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

A High-Performance Multimem SHA-256 Accelerator for Society 5.0

THI HONG TRAN, HOAI LUAN PHAM, and YASUHIKO NAKASHIMA

Graduation School of Information Science, Nara Institute of Science and Technology (NAIST), Nara, Japan (e-mail: hong, pham.hoai_luan.ox7, nakashim@is.naist.jp)

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

This research was supported by Japan Science and Technology Agency (JST) PRESTO Grant Number 2020A031.

ABSTRACT The development of a low-cost high-performance secure hash algorithm (SHA)-256 accelerator has recently received extensive interest because SHA-256 is important in widespread applications, such as cryptocurrencies, data security, data integrity, and digital signatures. Unfortunately, most current researches have focused on the performance of the SHA-256 accelerator but not on a system level, in which the data transfer between the external memory and accelerator occupies a large time fraction. In this paper, we solve the state-of-art problem with a novel SHA-256 architecture named the multimem SHA-256 accelerator that achieves high performance at the system on chip (SoC) level. Notably, our accelerator employs three novel techniques, the pipelined arithmetic logic unit (ALU), multimem processing element (PE), and shift buffer in shift buffer out (SBi-SBo), to reduce the critical path delay and significantly increase the processing rate. Experiments on a field-programmable gate array (FPGA) and an application-specific integrated circuit (ASIC) show that the proposed accelerator achieves significantly better processing rate and hardware efficiency than previous works. The accelerator accuracy is verified on a real hardware platform (FPGA ZCU102). The accelerator is synthesized and laid out with 180 nm complementary metal oxide semiconductor (CMOS) technology with a chip sized $8.5mm \times 8.5mm$, consumes 1.86 W, and provides a maximum processing rate of 40.96 Gbps at 80 MHz and 1.8 V. With FPGA Xilinx 16 nm FinFET technology, the accelerator processing rate is as high as 284 Gbps.

INDEX TERMS SHA-256, Blockchain, society 5.0, cryptography hash function, SHA-256 accelerator, FPGA, ASIC

I. INTRODUCTION

Cryptography secure hash algorithm (SHA)-2 plays an important role in developing super smart society 5.0 because it is required for data security and data integrity purposes in many applications, such as the digital signature algorithm (DSA) [1], hash-based message authentication code (HMAC) [2], pseudorandom number generation (PRNG) [3], and message authentication in internet protocol security (IPSec) [4]. Recently, the representative SHA-2 hash family named SHA-256 has become the key component for securing decentralized blockchain networks such as Bitcoin and Ethereum [5]. To secure the network, the double SHA-256 must be repeatedly computed a large number of times until a valid hash value that is smaller than a predefined target is found [5]. This results in the well-known disadvantage of the Bitcoin blockchain network of extremely high energy consumption. It was reported that the power consumption of the Bitcoin network reached 64 TWh in 2020, which is even higher than the total energy consumption of several nations such as Switzerland and the Czech Republic. In the field of the Internet of Things (IoT), SHA-256 should be implemented in power-constrained sensors or edge nodes to secure data and/or check the integrity of data. For these reasons, developing a high-performance SHA-256 accelerator has become a research trend in recent years.

Several techniques have been widely used in published works. For instance, unrolled architectures [6], [7] were implemented to improve the processing rate in trade-offs with large areas and high power consumption. Pipelined architectures have been developed in many works, such as [8], [9], and [10], to increase the processing rate of the SHA-256 core. Furthermore, [8] employed the parallel counter technique to reduce the circuit area and critical path delay. Recently, [11] introduced the rescheduling of the SHA-256 round computation method to reduce the critical path delay and improve the processing rate of the SHA-256 accelerator.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3063485, IEEE Access

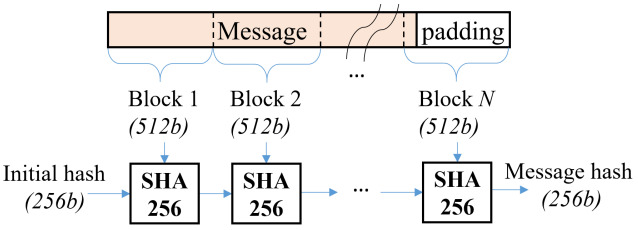

FIGURE 1. The generation of a SHA-256 hash value for a long message

Generally, these techniques merely rearranged the operators required for SHA-256 round computation such as adders, logic gates, and rotations, in an appropriate and efficient way to achieve high performance in terms of the processing rate, circuit area, or critical path delay. In another approach, the work in [5] introduced the idea of a compact message expander (CME) in which the input data characteristic of double SHA-256 was utilized to significantly optimize the hardware cost and power consumption of the circuit for the case of Bitcoin mining. However, the proposed work aimed to support the double SHA-256 only in Bitcoin mining.

The reality is that an SHA-256 circuit is just a component of an embedded system under the control of a microprocessor. The processing time of the SHA-256 circuit includes not only the time for calculating the hash value but also the time for transferring data between the external memory and SHA-256 circuit. Unfortunately, the aforementioned conventional works focused merely on the calculation time of the SHA-256 circuit. The data transfer time was ignored. An efficient SHA-256 accelerator must provide a high computation rate and be very compatible with the embedded systems so that the data transfer can be performed efficiently and fast.

