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] 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.

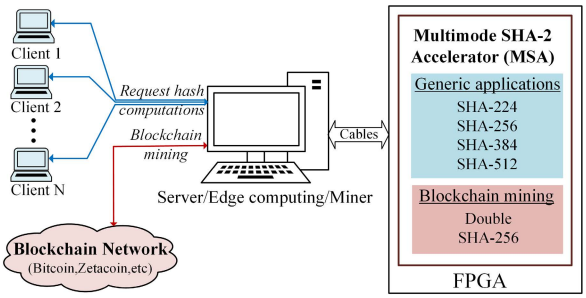

FIGURE 1. High-level diagram of the proposed system.

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

Algorithm 1 Hash = SHA-2(Message)

SHA224/256:
R = 64, S = 512, Llen = 64, D = 32, nH = 7/8
M consists of N 512-bit padded blocks.
$H^0_{[0:7]}$: 32-bit square root of the first 8 primes.
$K_{[0:63]}$: 32-bit square root of the first 64 primes.
SHA384/512:
R = 80, S = 1024, Llen = 128, D = 64, nH = 6/8
M consists of N 1024-bit padded blocks.
$H^0_{[0:7]}$: 64-bit square root of the first 8 primes.
$K_{[0:79]}$: 64-bit square root of the first 80 primes.
$L_{[0:Llen-1]}$ = length_in_bit(message)
N = (L ÷ S) + 1
($M^{[0:N-1]}$, N) = Padding (message)
Padding:
k = S − (1 + D + (L mod S))
Pad = {1, zeros(1,k), L}
$M^{[0:N-2]} = \text{message}[0 : ((N-2) * S) - 1]$
$M^{N-1} = \{\text{message}[(N-2) * S : L - 1], \text{Pad}\}$
for t ← 0 to (N-1) **do**
$W_{[0:R-1]}$ = Message_Expansion(M^t)
Message Expansion:
for i ← 0 to (R-1) **do**
if i < 16 **then**
$W_i = M^t_{[i*D:(i+1)*D]}$
else
$W_i = W_{i-16} + \sigma_0(W_{i-15}) + W_{i-7} + \sigma_1(W_{i-2})$
end if
end for
H^{t+1} = MessageCompression(H^t, K, W)
Message Compression:
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
for i ← 0 to (R-1) **do**
$T_1 = h + \Sigma_1(e) + Ch(e,f,g) + K_i + W_i$
$T_2 = h + \Sigma_0(a) + Maj(a,b,c)$
h = g, g = f, f = e, e = d + T_1,
d = c, c = b, b = a, a = T_1 + T_2
end for
$H^{t+1}_0 = H^t_0 + a, \ldots H^{t+1}_7 = H^t_7 + h$
end for
return Hash = $\{H^N_0, \ldots, H^N_{nH-1}\}$

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 $L + 1 + k \equiv 448 \bmod 512$ for SHA-224/256 functions or the equation $L + 1 + k \equiv 896 \bmod 1024$ 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 ($16 \leq i \leq R\text{-}1$) 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 ($0 \leq i \leq R\text{-}1$) 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.

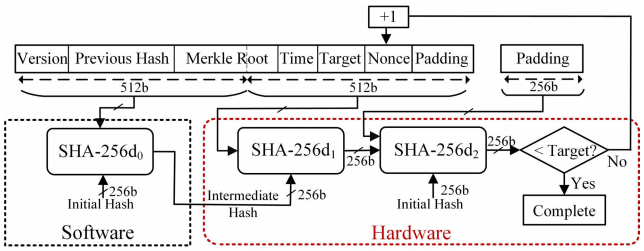

FIGURE 2. Double SHA-256 architecture for blockchain mining.

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

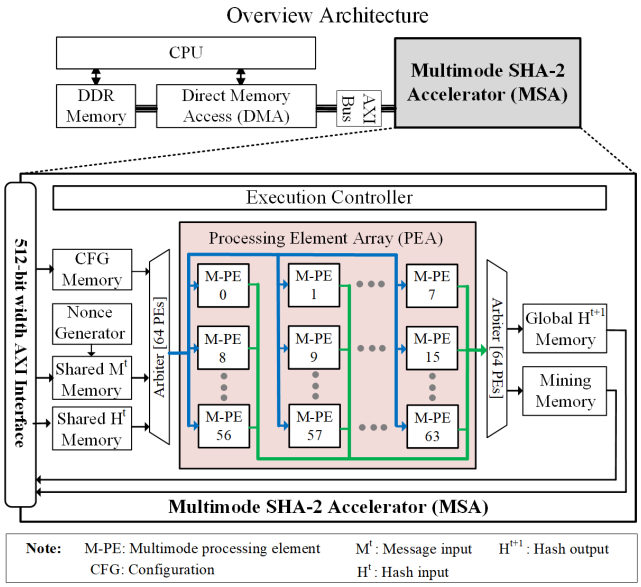

FIGURE 3. Overview architecture of the proposed multimode SHA-2 accelerator at the system-on-chip level.

