64-bit XOR computations between sixteen 32/64-bit message words (W_i, $0 \leq i \leq 15$) and sixteen 32/64-bit constants. Since message words in the CW-0 group always remain unchanged, the results of the XOR calculations between these message words and the constants are predictable and can

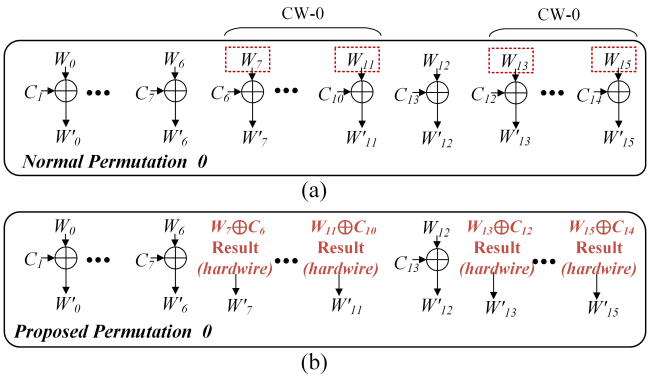

FIGURE 5. (a) Normal and (b) proposed permutation 0 circuits of the message permutation part of the fully pipelined BLAKE-256 accelerator for Decred mining.

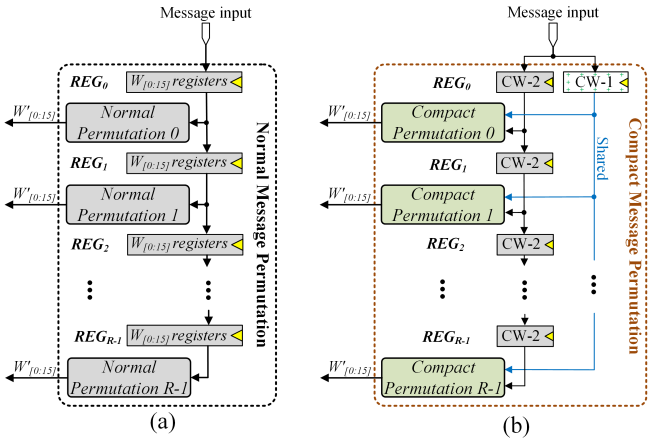

FIGURE 6. (a) Normal message permutation scheme vs. (b) proposed compact message permutation scheme.

be hardwired into the circuit to save XOR resources. For example, we illustrate the normal and proposed *permutation* 0 circuits of the message permutation part of the fully pipelined BLAKE-256 accelerator for Decred mining in Fig. 5. As shown in Fig. 5 (a), the normal *permutation* 0 circuit performs sixteen 32-bit XOR calculations between sixteen words and sixteen 32-bit constants; accordingly, the fourteen *permutation* circuits in the normal message permutation scheme require a total of 224 32-bit XOR gates. In contrast, as shown in Fig. 5 (b), the 32-bit XOR computations between message words in the CW-0 group (including W_7, ..., W_{11} and W_{13}, ..., W_{15}) and constants are replaced by hardwired values in the proposed circuit. As a result, the proposed *permutation* 0 circuit needs to perform only eight 32-bit XOR calculations, and the fourteen *permutation* circuits of the proposed message permutation scheme require only 112 32-bit XOR gates.

III. PROPOSED COMPACT MESSAGE PERMUTATION SCHEME

This section presents the proposed compact message permutation scheme for the fully pipelined BLAKE-256/512 accelerator for use in generic applications and blockchain mining.

A. OVERVIEW OF THE ARCHITECTURE

Fig. 6 shows the normal and proposed compact message permutation schemes for the fully pipelined BLAKE-256/512 accelerator. The normal and compact message permutation architectures each contain R pipeline stages (R is given in Table 1), and each pipeline stage in both architectures also returns the same sixteen permuted message words (denoted by W'_i, 0≤i≤15). Although the two architectures have the same functionality, the proposed message permutation scheme is lower in cost and smaller in area than the normal scheme. Fig. 6 (a) illustrates the overall architecture of the normal message permutation scheme, in which there are R permutation circuit blocks (denoted by *normal permutation* r, 0≤r≤R-1); each *normal permutation* circuit contains sixteen 32/64-bit XORs, and the register structure consumes R groups of sixteen 32/64-bit registers (denoted by REG_r, 0≤r≤R-1) to store the sixteen message words. With this register and permutation circuit structure, the normal message permutation architecture is suitable for message inputs consisting of sixteen message words, all in the CW-2 group. However, the normal message permutation scheme requires many registers and XORs. When the message inputs contain message words in the CW-1 or CW-0 group, some registers and XORs in the normal message permutation architecture will become unnecessary and redundant. Accordingly, we propose a compact message permutation scheme that utilizes the minimal numbers of registers and XORs to store and compute the necessary message words, as shown in Fig. 6 (b). Concretely, the register structure of the compact message permutation scheme is optimized to utilize R groups of 32/64-bit registers to store only message words in the CW-2 group, while only one cluster of 32/64-bit registers is used to store message words in the CW-1 group. Details of the register structure optimization are covered in Section III-B. In addition, the compact message permutation scheme requires R compact permutation circuit blocks (denoted by *compact permutation* r, 0≤r≤R-1). Each *compact permutation* circuit contains only enough 32/64-bit XORs to perform computations on message words in the CW-2 and CW-1 groups, whereas the 32/64-bit XORs for computations on message words in the CW-0 group are removed. The details of the XOR optimization for compact permutations are presented in Section III-C.

B. REGISTER OPTIMIZATION

This section analyzes the theory of register optimization and the register optimization coefficient. In addition, the hardware architecture of the optimized register structure in the compact message permutation scheme for generic applications and blockchain mining is presented.

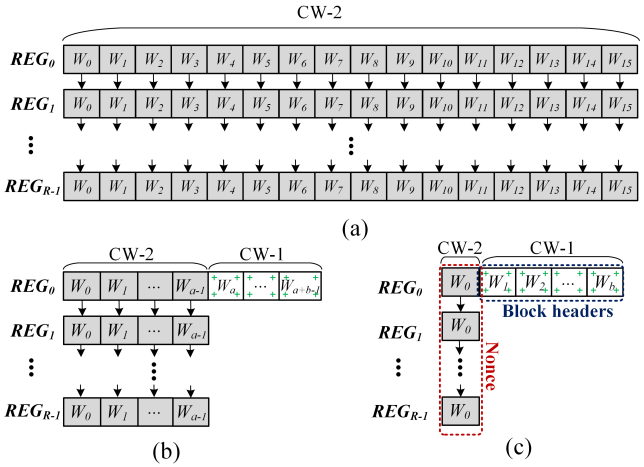

FIGURE 7. Register structures of (a) the normal message permutation scheme vs. (b-c) the proposed compact message permutation scheme for (b) generic applications and (c) blockchain mining.