In this study, we propose a high-performance hardware architecture for SHA-256 by considering both the hash calculation time and data transfer time. We named the architecture the multimem SHA-256 accelerator because multiple local memories are implemented in each processing element (PE) to temporarily store the input and output data of SHA-256. These memories work appropriately and efficiently to reduce the data transfer time between the accelerator and external memory. Several ideas of interest such as pipelined arithmetic logic units (ALUs), multimem PEs, and shift buffer in shift buffer out (SBi-SBo), are introduced in this paper.

The remainder of this paper is organized as follows. Section II provides the background of the research. Section III describes our proposed multimem SHA-256 architecture. Section IV reports our evaluation in terms of application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) experiments. Finally, section V concludes the paper.

II. BACKGROUND

A. SHA-256 ALGORITHM

The SHA-2 family is a set of cryptographic hash functions designed and published by the US National Institute of

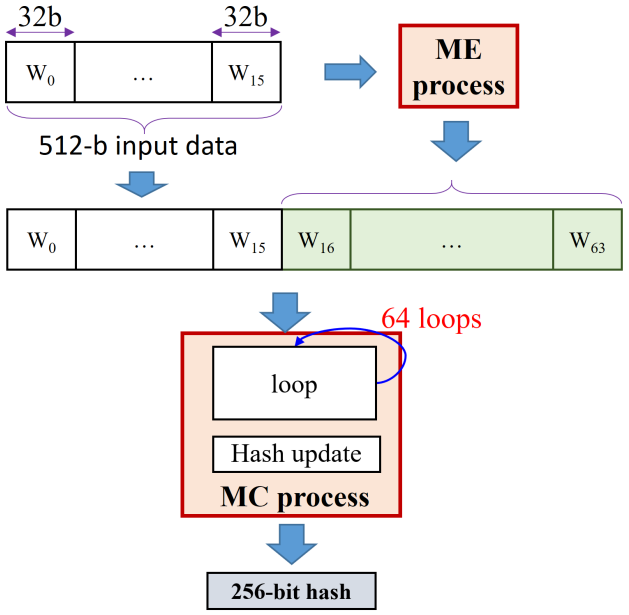

FIGURE 2. The overview operation of the SHA-256 algorithm

Standards and Technology (NIST) in 2002 [12]. The SHA-2 family includes six hash functions named SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. They are actually the same algorithm but with different word lengths, constant parameters, and initialization values. SHA-256 is known as representative of the SHA-2 family and is currently applied to secure data in many applications such as decentralized blockchain, DSA, and HMAC. SHA-256 calculates a 256-bit hash value for an input message of 512 bits. The real application may need to calculate the hash value for a very long message. In such cases, the message is divided into many 512-bit data blocks. If the last block is smaller than 512 bits, padding is added. The hash calculation for a long message is shown in Fig. 1. The SHA-256 algorithm computes intermediate hash values for data blocks one by one, in which the hash result of the current block becomes the input initial hash for hash computing of the next data block. The result of the final data block is considered to be the hash value of the entire message.

Fig. 2 illustrates the overview operation of the SHA-256 algorithm. It includes two processes named the message expander (ME) and message compressor (MC). The ME process expands the 512-bit input message into 64 chunks of 32-bit data W_j ($0 \leq j \leq 63$). In the first 16 rounds, the ME parses the 512-bit message into 16 32-bit data chunks (denoted as W_j, j = 0 to 15, where j is the round index). In the final 48 rounds, the ME calculates 48 chunks of 32-bit data W_j ($16 \leq j \leq 63$) following eq. (1). Three 32-bit adders and two logical functions $\sigma_0(x)$ and $\sigma_1(x)$ are needed to compute W_j ($16 \leq j \leq 63$). $\sigma_0(x)$ and $\sigma_1(x)$ are computed as eq. (2) and eq. (3). Note that $S^n(x)$ and $R^n(x)$ denote the right rotation and the right shift of data x by n bits,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3063485, IEEE Access

respectively.

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

The MC process computes the 256-bit hash value from the outputs of the ME process (64 chunks of W_j ($0 \leq j \leq 63$)). The process involves two main steps: *loops* and *hash updates*. In the *loop* step, eight loop hash values (denoted a, b, c, d, e, f, g, h) are initialized by the initial hash values H_0, H_1,...,H_7. The loop hash values a, b, c, d, e, f, g, h are then computed and updated through 64 loops. In each loop (loop j, $0 \leq j \leq 63$), eqs. (4) to (8) are applied.