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

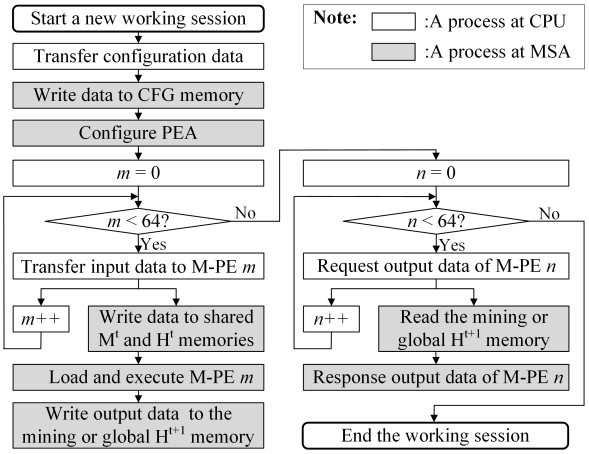

FIGURE 4. High-level flowchart of the working session on FPGA.

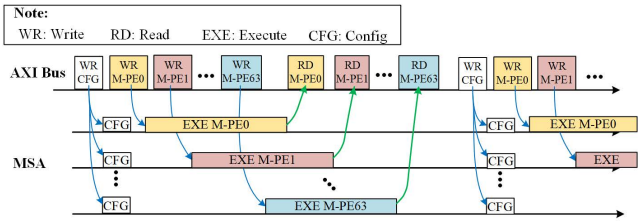

FIGURE 5. Timing chart of the multimode processing element execution.

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

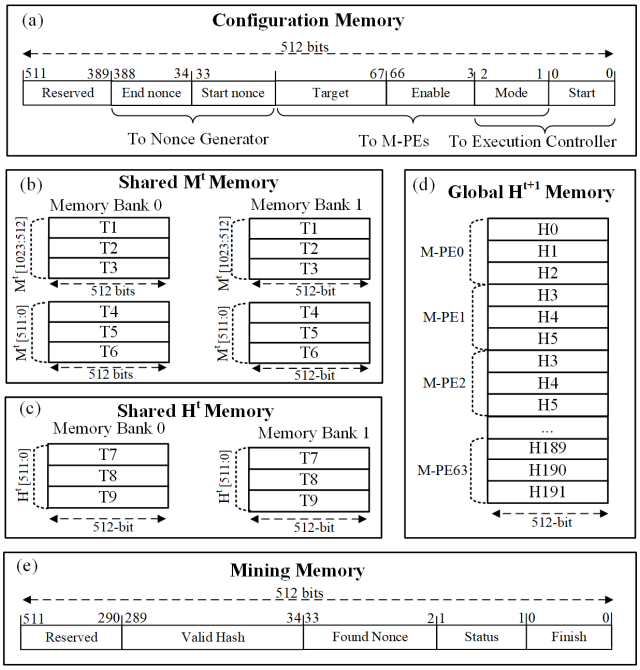

FIGURE 6. The memory organization of the MSA: (a) config memory; (b) shared W^t memory; (c) shared H^t memory; (d) global H^{t+1} memory; (e) mining memory.

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

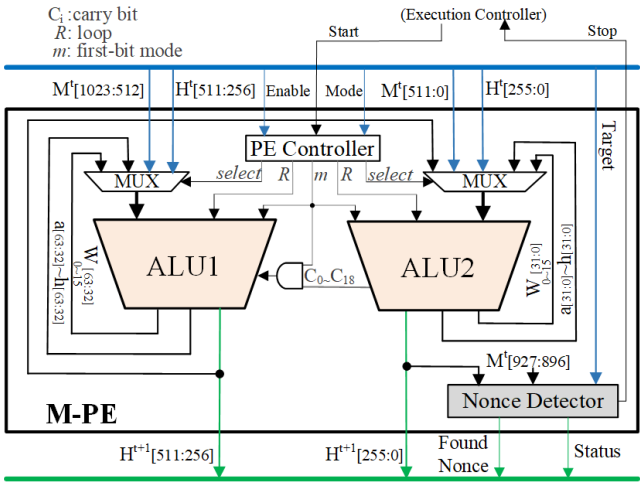

FIGURE 7. Multimode processing element (M-PE) architecture.