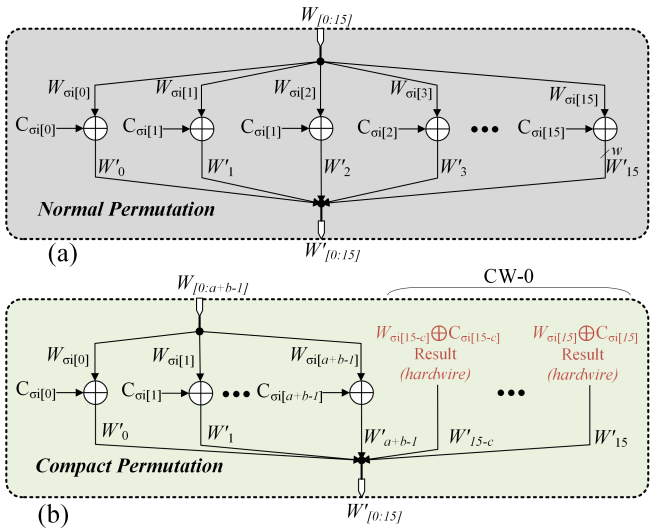

FIGURE 8. (a) Normal permutation circuit in the normal message permutation scheme and (b) compact permutation circuit in the compact message permutation scheme.

1) Basic theory of optimization

In the normal message permutation scheme, sixteen registers are used to store sixteen words (denoted by W_i, $0 \leq i \leq 15$) in each pipeline stage. The total number of registers ($\text{REG}_{\text{Total}}$) for R pipeline stages in the normal message permutation scheme is calculated as shown in eq. (1).

$$\text{REG}_{\text{Total}} = R \times \sum_{i=0}^{15} W_i \tag{1}$$

In the proposed message permutation scheme, the positions of message words in the CW-1 group are represented by a vector $P_{\text{CW-1}}$, where $P_{\text{CW-1}}$ is $[P_0,P_1,...,P_{15}]^T$. For example, P_t will equal 1 if W_t belongs to the CW-1 group and 0 otherwise. The number of optimizable registers ($\text{OP-REG}_{\text{CW-1}}$) due to message words belonging to the CW-1 group is calculated as shown in eq. (2).

$$\text{OP-REG}_{\text{CW-1}} = (R-1) \times [W_0, W_1, .., W_{15}] \times P_{\text{CW-1}} \tag{2}$$

Additionally, the positions of message words in the CW-0 group are represented by a vector $Q_{\text{CW-0}}$, where $Q_{\text{CW-0}}$ is $[Q_0,Q_1,...,Q_{15}]^T$. For example, Q_t will equal 1 if W_t belongs to the CW-0 group and 0 otherwise. The number of optimizable registers ($\text{OP-REG}_{\text{CW-0}}$) due to message words belonging to the CW-0 group is calculated as shown in eq. (3).

$$\text{OP-REG}_{\text{CW-0}} = R \times [W_0, W_1, .., W_{15}] \times Q_{\text{CW-0}} \tag{3}$$

The total number of optimizable registers due to message words belonging to the CW-1 and CW-0 groups ($\text{OP-REG}_{\text{CW-1|0}}$) is calculated as shown in eq. (4).

$$\begin{aligned}\text{OP-REG}_{\text{CW-1|0}} &= \text{OP-REG}_{\text{CW-0}} + \text{OP-REG}_{\text{CW-1}} \\ &= R \times [W_0, W_1, .., W_{15}] \times (P_{\text{CW-1}} + Q_{\text{CW-0}}) \\ &\quad - [W_0, W_1, .., W_{15}] \times P_{\text{CW-1}}\end{aligned} \tag{4}$$

Overall, the register optimization coefficient (OC_{REG}) of the proposed message permutation scheme compared to the normal message permutation scheme is calculated as shown in eq. (5).

$$\text{OC}_{\text{REG}} = \frac{\text{OP-REG}_{\text{CW-1|0}}}{\text{REG}_{\text{Total}}} \tag{5}$$

2) Optimized register structure

Fig. 7 shows the register structures in the normal and compact message permutation architectures for generic applications and blockchain mining. Specifically, Fig. 7 (a) shows the register structure in the normal message permutation scheme, in which sixteen registers for storing sixteen message words are linearly propagated through R rounds. Since the register structure includes the full sixteen registers in every pipeline stage, the normal message permutation scheme is most suitable for message inputs consisting of sixteen message words, all in the CW-2 group. However, when the message inputs contain message words in the CW-1 or CW-0 group, many registers in the register structure will have unchanged values and become redundant. Therefore, we propose a new register structure to reduce redundant registers when message words in the CW-1 or CW-0 group are introduced into the message inputs. Accordingly, Fig. 7 (b) and Fig. 7 (c) show our proposed register structures for compact message permutation for generic applications and blockchain mining, respectively. We denote the numbers of message words in the CW-2, CW-1, and CW-0 groups by a, b, and c, respectively. a, b and c are calculated as shown in eq. (6), eq. (7), and eq. (8), respectively.

$$a = 16 - b - c \tag{6}$$

$$b = Sum(P_{\text{CW-1}}) = \sum_{i=0}^{15} P_i \tag{7}$$

$$c = Sum(Q_{\text{CW-0}}) = \sum_{i=0}^{15} Q_i \tag{8}$$

In the compact message permutation scheme, the redundant registers for storing the c message words in the CW-0 group are removed. In addition, the registers for storing the b message words in the CW-1 group are placed only in the first-round stage and are shared with the stages for other rounds. Finally, each pipeline stage includes registers for storing the a message words in the CW-2 group. Despite the significant pruning of the registers, the register structure of the compact message permutation architecture still ensures the storage of sufficient necessary message words for correct functionality, yielding the same results as the normal message permutation scheme. In generic applications, the values of a, b, and c are arbitrary and depend on the user's purpose, as shown in Fig. 7 (b). In blockchain mining, the value of a is usually one because only the message word of the *nonce* field belongs to the CW-2 group, whereas the values of b and c are arbitrary, as shown in Fig. 7 (c).

C. XOR OPTIMIZATION

This section analyzes the theory of XOR optimization and the XOR optimization coefficient. Moreover, the hardware architecture of the optimized permutation circuit is presented.