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

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

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

In the *hash update* step, the final 256-bit hash value, which is divided into 8 chunks of 32-bit data HO_0, HO_1,...,HO_7, is computed by adding the initial hashes H_0, H_1,...,H_7 to the loop hashes a, b, c, d, e, f, g, h, as illustrated in eq. (13).

$$HO_0 = H_0 + a; \ldots; HO_7 = H_7 + h \quad (13)$$

B. SHA-256 CHARACTERISTICS AND PRELIMINARY IDEA FOR THE HIGH PERFORMANCE ACCELERATOR

There are three characteristic points of SHA-256 that should be noted. **First**, the SHA-256 algorithm needs only low-cost arithmetic logic operators such as adders, shifts, rotations, and XORs. No complex operators such as multipliers and dividers are required. **Second**, the number of operators (nots, XORs, adders, shifts, rotations, etc.) per loop calculation is significantly large (approximately 50 operators per loop). **Third**, data dependence among the loops is present. This means that the current loop needs the result of the previous loops for calculation. In addition, some common data are required in multiple loops. Because of these characteristics, the current high-performance CPU (Intel multicores) and GPU (GPU tesla V100, GTX 1080, etc.) platforms do not perform well in calculating the SHA-256 algorithm. Although these general purpose platforms are rich in hardware resources, they still need a large number of clock cycles to compute a single loop of SHA-256, which results in a low processing rate. One of the main reasons is that memory blocks such as double data random access memory (DDRAM) and cache L1 and L2 of these platforms are placed far from the calculation unit, and the data transfer time between the memories and calculation units thus constitutes a large amount of the total processing time. Furthermore, most of the existing hardware resources of CPUs and GPUs, such as multipliers and exponentials, are useless for SHA-256 but still consume energy. Despite these disadvantages, multicore CPUs and GPUs are currently considered to be the most applicable hardware platforms for calculating SHA-256 in Bitcoin mining and other blockchain networks.

In another approach, the systolic array-based accelerator named EMAXVR in [13] and its improved version in [14] were applied to reduce the data access time by implementing local memory near the ALU. Although this platform achieves high performance on image processing and AI learning [13], [14], its performance for computing SHA-256 is very poor [15]. The key reason for the low processing rate is the data-dependent characteristic of the SHA-256 algorithm, as mentioned above. The processing element (PE) architecture of EMAXVR is not yet suitable for SHA-256 computation; therefore, a large number of PEs are required to compute a single hash loop. Furthermore, the data dependence among the loops requires 1) multiple copies of data to be stored in different PEs, which results in low hardware efficiency, and 2) external memory data transfer after the completion of a loop, which results in a low processing rate.

Studying the weaknesses of the existing hardware platforms for SHA-256 and deeply understanding the characteristics of SHA-256, we propose a novel multimem SHA-256 accelerator that achieves high performance in both hardware efficiency and processing rate. We propose multiple local memory structures near the ALU for reducing data access time and improving hardware efficiency. Furthermore, we design a processing element (PE) architecture that is able to compute all loops of the hash function without requiring data from the nearby PEs. Thus, the PEs can work independently and parallel, and no wire connections are required among the PEs. As a result, the accelerator does not need to frequently request data from external memories, and the data access time is significantly shortened. Furthermore, the hardware efficiency and processing rate of the SHA-256 accelerator are expected to be remarkably enhanced.

III. THE PROPOSED MULTIMEM SHA-256 ACCELERATOR

A. OVERVIEW OF THE ARCHITECTURE

Fig. 3 shows the overview architecture of our proposed accelerator. It can be employed in any embedded system via an advanced extensible interface (AXI) bus. The CPU is the main microprocessor controlling operation of the entire embedded system. Basically, the CPU is busy with many tasks, including the control of the hash calculation inside the SHA-256 accelerator. In the task of controlling the accelerator, the CPU must send commands to transfer data from DDR memory to the SHA-256 accelerator via data buses (such

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3063485, IEEE Access

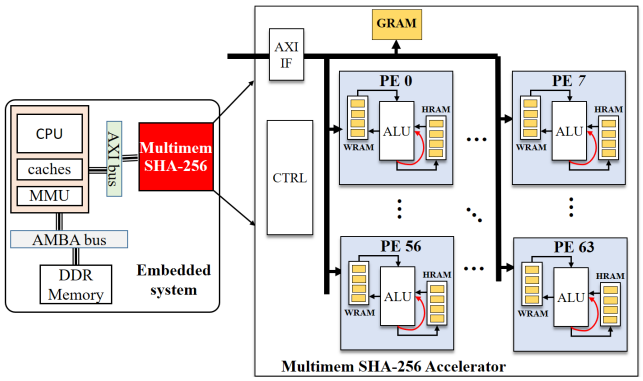

FIGURE 3. Overview of the architecture of the proposed multimem SHA-256 accelerator