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

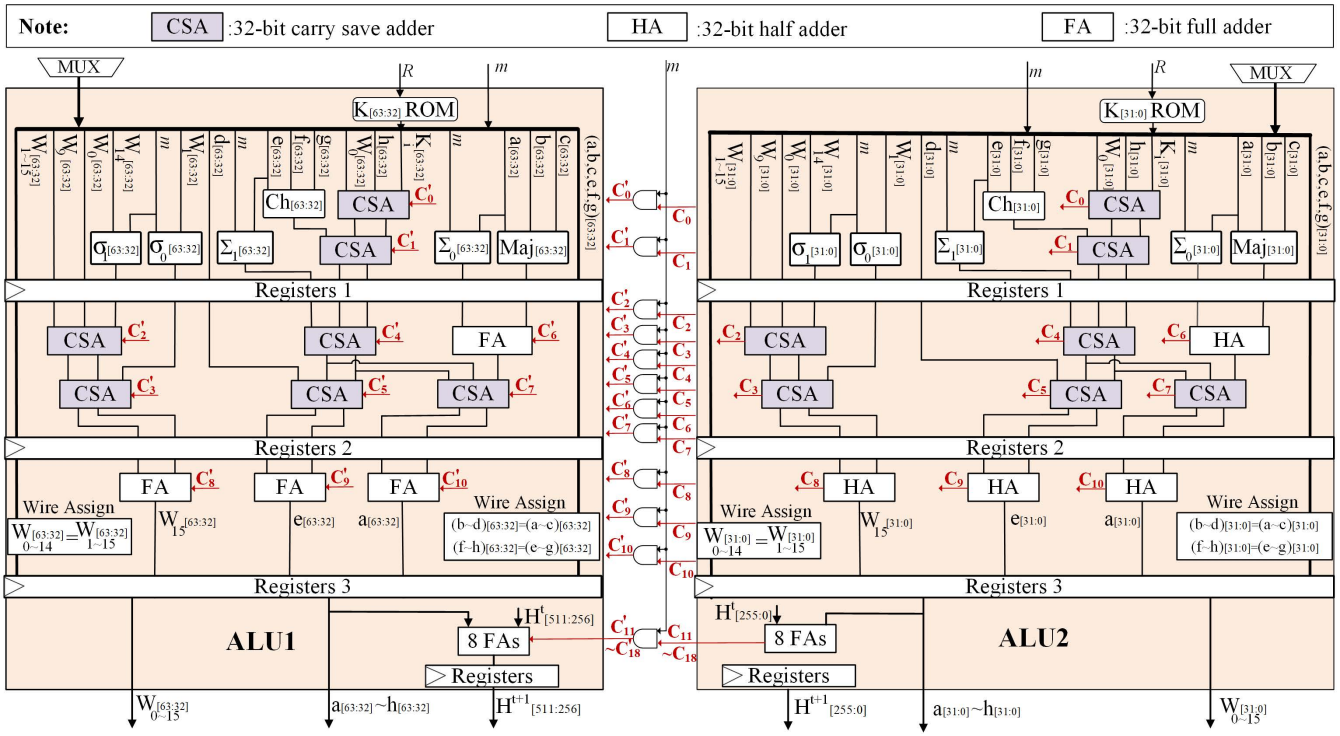

FIGURE 8. The three-stage pipelined dual ALU architecture.

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

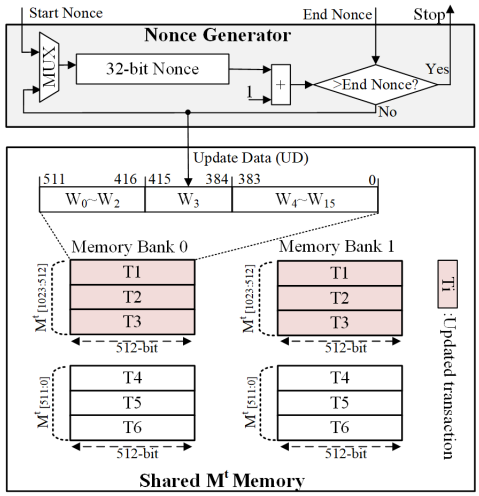

FIGURE 9. Nonce generator architecture.