1) Basic theory of optimization

The XOR operation between message word W_i and the corresponding constant is denoted by $X(W_i)$, where $0 \leq i \leq 15$. In the normal message permutation scheme, the total number of XOR gates ($\text{XOR}_{\text{Total}}$) required for R pipeline stages is calculated as shown in eq. (9).

$$\text{XOR}_{\text{Total}} = R \times \sum_{i=0}^{15} X(W_i) \tag{9}$$

In the compact message permutation scheme, the results of the XOR calculations between message words in the CW-0 group and constants are predictable. Therefore, we replace the XOR computations between these message words and constants with hardwired values to reduce the utilization of XOR resources. The number of optimizable XOR gates ($\text{OP-XOR}_{\text{CW-0}}$) due to message words belonging to the CW-0 group is calculated as shown in eq. (10).

$$\text{OP-XOR}_{\text{CW-0}} = R \times [X(W_0), X(W_1), .., X(W_{15})] \times Q_{\text{CW-0}} \tag{10}$$

Note that the positions of message words in the CW-0 group are represented by the vector Q_{ctw}, where Q_{ctw} is $[Q_0,Q_1,...,Q_{15}]^T$.

Overall, the XOR optimization coefficient (OC_{XOR}) of the compact message permutation scheme compared to the normal message permutation scheme is calculated as shown in eq. (11).

$$\text{OC}_{\text{XOR}} = \frac{\text{OP-REG}_{\text{CW-0}}}{\text{XOR}_{\text{Total}}} \tag{11}$$

2) Hardware Architecture

Fig. 8 shows the permutation circuit architectures in the normal and proposed message permutation schemes. Specifically, Fig. 8 (a) illustrates the normal permutation circuit for performing sixteen XOR computations between sixteen message words and sixteen constants. With the full sixteen XOR gates in each pipeline stage, the normal permutation circuit is best suited for message inputs consisting of sixteen message words all belonging to the CW-2 or CW-1 group. On the other hand, when the message inputs have the same length, message words in the CW-0 group are introduced by the padding values. Accordingly, a compact permutation circuit is proposed to reduce the number of XOR gates needed for computations on message words in the CW-0 group, as shown in Fig. 8 (b). Specifically, the results of the XOR calculations between message words in the CW-0 group and constants are predictable in every pipeline stage. Therefore, in the compact permutation circuit for each pipeline stage, the XOR calculations between message words in the CW-0 group and constants are eliminated and replaced by hardwired values to reduce XOR resource utilization.

IV. VERIFICATION AND EVALUATION

This section presents the verification and evaluation of the proposed compact message permutation scheme for the fully pipelined BLAKE-256/512 accelerator. Throughout this section, for concise differentiation, the fully pipelined BLAKE-256/512 accelerator with normal message permutation is referred to as the normal BLAKE-256/512 accelerator, and the fully pipelined BLAKE-256/512 accelerator with compact message permutation is referred to as the proposed BLAKE-256/512 accelerator.

A. FPGA-BASED VERIFICATION OF COMPACT MESSAGE PERMUTATION

This section presents the implementation and verification of the proposed fully pipelined BLAKE-256/512 accelerator on a Xilinx Alveo U280 FPGA at the system-on-chip (SoC) level, as shown in Fig. 9. The experimental equipment consists of two main devices: the Alveo FPGA and a host PC with an Intel Xeon E5-2620v2 CPU @2.10 GHz with 94 GB of RAM. The Alveo FPGA and host PC exchange data via Joint Test Action Group (JTAG) and Universal Asynchronous Receiver/Transmitter (UART) connectors. The Alveo FPGA contains the following cores: a clock generator, a MicroBlaze Processor (MP), a test framework IP, and a ChipScope Integrated Logic Analyzer (ChipScope ILA). Concretely, the clock generator provides a 100 MHz operating frequency for all other cores. The MP sends messages and hash inputs from the host PC to the test framework IP. The test framework IP consists of two accelerators: the normal and proposed

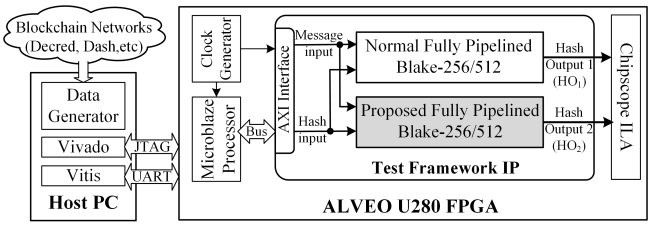

FIGURE 9. Verification of the proposed fully pipelined BLAKE-256/512 accelerator with compact message permutation on a Xilinx Alveo U280 FPGA at the SoC level.

BLAKE-256/512 accelerators. The ChipScope ILA is a customizable logic analyzer core for monitoring the hash outputs of the normal (denoted by HO_1) and proposed (denoted by HO_2) BLAKE-256/512 accelerators. The compact message permutation scheme in the proposed BLAKE-256/512 accelerator will be determined to be working properly if HO_1 and HO_2 are the same. On the other hand, the host PC executes Vivado, Vitis, and a "Data Generator" C program. Specifically, the Xilinx Vitis tool runs an embedded C program to transfer message and hash inputs to the test framework IP via the MP. In addition, the Vivado tool loads the SoC-based design into the Alveo FPGA. In this experiment, we use Vivado and Vitis version 2019.2. Furthermore, the "Data Generator" generates message and hash inputs for the two accelerators.

We implemented and verified the proposed BLAKE-256/512 accelerator for two types of applications: generic applications and blockchain mining. For generic applications, message inputs with a given number of CW-1 or CW-0 words were randomly generated by the "Data Generator" program. For blockchain mining, message inputs were extracted from the block headers of two blockchain networks, namely, the Decred network for BLAKE-256r14 verification and the Dash network for BLAKE-512r16 verification. Based on a certain number of CW-1 or CW-0 words, the compact message permutation architecture for the proposed BLAKE-256/512 accelerator was designed to reduce the hardware resource utilization in terms of XORs and registers. For 100,000 different message inputs, all HO_1 and HO_2 values were the same. This demonstrates that the proposed BLAKE-256/512 accelerator works properly for both generic applications and blockchain mining.

B. THEORETICAL EVALUATION

To prove the effectiveness of the compact message permutation scheme, this section theoretically evaluates the register and XOR optimization coefficients (OC_{REG} and OC_{XOR}) based on different numbers of message words belonging to the CW-1 or CW-0 group (denoted by CW-1|0 words). Since the number of CW-1|0 words is highly dependent on the particular application, we will examine all possible cases of numbers of CW-1|0 words ranging from zero to fifteen.

Table 4 presents the OC_{REG} and OC_{XOR} results for five BLAKE-256/512 functions, namely, BLAKE-256r8, BLAKE-256r10, BLAKE-256r14, BLAKE-512r14, and BLAKE-512r16, based on numbers of CW-1|0 words ranging from zero to fifteen. Note that OC_{REG} and OC_{XOR} are calculated using eq. (5) and eq. (11), respectively. Since eq. (11) is affected only by the number of CW-0 words (considered equal to the number of CW-1|0 words in Table 4) but not the number of rounds (R), the OC_{XOR} results are the same for all five BLAKE-256/512 functions with different numbers of rounds. Thus, the results for OC_{XOR} in Table 4 represent all five BLAKE-256/512 functions.