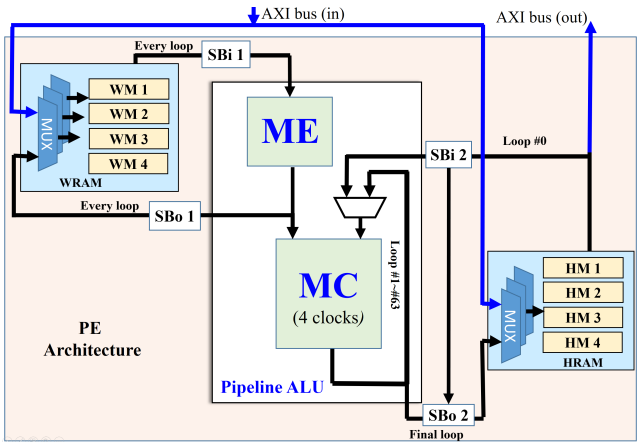

FIGURE 4. Block diagram of a processing element (PE)

as advanced microcontroller bus architecture (AMBA) buses and AXI buses), reconfigure the accelerator, and send commands to read results from the accelerator to DDR memory. The data transfer between the accelerator and DDR memory is basically much slower than the data transfer inside the accelerator. In addition, the data buses are not dedicated to only the operation of the SHA-256 accelerator. They may not immediately respond to the request from the accelerator and may make the accelerator wait for data. Therefore, we implement multiple memory blocks inside the accelerator to reduce the number of external DDR memory requests and to increase the accelerator processing rate.

The main component of the accelerator is an array of PEs responsible for computing the SHA-256 hash value. In addition, it has an AXI bus interface (AXI IF), a global memory (GRAM) for storing global parameters such as the accelerator configuration parameters, and a controller (CTRL) providing control signals for the PEs, AXI IF, and GRAM.

Fig. 4 shows the internal architecture of a PE. It includes a pipelined ALU, multiple memory structures named WRAM and HRAM, and shift buffer in (SBi1, SBi2) shift buffer out (SBo1, SBo2). The pipelined ALU includes the arithmetic

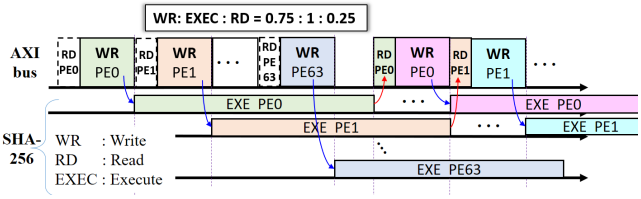

FIGURE 5. Timing chart of the entire multimem SHA-256 accelerator

logic operators required for computing all loops of the hash function. The WRAM and HRAM blocks include local memories storing the input and output data of the PE so that the PE can compute hash values of a large number of SHA-256 functions rapidly without requesting data from external DDR memory. The SBi1, SBi2, SBo1, and SBo2 temporarily store and rearrange the data flow between the pipelined ALU and the WRAM/HRAM blocks. All these blocks cooperate together to achieve the following two things: 1) a PE can compute all the loops, and 2) the hardware efficiency of the ALU is 100%.

In general, a PE of our accelerator computes all 64 loops of SHA-256 within 64 clock cycles on average at a latency of $64 \times 4 = 256$ clock cycles. In other words, the accelerator including 64 PEs can compute a hash value (64 loops) per clock cycle. Furthermore, the memory resources of WRAM and HRAM are efficiently allocated so that the 64-PE accelerator can uninterruptedly compute an amount of $64 \times 4 \times L$ functions without further requesting data from external memory, in which L is a pre-defined positive integer parameter. The AXI bus and PEs are designed to work in pipe-lined and parallel as shown in fig. 5. When the AXI bus is reading/writing data to/from a PE, the other PEs of accelerator is executing the hash calculation. As a result, the data transfer time between the accelerator and external DDR memory does not affected to the entire system processing rate (if the embedded system is equipped a fast enough AXI bus).

B. PIPELINED ALU ARCHITECTURE

As shown in Fig. 4, the pipelined ALU includes two blocks named ME and MC that compute the ME and MC processes of the SHA-256 function, respectively. Fig. 6 shows the circuit inside the pipelined ALU. To achieve high maximum frequency and high processing throughput, the ME block is designed to compute the ME process in 2 pipelined clock cycles named ME-1 and ME-2, while the MC block computes the MC process in 4 pipelined clock cycles named MC-1, MC-2, MC-3, and MC-4. Two stages of the ME process in parallel with the first two stages of the MC (named MC-1 and MC-2). The last two stages of the MC receive the output data from the ME block as well as its previous stage (MC-1 and MC-2) outputs to complete the loop calculation.

With this architecture, the ALU can compute a hash function loop in 4 pipelined clock cycles. It then feeds back the result of the current loop to recursively compute the next loop. Therefore, all 64 loops can be completed in a single