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

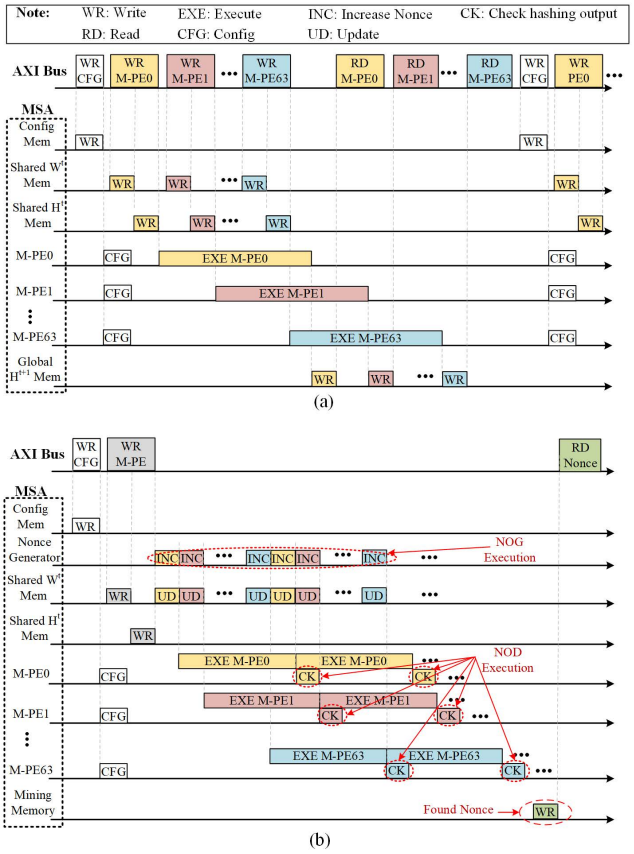

FIGURE 10. Detailed timing chart of the proposed MSA in (a) generic application and (b) blockchain mining.

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,..., 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

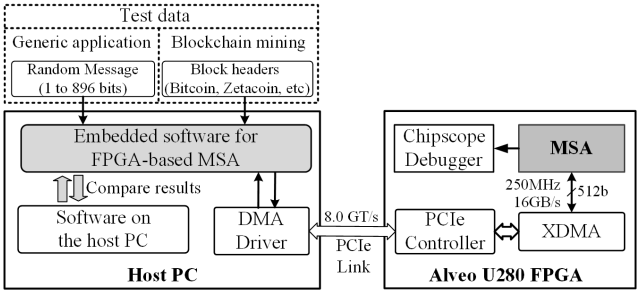

FIGURE 11. Implementation and verification of the proposed MSA on a Xilinx Alveo U280 FPGA.

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).

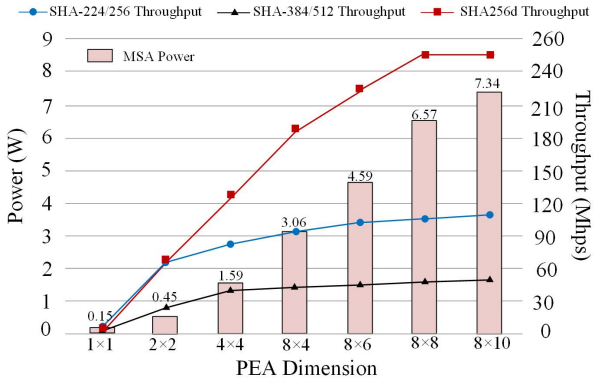

FIGURE 12. Power and throughput of the seven MSA versions with seven different processing element array (PEA) dimensions.

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.

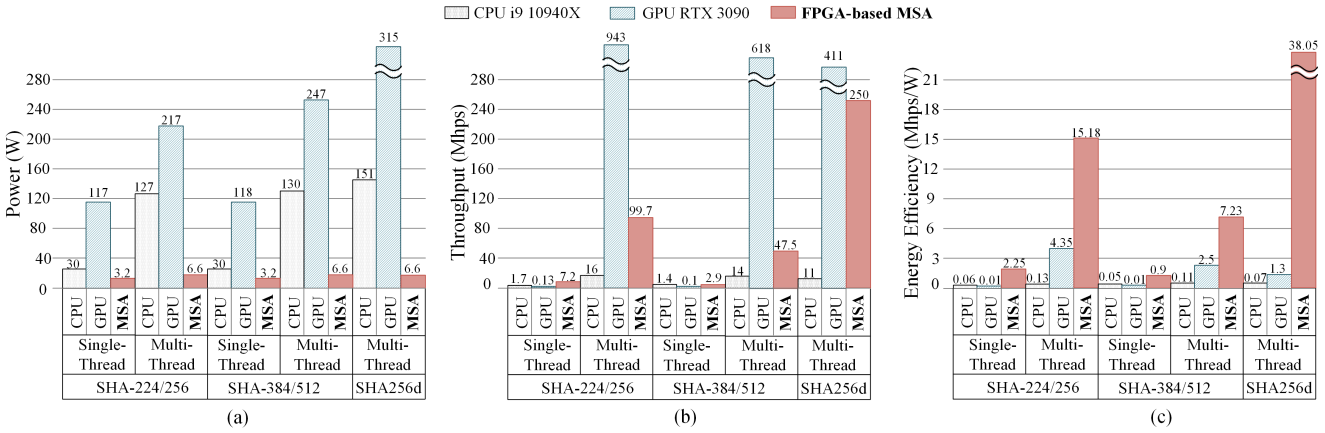

FIGURE 13. Comparison of the proposed MSA with a state-of-the-art CPU and GPU: (a) Power, (b) throughput, and (c) energy efficiency comparisons.

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.