TABLE 4. Register and XOR gate optimization coefficients (OC_{REG} and OC_{XOR}) based on different numbers of PUCWs.

CW-1|0 words	OC_{REG} (%)					OC_{XOR} (%)
	BLAKE-256r8	BLAKE-256r10	BLAKE-256r14	BLAKE-512r14	BLAKE-512r16	BLAKE-256/512
0	0	0	0	0	0	0
1	5.4	5.6	5.8	6.8	5.9	6.3
2	10.7	11.3	11.6	11.6	11.7	12.5
3	16.1	16.9	17.4	17.4	17.6	18.8
4	21.4	22.5	23.2	23.2	23.4	25.0
5	26.8	28.1	29.0	29.0	29.3	31.3
6	32.1	33.8	34.8	34.8	35.2	37.5
7	37.5	39.4	40.6	40.6	41.0	43.8
8	42.9	45.0	46.4	46.4	46.9	50.0
9	48.2	50.6	52.2	52.2	52.7	56.3
10	53.6	56.3	58.0	58.0	58.6	62.5
11	58.9	61.9	63.8	63.8	64.5	68.8
12	64.3	67.5	69.6	69.6	70.3	75.0
13	69.6	73.1	75.4	75.4	76.2	81.3
14	75.0	78.8	81.3	81.3	82.0	87.5
15	80.4	84.4	87.1	87.1	87.9	93.8

The OC_{REG} results for the five BLAKE-256/512 functions improve linearly as the number of CW-1|0 words increases. At fifteen message words in the CW-1 or CW-0 group, the OC_{REG} results for the five BLAKE-256/512 functions peak at greater than **80%** optimization. Specifically, among the five BLAKE-256/512 functions, BLAKE-512r16 has the highest OC_{REG} with **80.4%** optimization at fifteen message words in the CW-1 or CW-0 group, and BLAKE-512r16 has the highest OC_{REG} with **87.9%** optimization.

The OC_{XOR} results for the five BLAKE-256/512 functions show linear improvement with an increasing number of CW-1|0 words, reaching a peak of **93.8%** at fifteen message words in the CW-0 group.

In general, the OC_{REG} and OC_{XOR} values for the five BLAKE-256/512 functions increase linearly as the number of CW-1|0 words increases. At fifteen message words in the CW-1 or CW-0 group, as is usually encountered in blockchain mining, the fully pipelined BLAKE-256/512 accelerator has the highest register and XOR optimization coefficients.

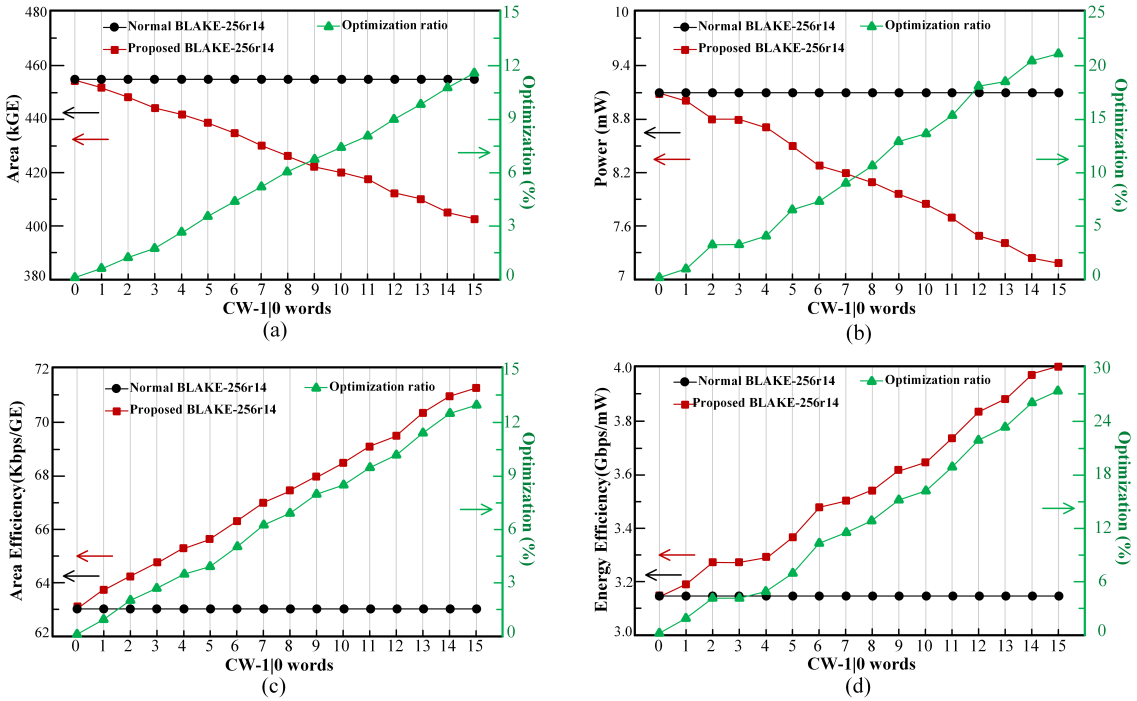

FIGURE 10. Normal vs. proposed fully pipelined BLAKE-256r14 accelerators in terms of (a) area, (b) power consumption, (c) area efficiency, (d) and energy efficiency.

C. EXPERIMENTAL EVALUATION

Since the fully pipelined BLAKE-256/512 accelerator concept is often oriented toward ASIC fabrication to maximize performance and power consumption, this section proves the effectiveness of the proposed compact message permutation scheme based on an ASIC implementation. For this experiment, the BLAKE-256r14 and BLAKE-512r16 functions are selected for ASIC implementation because they are the most commonly used BLAKE-256/512 functions at present. The factors considered for evaluation include area, power, area efficiency, and energy efficiency. In our experiment, we used "Design Compiler version N-2017.09-SP1" and "IC Compiler version Q-2019.12-SP4" for ASIC synthesis with the *Ptt_V0p75_T25* library for Renesas 65 nm silicon-on-thin-buried-oxide (SOTB) technology.