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3063485, IEEE Access

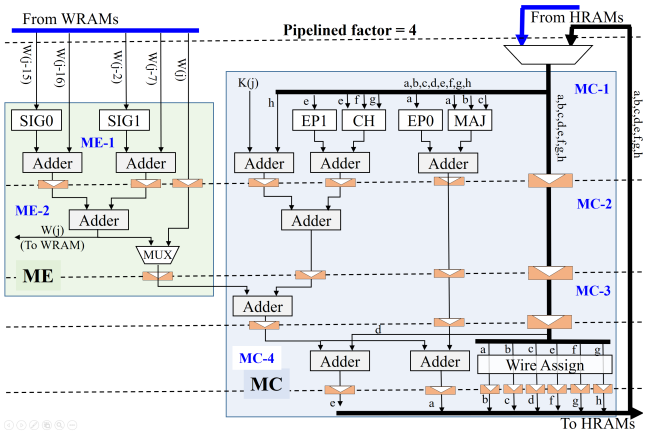

FIGURE 6. The pipelined arithmetic logic unit (ALU) architecture

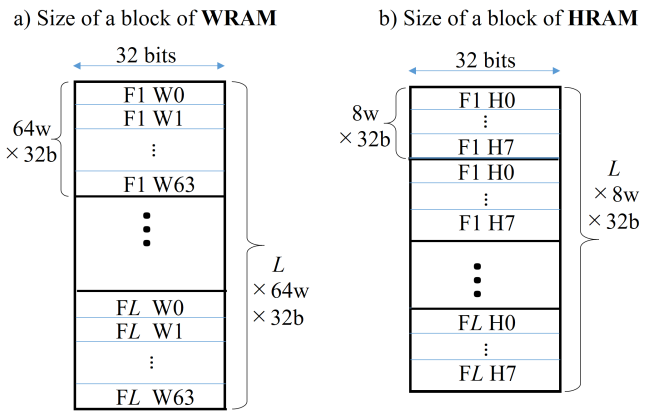

FIGURE 8. The memory map of a RAM block inside a) the WRAM and b) the HRAM

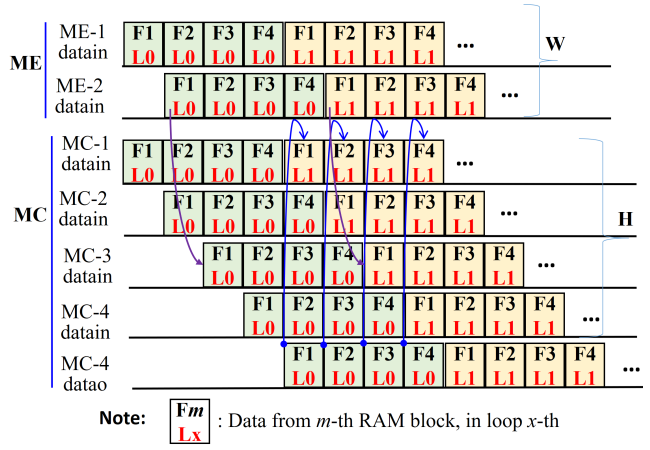

FIGURE 7. The timing chart of the ME and MC blocks inside the pipelined ALU when 4 data flows are processed

ALU. To efficiently use the hardware resources of the ALU (100% hardware efficiency), four data flows from the local memories WRAM and HRAM should be input to the 4 pipelined ALUs. The operation of the ME block is as follows. In the first 16 loops (loop $j, 0 \leq j \leq 15$), the ME block merely passes the input data W_j to its output port. In the last 48 loops (loop $j, 16 \leq j \leq 63$), the ME block computes W_j from the four input data W_{j-16}, W_{j-15}, W_{j-7}, W_{j-2} following eq. (1) to eq. (3). The result W_j is then 1) passed to the MC-3 stage of the MC block for further computing and 2) written into WRAM for future use. The operation of the MC block is as follows. In the first loop (loop $j = 0$), the loop hash values a, b, ..., and h are assigned to the initial hash values H_0, H_1, ..., and H_7 read from the HRAM. The values of a, b, ..., and h are then computed and updated in four pipelined stages MC-1, MC-2, MC-3, and MC-4 following eq. (4) to eq. (12). Because four data flows share the same hardware resources of the ALU, all the ALU stages are always busy. This means that we achieve 100% hardware efficiency for the ALU resources. By the time MC-4 stage is computing for data flow 1, MC-3 computes for data flow 2, MC-2 for data flow 3, and MC-1 for data flow 4.

Fig. 7 illustrates the timing chart of the ME and MC blocks when loop 0 and loop 1 are computed for the four hash functions F1, F2, F3, and F4. The updated values of a, b, ..., and h after the MC-4 stage are then fed back to become the inputs of the MC-1 stage. The calculation for the next loop (loop $j = 1$) is then started. This operation is repeated until all 64 loops of the hash function are completed. The values of a, b, ..., and h after the completion of 64 loops are transferred to SBo2 to be written into the HRAM.

C. MULTIPLE LOCAL MEMORY STRUCTURES WRAM AND HRAM

The local memories WRAM and HRAM store the input and output data of the ALU. While WRAM stores the message words W_j ($0 \leq j \leq 63$), HRAM stores the hash values H_0, ..., and H_7 of the message. Because the ALU requires 4 data flows to efficiently use the hardware resources, we implement 4 identical RAM blocks in both WRAM and HRAM, named WM1, WM2, WM3, WM4, and HM1, HM2, HM3, HM4 (see fig. 4). Two-port RAMs are used because the RAMs may need to simultaneously read and write data.