1) BLAKE-256r14

The ASIC synthesis results show that the normal and proposed BLAKE-256r14 accelerators both provide a throughput of 28.67 Gbps at 56 MHz. Moreover, based on the ASIC synthesis results, we present the area, power consumption, area efficiency, and energy efficiency of the normal and proposed BLAKE-512r14 accelerators for different numbers of CW-1|0 words, as shown in Fig. 10. Because the normal message permutation scheme is fixed for any number of CW-1|0 words, the area, power, area efficiency, and energy efficiency of the normal BLAKE-256r14 accelerator remain at constant values of 454 kGE (thousand gate equivalent), 9.1 mW, 63.02 kbps/GE, and 3.15 Gbps/mW, respectively. Meanwhile, the compact message permutation scheme of the proposed BLAKE-256r14 accelerator specifies how to develop an architecture that is suitable for processing message inputs with each specific number of CW-1|0 words so as to greatly reduce the necessary numbers of registers and XORs. Therefore, with an increasing number of CW-1|0 words, the area and power consumption of the proposed BLAKE-256r14 accelerator are significantly reduced, as shown in Fig. 10 (a) and (b), respectively. In particular, the area and power consumption of the proposed BLAKE-256r14 accelerator are optimized by **11.35%** (403 vs. 454 kGE) and **21.10%** (7.2 vs. 9.1 mW), respectively, compared to the normal BLAKE-256r14 accelerator at fifteen message words in the CW-1 or CW-0 group. Since the area and power consumption of the proposed BLAKE-256r14 accelerator are greatly reduced while the throughput remains unchanged, the area efficiency and energy efficiency are significantly increased, as shown in Fig. 10 (c) and (d), respectively. In particular, the area efficiency and energy efficiency of the proposed BLAKE-256r14 accelerator are remarkably improved by **13.12%** (71.29 vs. 63.02 kbps/GE) and **27.09%** (4.00 vs. 3.15 Gbps/mW), respectively, compared to the normal BLAKE-256r14 accelerator at fifteen message words in the CW-1 or CW-0 group.

2) BLAKE-512r16

The ASIC synthesis results show that the normal and proposed BLAKE-512r16 accelerators both deliver a throughput of 50.54 Gbps at 49 MHz. In addition, based on the ASIC

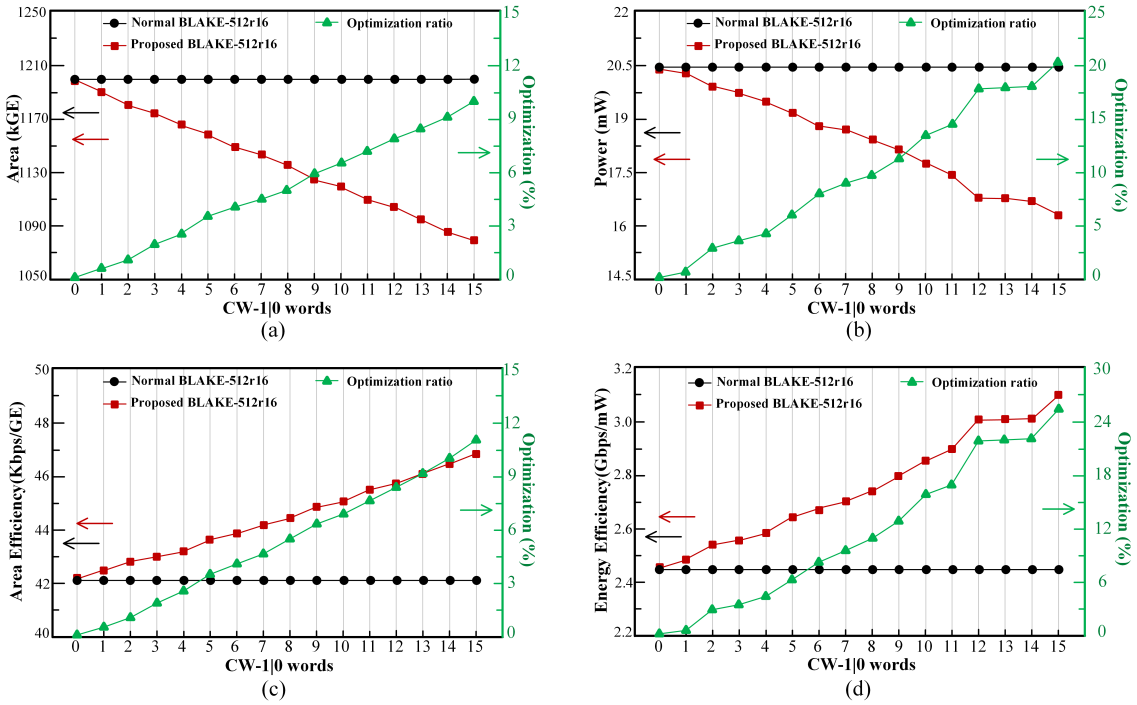

FIGURE 11. Normal vs. proposed fully pipelined BLAKE-512r16 accelerators in terms of (a) area, (b) power consumption, (c) area efficiency, (d) and energy efficiency.

synthesis results, Fig. 11 shows the area, power consumption, area efficiency, and energy efficiency of the normal and proposed BLAKE-512r14 accelerators for different numbers of CW-1|0 words. Since the normal message permutation scheme is fixed for any number of CW-1|0 words, the area, power consumption, area efficiency, and energy efficiency of the normal BLAKE-512r16 accelerator remain constant at 1,198 kGE, 20.5 mW, 42.18 kbps/GE, and 2.47 Gbps/mW, respectively. In contrast, the compact message permutation scheme of the proposed BLAKE-512r16 accelerator allows a suitable architecture to be designed for processing message inputs with any specific number of CW-1|0 words so as to greatly reduce the necessary numbers of registers and XORs. It is evident that with an increasing number of CW-1|0 words, the area and power consumption of the proposed BLAKE-512r16 accelerator are markedly reduced, as shown in Fig. 11 (a) and (b), respectively. Concretely, the area and power consumption of the proposed BLAKE-512r16 accelerator are optimized by **9.86%** (1,080 vs. 1,198 kGE) and **20.32%** (16.3 vs. 20.5 mW), respectively, compared to the normal BLAKE-512r16 accelerator at fifteen message words in the CW-1 or CW-0 group. In addition, because of the reductions in area and power consumption, the area efficiency and energy efficiency of the proposed BLAKE-512r16 accelerator are significantly improved, as shown in Fig. 11 (c) and (d), respectively. Specifically, the area efficiency and energy efficiency of the proposed BLAKE-512r16 accelerator are improved by **10.9%** (46.80 vs. 42.18 kbps/GE) and **25.50%** (3.10 vs. 2.47 Gbps/mW), respectively, compared to the normal BLAKE-512r16 accelerator at fifteen message words in the CW-1 or CW-0 group.

Overall, with an increasing number of CW-1|0 words, the proposed BLAKE-256r14/BLAKE-512r16 accelerator is significantly superior to the normal BLAKE-256r14/BLAKE-512r16 accelerator in terms of area, power consumption, area efficiency, and energy efficiency. This shows that the compact message permutation scheme helps considerably optimize the area, power consumption, area efficiency, and energy efficiency of a fully pipelined BLAKE-256/512 accelerator in an ASIC implementation.

D. PERFORMANCE EVALUATION: OUR PROPOSAL VS. OTHER FPGA-BASED WORKS

This section presents a performance comparison between the proposed BLAKE-256/512 accelerator and other FPGA-based BLAKE-256/512 designs. Because the characteristics of message words in the CW-1 or CW-0 group are most clearly evident in the blockchain mining process, this evaluation is conducted based on blockchain mining applications. Accordingly, two blockchains are selected, Decred and Dash, whose mining processes use BLAKE-256r14 and BLAKE-512r16, respectively.

For fair comparison with existing BLAKE-256/512 designs such as [10]–[13], [16], [23], [24], we synthesized the proposed BLAKE-256r14 and BLAKE-512r16 accelerators on Xilinx Virtex-5 and Virtex-6 FPGA boards. We also synthesized corresponding normal BLAKE-256r14 and BLAKE-512r16 accelerators to clarify the effectiveness of

TABLE 5. Comparison between the proposed design and other FPGA-based works.

FPGA device	Algorithm	Reference	Frequency (MHz)	#Cycles/Hash	Slices	Throughput (Mbps)	Area efficiency (Mbps/slice)
Virtex-5	BLAKE-256r14	[12], FPT 2010	372	1,164*	56	163.6	2.92
		[16], FLP 2010	118	56*	1,118	1,078.9	0.96
		[24], SSD 2015	79	16	691	2,528	3.66
		Normal fully pipelined	72	1	12,087	36,864	3.05
		Proposed fully pipelined	**72**	**1**	**9,621**	**36,864**	**3.83**
	BLAKE-512r16	[12], FPT 2010	358	1324**	108	276.9	2.56
		[16], FLP 2010	91	64**	1,718	1,456	0.85
		Normal fully pipelined	61	1	27,167	62,464	2.30
		Proposed fully pipelined	**61**	**1**	**22,542**	**62,464**	**2.77**
Virtex-6	BLAKE-256r14	[10], CARDIS 2011	274	1,182	117	118.7	1.01
		[11], TSCC 2012	268	308	166	445.5	2.68
		[13], TCAS-I 2014	349	1,184	50	150.9	3.02
		Normal fully pipelined	76	1	12,137	38,912	3.21
		Proposed fully pipelined	**76**	**1**	**9,839**	**38,912**	**3.95**
	BLAKE-512r16	[13], TCAS-I 2014	329	1,344	91	250.7	2.75
		[23], Trans Comput 2014	133	34	2,153	4,005.6	1.86
		Normal fully pipelined	67	1	27,904	68,608	2.46
		Proposed fully pipelined	**66**	**1**	**22,542**	**67,584**	**3.00**

* : BLAKE-256r10 is normalized to BLAKE256r14 by adding more 4 rounds.
** : BLAKE-512r14 is normalized to BLAKE-512r16 by adding more 2 rounds.

the proposed compact message permutation scheme compared to normal message permutation. The factors considered for comparison here include area, throughput, and area efficiency.

Throughput, measured in megabits per second (Mbps), is calculated using eq. (12), where BlockSize is equal to 512 for BLAKE-256r14 and 1024 for BLAKE-512r16s.

$$\text{Throughput} = \frac{\text{BlockSize} \times \text{Frequency}}{\#\text{Cycles/Hash}} \tag{12}$$

Then, the trade-off between throughput and area (referred to as area efficiency) is calculated as shown in eq. (13).

$$\text{Area efficiency} = \frac{\text{Throughput}}{\text{Area}} \tag{13}$$

Table 5 shows the area, throughput, and area efficiency of the proposed work and related works on the Virtex-5 and Virtex-6 FPGA boards. Note that the BLAKE-256/512 designs presented in [12], [16] are for the BLAKE-256r10 and BLAKE-512r14 functions, which are normalized to BLAKE-256r14 and BLAKE-512r16 for fair comparison with the other designs by adding 4 and 2 more rounds, respectively.

On the Virtex-5 FPGA board, the proposed BLAKE-256r14 accelerator occupies 9,621 slices and reaches 36,864 Mbps at a maximum frequency of 72 MHz. Accordingly, the throughput of the proposed BLAKE-256r14 accelerator is **225.3 times** (36,864 vs. 163.6), **34.2 times** (36,864 vs. 1,078.9), and **14.6 times** (36,864 vs. 1,078.9) higher than those of the designs in [12], [16], and [24], respectively. In addition, the proposed BLAKE-256r14 accelerator reaches an area efficiency of 3.83 Mbps/slice, which is **1.3 times** (3.83 vs. 2.92), **4 times** (3.83 vs. 0.96), **1.05 times** (3.83 vs. 3.66), and **1.3 times** (3.83 vs. 3.05) better than those of the designs in [12], [16], and [24] and the normal BLAKE-256r14 accelerator, respectively. In the synthesis results for BLAKE-512r16, the proposed BLAKE-512r16 accelerator utilizes 22,542 slices and delivers 62,464 Mbps at a maximum frequency of 61 MHz. The throughput of the proposed BLAKE-512r16 accelerator is **225.6 times** (62,464 vs. 276.9) and **42.9 times** (62,464 vs. 1,456) greater than those of the designs in [12] and [16], respectively. Moreover, the area efficiency of the proposed BLAKE-512r16 accelerator is 2.77 Mbps/slice, which is **1.08 times** (2.77 vs. 2.56), **3.3 times** (2.77 vs. 0.85), and **1.2 times** (2.77 vs. 2.3) greater than those of the architectures in [16] and [12] and the normal BLAKE-512r16 accelerator, respectively.

On the Virtex-6 FPGA board, the proposed BLAKE-256r14 accelerator utilizes 9,621 slices and provides 38,912 Mbps at a maximum frequency of 76 MHz. This throughput is **327.8 times** (38,912 vs. 118.7), **87.3 times** (38,912 vs. 445.5), and **257.9 times** (38,912 vs. 150.9) higher than those of the architectures in [10], [11], and [13], respectively. Furthermore, the proposed fully pipelined BLAKE-256r14 accelerator achieves an area efficiency of 3.95 Mbps/slice, which is **3.9 times** (3.95 vs. 1.01), **1.5 times** (3.95 vs. 2.68), **1.3 times** (3.95 vs. 3.02), and **1.2 times** (3.95 vs. 3.21) better than those of the designs in [10], [11], and [13] and the normal BLAKE-256r14 accelerator, respectively. In the synthesis results for BLAKE-512r16, the proposed BLAKE-

TABLE 6. Comparison between the proposed fully pipelined BLAKE-256r14 accelerator and state-of-the-art CPU and GPU platforms.