Fig. 8 a and b shows the size of each RAM block on the WRAM and HRAM sides, respectively. Each block of WRAM, such as WM1 and WM2, is sized $L \times 64 \times 32$ bits and can simultaneously store 64 words W_j ($0 \leq j \leq 63$) of L hash functions, where $L \times 16 \times 32$ bits equivalent to 16 words W_j ($0 \leq j \leq 15$) of L hash functions are written into each RAM before starting the hash calculation. The remaining $L \times 48 \times 32$ bits, equivalent to 48 words W_j ($16 \leq j \leq 63$) of L hash functions, are written during the hash calculation by the ALU. Note that L is a parameter that can be changed before fabricating the chip.

Each block of HRAM, such as HM1 and HM2, is sized $L \times 8 \times 32$ bits and can store 8 chunks of 32-bit hash values H_0, ..., and H_7 of L hash functions. Our accelerator can work in two different modes. The first mode computes the hash values of independent hash functions. In this mode, the initial hash value of L hash functions is written into all

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3063485, IEEE Access

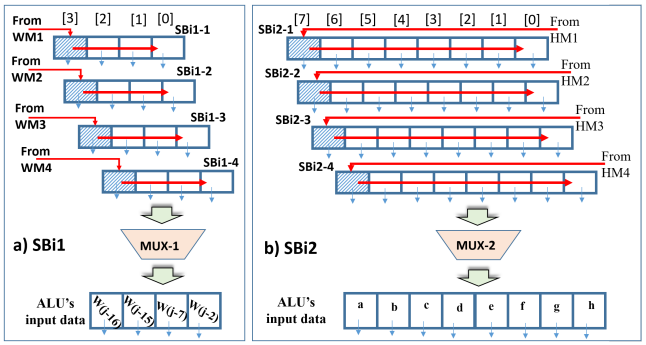

FIGURE 9. SBi1 and SBi2 architectures

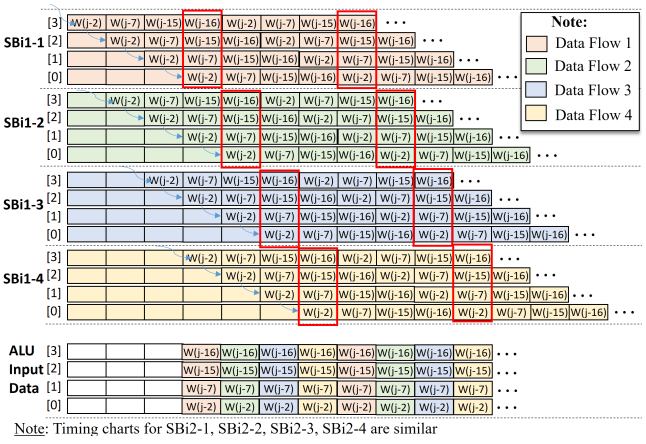

FIGURE 10. Timing chart of SBi1

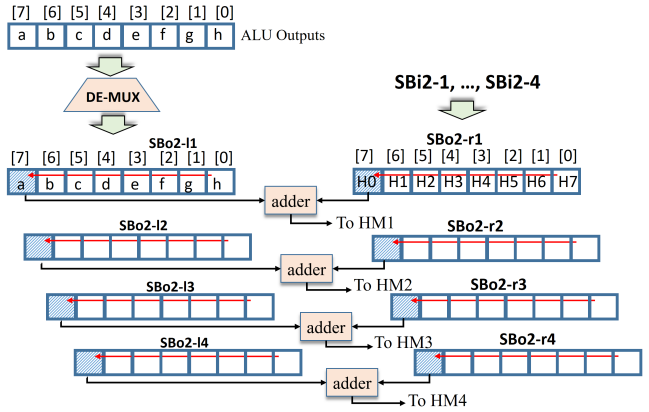

FIGURE 11. SBo2 architecture

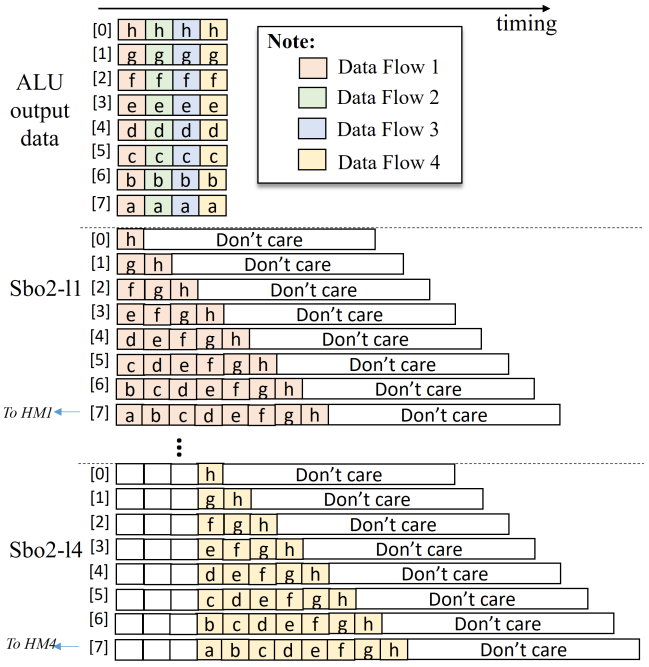

FIGURE 12. Timing chart of SBo2

blocks of the HRAM before starting the hash calculation. The content of the HRAM blocks is then be overwritten by the final hash values calculated by the ALU. These values are read out of the accelerator after the hash calculation of all L hash functions is completed. The second mode computes the hash values of a long message divided into L 512-bit data blocks, as illustrated in fig. 1. In this mode, only the initial hash value of the first data block is written into 4 RAM blocks of HRAM before the accelerator is activated. During the hash calculation, the output hash value of a data block is written into the HRAM to be used as the initial hash value of the next adjacent data block. The output hash value of the final data block is the hash value of the entire message, which is then read out of the accelerator.

D. SHIFT BUFFER IN SHIFT BUFFER OUT (SBI-SBO)

The ALU requires multiple input data such as W_{j-16}, W_{j-15}, W_{j-7}, W_{j-2}, and $a, \ldots, h$ at the same time. However, each RAM block is able to read out only one data point per clock cycle. Similarly, the ALU simultaneously outputs many data values, such as the updated versions of $a, \ldots,$ and h, but only one data point can be written into a block of HRAM per clock cycle. We thus cannot directly connect the data flows between WRAM/HRAM and ALU. The SBi and SBo are required to collect and schedule the timing of data flows.

Fig. 9 a and b shows the SBi1 and SBi2 architectures that connect the outputs of WRAM and HRAM to the ALU inputs, respectively. SBi1 includes 4 shift buffers named SBi1-1, SBi1-2, SBi1-3, and SBi1-4 receiving data flows from WM1, WM2, WM3, and WM4 of WRAM, respectively. Each shift buffer has 4 32-bit shift registers that receive data from the WRAM and then shift the data to the right once per clock cycle. The timings of 4 data flows from 4 RAM blocks WM1, ..., WM4 are interleaved, as shown in fig. 10. The multiplexer MUX-1 then selects the appropriate dataset from 4 data flows one per clock cycle to input to the ALU.

Similarly, SBi2 shown in Fig. 9b has 4 shift buffers named SBi2-1, SBi2-2, SBi2-3, and SBi2-4 receiving data flows from HM1, HM2, HM3, and HM4 of HRAM, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3063485, IEEE Access

TABLE 1. Comparison based on ASIC results.

Design Type	Research	Tech. (T)	Measured Frequency (MHz)	Measured Area (Gate)	# Clock Cycles /Hashes	Throughput P (Mbps)		Normalized Hardware Efficiency (Kbps/Gate)
						Measured Value	Normalized Value *	
Only ALU	[17], 2003	180 nm	200	22,000	65	1,575	1,575	88
	[18], 2005	130nm	333	15,329	72	2,370	1,712	112
	Prop.	**180nm**	**366**	**23,870**	**64**	**2,930**	**2,930**	**123**
No RAM PE	[22], 2009	250nm	136	8,588	490	142	197	23
	Prop.	**180nm**	**304**	**56,800**	**64**	**2,432**	**2,432**	**43**
No RAM 64-PE	**Prop.**	**180nm**	**272**	**3,375,700**	**1**	**139,131**	**139,131**	**41**

* : Values are normalized to 180 nm, $P_{180} = P_T \times (\frac{180}{T})$

Each shift buffer has 8 32-bit shift registers that receive data $a, \ldots, h$ from the HRAM and then shift the data to the right once per clock cycle. The multiplexer MUX-2 then selects the appropriate dataset from one of four data flows to give to the ALU.

SBo1 delivers the result W_j of the ME block to the corresponding RAMs of the WRAM. Because only one value W_j is generated per clock cycle, shift buffers are not required, and only a 1-to-4 multiplexer is needed inside SBo1.

SBo2 has two main functions. First, it collects and schedules the loop hash words $a, \ldots, h$ output from the ALU after 64 loops are complete. Second, it calculates the final hash values $HO_1, \ldots, HO_7$ by adding the loop hash words $a, \ldots, h$ and the initial hash values $H_0, \ldots, H_7$ following eq. (13). Fig. 11 shows the inside of SBo2. It includes four shift buffers on the left named SBo2-l1, SBo2-l2, SBo2-l3, and SBo2-l4 and four shift buffers on the right named SBo2-r1, SBo2-r2, SBo2-r3, and SBo2-r4. Each shift buffer has 8 32-bit shift registers. The buffers on the left (SBo2-l1, SBo2-l2, SBo2-l3, SBo2-l4) store the ALU outputs $a, \ldots, h$, while the buffers on the right (SBo2-r1, SBo2-r2, SBo2-r3, SBo2-r4) receive the initial hash values $H_0, \ldots, H_7$ from SBi2. Each left-right set of buffers aims to process one data flow. The content of the left-right set of buffers is shifted to the left (direction of the most significant register) once per clock cycle. The values from the most significant registers are read out and added. The addition results become the final hash words $HO_1, \ldots, HO_7$ that are written into the corresponding RAM blocks (HM1, ..., HM4) of the HRAM.