Platform		Technology	Power (W)		Frequency	Hash rate	Energy efficiency
			Measured	TDP	(MHz)	(Mhash/s)	(Mhash/s/W)
CPU: Intel i9-10940X		ASIC, 14 nm	150	165	3,300	18	0.1
GPU: GTX 1080 Ti, 11 GB		ASIC, 16 nm	227	280	1,481	592	2.6
GPU: Tesla V100, 16 GB		ASIC, 12 nm	120	250	1,245	972	8.1
GPU: RTX 3090, 24 GB		ASIC, 8 nm	337	350	1,395	1,370	4.1
FPGA: Alveo U280	Normal fully pipelined	FPGA, 16 nm	**1.23**	225	100	100	**81.3**
	Proposed fully pipelined		**1.02**		**100**	**100**	**98.0**

512r16 accelerator utilizes 22,542 slices and delivers 67,584 Mbps at a maximum frequency of 66 MHz. Accordingly, the throughput of the proposed BLAKE-512r16 accelerator is **22.5 times** (67,584 vs. 250.7) and **269.6 times** (67,584 vs. 4,005.6) higher than those of the designs in [23] and [13], respectively. Additionally, the area efficiency of the proposed BLAKE-512r16 accelerator is 3 Mbps/slice, which is **1.6 times** (3.0 vs. 2.75), **1.09 times** (3.0 vs. 1.86), and **1.2 times** (3.0 vs. 2.46) greater than those of designs in [23] and [13] and the normal BLAKE-512r16 accelerator, respectively.

Overall, the proposed BLAKE-256r14/512r16 accelerator is significantly superior to other related FPGA-based designs in both throughput and area efficiency. In addition, the area efficiency of the proposed BLAKE-256r14/512r16 accelerator is greatly improved compared to that of the normal fully pipelined BLAKE-256r14/512r16 accelerator. This indicates that the compact message permutation scheme dramatically improves the area efficiency for fully pipelined BLAKE-256r14/512r16 accelerators on FPGAs.

E. PERFORMANCE EVALUATION: OUR PROPOSAL VS. STATE-OF-THE-ART CPUS AND GPUS

Although FPGA-based designs can be used to implement BLAKE-2564r14/512r16 for blockchain mining, the poor throughput of related FPGA-based designs can make the mining process inefficient or infeasible. Therefore, the proposed fully pipelined BLAKE-256r14/512r16 accelerator should also be compared with high-performance platforms such as CPUs and GPUs, which are commonly used for blockchain mining. Accordingly, this section compares the proposed BLAKE-256r14/512r16 accelerator with the most powerful CPU and GPUs currently used in blockchain mining, such as the Intel i9-10940X CPU, the GTX 1080 Ti GPU, the RTX 3090 GPU, and the Tesla V100 GPU.

Currently, only BLAKE-256r14 is used as an independent function for blockchain mining in several cryptocurrencies, e.g., Decred and HyperCash, while BLAKE-512r16 is often used as one of multiple hash functions in a hashing sequence, e.g., X11 in Dash mining. To clarify the effectiveness of the compact message permutation scheme, we evaluate only the proposed BLAKE-2564r14 accelerator with CPUs and GPUs used in blockchain mining, especially in Decred mining. Specifically, the proposed BLAKE-256r14 accelerator is implemented at the SoC level on the Alveo U280 FPGA board. It occupies 21,759 look-up tables (LUTs) and 8,910 flip-flops (FFs), delivers a throughput of 100 Mhash/s (megahashes per second) at a 100 MHz operating frequency, and consumes 1.02 W. Furthermore, we have also implemented the normal BLAKE-256r14 accelerator at the SoC level for comparison with the proposed BLAKE-256r14 accelerator. The normal BLAKE-256r14 accelerator utilizes 22,673 LUTs and 15,919 FFs, produces a throughput of 100 Mhash/s at a 100 MHz operating frequency, and consumes 1.23 W. Note that the normal and proposed BLAKE-256r14 accelerators both have a single core and utilize only approximately 2% of the FPGA resources. Theoretically, we could expand both the normal and proposed BLAKE-256r14 accelerators to 46 cores to achieve a hash rate of 4,600 Mhash/s. However, the present evaluation focuses only on the energy efficiency of the single-core versions of the normal and proposed BLAKE-256r14 accelerators. Meanwhile, to achieve the maximum performance of the CPU and GPUs, we use the *cpuminer* and *ccminer* open-source mining software tools to execute the BLAKE-256r14 computation.

Table 6 presents the power consumption, hash rate, and energy efficiency results for the proposed BLAKE-256r14 accelerator and the CPU/GPUs. Concretely, the power consumption of the proposed BLAKE-256r14 accelerator is significantly lower than that of the CPU and GPUs. Notably, GPUs offer better performance than either the proposed BLAKE-256r14 accelerator or the CPU. For example, the fastest GPU device, the RTX 3090, is **13.7 times** (1,370 vs. 100) faster than the proposed BLAKE-256r14 accelerator. However, thanks to its exceptionally low power consumption, the proposed BLAKE-256r14 accelerator achieves significantly better energy efficiency than any CPU or GPU. Specifically, the energy efficiency of the proposed BLAKE-256r14 accelerator is **980 times** (98.0 vs 0.1), **37.5 times** (98.0 vs 2.6), **23.9 times** (98.0 vs 4.1), and **12.1 times** (98.0 vs 8.1) higher than those of the i9 CPU, the GTX 1080 GPU, the RTX 3090 GPU, and the Tesla V100 GPU, respectively. Moreover, the energy efficiency of the proposed BLAKE-256r14 accelerator is **1.2 times** (98.0 vs 81.3) higher than that of the normal BLAKE-256r14 accelerator, which shows that the compact message permutation scheme significantly im-

proves the energy efficiency of the fully pipelined BLAKE-256r14 accelerator.

V. CONCLUSION

The development of a low-power and high-performance BLAKE accelerator has recently received extensive interest because the BLAKE algorithm is widely applied in many applications, ranging from the Internet of Things (IoT) to cryptocurrency. However, the performance of existing BLAKE-256/512 hardware is often low, making such devices difficult to apply for high-speed applications such as blockchain mining. Therefore, we have introduced the first fully pipelined BLAKE-256/512 accelerator to simultaneously achieve high performance and hardware efficiency. In addition, based on the word change rates in consecutive message inputs, we have proposed a compact message permutation scheme that incorporates two new optimization techniques to reduce the numbers of registers and XOR gates needed in a fully pipelined BLAKE-256/512 accelerator. An ASIC-based experiment shows that this compact message permutation scheme helps significantly reduce the area and power consumption of a fully pipelined BLAKE-256/512 accelerator. We have verified the performance of the fully pipelined BLAKE-256/512 accelerator with compact message permutation on a real hardware platform (an Alveo U280 FPGA). When applied for blockchain mining, the fully pipelined BLAKE-256r14 accelerator with compact message permutation implemented on an Alveo U280 FPGA achieves improvements in energy efficiency by factors of 980 and 23.9 compared with the fastest current CPU (the Intel i9-10940X) and GPU (the RTX 3090), respectively, that are used for this application. Moreover, experiments on several Xilinx FPGA boards prove that the proposed fully pipelined BLAKE-256/512 accelerator with compact message permutation is significantly superior to related FPGA-based works in both throughput and area efficiency.

Despite its advantages in performance and hardware efficiency, the fully pipelined BLAKE-256/512 accelerator still lacks the flexibility to be configured for computing many BLAKE functions. In our future research, we will develop a BLAKE accelerator with high performance and flexibility that can support the hash functions of new BLAKE generations, such as BLAKE2 and BLAKE3.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] J. Zhai, C. M. Park, and G.-N. Wang, "Hash-based rfid security protocol using randomly key-changed identification procedure," in Computational Science and Its Applications - ICCSA 2006, M. L. Gavrilova, O. Gervasi, V. Kumar, C. J. K. Tan, D. Taniar, A. Laganá, Y. Mun, and H. Choo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 296–305.

[2] Y. Zhang, P. Wang, X. Zhang, X. Weng, and Z. Yu, "A pufs-based hardware authentication blake algorithm in 65 nm cmos," International Journal of Electronics, vol. 103, no. 6, pp. 1056–1066, 2016.

[3] 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.

[4] F. Fernandes, R. Gupta, S. Sivanantham, and K. Sivasankaran, "Implementation of blake 256 hash function for password encryption and parallel crc," in 2015 Online International Conference on Green Engineering and Technologies (IC-GET), 2015, pp. 1–4.

[5] 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.

[6] 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, A. Lopata, D. Gudonienė, and R. Butkienė, Eds. Cham: Springer International Publishing, 2021, pp. 185–193.

[7] Decred-secure. adaptable. sustainable. Accessed: Dec. 22, 2021. [Online]. Available: https://www.decred.org

[8] H. Cho, "Asic-resistance of multi-hash proof-of-work mechanisms for blockchain consensus protocols," IEEE Access, vol. 6, pp. 66 210–66 222, 2018.

[9] J. Li and R. Karri, "Compact hardware architectures for blake and lake hash functions," in Proceedings of 2010 IEEE International Symposium on Circuits and Systems. IEEE, 2010, pp. 2107–2110.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Y. Zhang, J. Han, X. Weng, Z. He, and X. Zeng, "Design approach and implementation of application specific instruction set processor for sha-3 blake algorithm," IEICE transactions on electronics, vol. 95, no. 8, pp. 1415–1426, 2012.

[15] V. F. Pereira, E. D. Moreno, W. R. A. Dias, and D. O. D. dos Santos, "Specific processor in fpga for blake algorithm," in 2013 IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS), 2013, pp. 1–5.

[16] 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.

[17] B. Jungk and J. Apfelbeck, "Area-efficient fpga implementations of the sha-3 finalists," in 2011 International Conference on Reconfigurable Computing and FPGAs, 2011, pp. 235–241.

[18] L. Henzen, J.-P. Aumasson, W. Meier, and R. C.-W. Phan, "Vlsi characterization of the cryptographic hash function blake," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 10, pp. 1746–1754, 2011.

[19] 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.

[20] Z. Liu, X. Dong, Y. Zhao, and D. Li, "Hardware implementation of sha-3 candidate based on blake-32," in 2012 5th International Conference on BioMedical Engineering and Informatics, 2012, pp. 1317–1320.

[21] F. Kahri, B. Bouallegue, M. Machhout, and R. Tourki, "An fpga implementation of the sha-3: The blake hash function," in 10th International Multi-Conferences on Systems, Signals Devices 2013 (SSD13), 2013, pp. 1–5.

[22] M. Srivastav, X. Guo, S. Huang, D. Ganta, M. B. Henry, L. Nazhandali, and P. Schaumont, "Design and benchmarking of an asic with five sha-3

finalist candidates," Microprocessors and Microsystems, vol. 37, no. 2, pp. 246–257, 2013.

[23] S. Ghosh and I. Verbauwhede, "Blake-512-based 128-bit cca2 secure timing attack resistant mceliece cryptoprocessor," IEEE Transactions on Computers, vol. 63, no. 5, pp. 1124–1133, 2014.

[24] F. Kahri, H. Mestiri, B. Bouallegue, and M. Machhout, "Efficient fpga hardware implementation of secure hash function sha-256/blake-256," in 2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15), 2015, pp. 1–5.

[25] J. Sugier, "Implementation efficiency of BLAKE and other contemporary hash algorithms in popular FPGA devices," in Dependability Engineering and Complex Systems. Springer International Publishing, 2016, pp. 457–467.

[26] ——, "Spartan fpga devices in implementations of aes, blake and k eccak cryptographic functions," in International Conference on Dependability and Complex Systems. Springer, 2018, pp. 461–470.

[27] M. M. Kermani, S. Bayat-Sarmadi, A.-B. Ackie, and R. Azarderakhsh, "High-performance fault diagnosis schemes for efficient hash algorithm blake," in 2019 IEEE 10th Latin American Symposium on Circuits Systems (LASCAS), 2019, pp. 201–204.

[28] Block header specifications. Accessed: Dec. 22, 2021. [Online]. Available: https://devdocs.decred.org/developer-guides/block-header-specifications/

[29] E. Biham and O. Dunkelman, "A framework for iterative hash functions—haifa," Computer Science Department, Technion, Tech. Rep., 2007.

[30] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "Sha-3 proposal blake," Submission to NIST, vol. 92, 2008.

[31] J.-P. Aumasson, W. Meier, and R. C.-W. Phan, "The hash function family lake," in International Workshop on Fast Software Encryption. Springer, 2008, pp. 36–53.

[32] D. J. Bernstein et al., "Chacha, a variant of salsa20," in Workshop record of SASC, vol. 8, 2008, pp. 3–5.

[33] H. E. Michail, G. S. Athanasiou, V. I. Kelefouras, G. Theodoridis, T. Stouraitis, and C. E. Goutis, "Area-throughput trade-offs for sha-1 and sha-256 hash functionsâĂŹ pipelined designs," Journal of Circuits, Systems and Computers, vol. 25, no. 04, p. 1650032, 2016.

[34] L. Li, S. Lin, S. Shen, K. Wu, X. Li, and Y. Chen, "High-throughput and area-efficient fully-pipelined hashing cores using bram in fpga," Microprocessors and Microsystems, vol. 67, pp. 82–92, 2019.

[35] T. N. T. T. Duong, Le Vu Trung and L. D. Khai, "A fast approach for bitcoin blockchain cryptocurrency mining system," Integration, vol. 74, pp. 107–114, 2020.

[36] 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. 139 634–139 646, 2020.

[37] 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 Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 4, pp. 1553–1566, 2021.