Fig. 12 illustrates the timing chart of the SBo2. First, the ALU output data, such as $a, \ldots,$ and h, of the 4 data flows are stored in the 8 registers of the shift buffers SBo2-l1 to SBo-l4 in 4 interleave clock cycles. The values of the most significant register (for example SBo2-l1[7]), which are a, b, c, d, e, f, g, and h at clock cycles #1, #2, #3, #4, #5, #6, #7, respectively, are read out to add to the initial hash values H_0, $\ldots$, H_7 before writing into the HRAM (for example WM1). It needs 8 clock cycles to complete the writing of 8 32-bit hash words for one data flow and $8 + 3 = 11$ clock cycles to complete the writing for 4 data flows in pipelined mode.

IV. EVALUATION

In this section, we evaluate the performance of our proposed multimem SHA-256 accelerator from both ASIC and FPGA experimental results.

A. ASIC EXPERIMENT

The proposed multimem SHA-256 circuits for cases of 1-PE and 64-PEs were coded in Verilog and synthesized in an ASIC using a Synopsys Design Compiler with the Rohm $0.18\mu m$ CMOS standard cell library [16]. Table 1 shows the synthesized area of the SHA-256 circuit in comparison with those of other works. To ensure a fair comparison, we provide the processing throughput in terms of both the number of clocks per hash function and Mbps. We also normalize the processing throughput Mbps and hardware efficiency of other works to the same 180 nm technology. The proposed architecture in cases of both 1 PE and 64 PEs exhibits significantly better processing throughput and hardware efficiency. Compared with the works in [17] and [18], which reported only the ALU circuit results, our ALU generates a processing rate as high as 2,930 Mbps, which is 1.7 times and 1.9 times faster than the works in [17] and [18], respectively. Our ALU hardware efficiency is 123 Kbps/gate, which is 1.1 times and 1.4 times better than those of [17] and [18], respectively.

The proposed architecture with only 1 PE (No RAM PE) provides a processing rate as high as 2,431 Mbps, which is 12.3 times faster than that of the work in [22]. Its hardware efficiency is 43 Kbps/gate, which is almost 2 times better than that of [22]. Notably, our architecture with 64 PEs achieves the best processing rate. It provides up to 139 Gbps at a frequency of 272 MHz, which is 70 times faster than [22]. Note that the term "No RAM" refers to the synthesis result except the local memory resources inside the HRAM and WRAM.

Furthermore, we successfully layout the proposed SHA-256 circuit in the case of 64 PEs in ASIC technology with a Rohm 180 nm CMOS standard cell library. The size of the layout chip is $8.5mm \times 8.5mm$ and chip processing rate is 40.96 Gbps when a global operating voltage of 1.8 V and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3063485, IEEE Access

TABLE 2. A comparison of FPGA synthesis results

Device	Design Type	Research	Frequency (MHz)	Area		# Clock Cycles /Hashes	Throughput (Mbps)	Hardware Efficiency (Kbps/LUT)
				Slices/LUTs	FFs			
Virtex 2	Only ALU	[19], 2014	35	431	-	280	65	150
		[20], 2011	104	1,117	-	65	820	734
		[6], 2006	133	1,373	-	68	1,009	735
		[21], 2018	134	502	-	129	532	1,059
		Prop.	**249**	**1,067**	**1,901**	**64**	**1,989**	**1,864**
	No RAM PE	[22], 2009	136	779	-	490	75	96
		Prop.	**153**	**3,615**	**5,093**	**64**	**1,222**	**338**
	Full 64-PE	**Prop.**	**153**	**228,976**	**325,851**	**1**	**78,223**	**342**
Virtex E	Full PE	[23], 2002	88	1,261	-	520	87	69
		Prop.	**83**	**3,625**	**14,392**	**64**	**662**	**183**
	Full 64-PE	**Prop.**	**83**	**233,067**	**920,756**	**1**	**42,399**	**182**
Virtex 4	Only ALU	[19], 2014	50	422	-	280	91	217
		[11], 2020	256	979	-	66	1,984	2,027
		[10], 2017	171	610	-	65	1,345	2,205
		[21], 2018	222	485	-	129	881	1,817
		Prop.	**390**	**1,067**	**1,899**	**64**	**3,124**	**2,928**
	Full 64-PE	**Prop.**	**249**	**236,452**	**917,988**	**1**	**127,314**	**538**
Virtex 5	Only ALU	[19], 2014	64	139	-	280	118	847
		[21], 2018	272	273	-	129	1,080	3,956
		Prop.	**420**	**772**	**1899**	**64**	**3,360**	**4,353**
	No RAM PE	[24], 2014	179	2,796	-	65	1,410	505
		Prop.	**278**	**2,909**	**5,151**	**64**	**2,229**	**765**
	Full 64-PE	**Prop.**	278	**242,666**	**929,797**	**1**	**142,515**	**587**
Virtex 6	Only ALU	[25], 2015	271	960	-	68	2,040	2,125
		Prop.	**525**	**775**	**1,889**	**64**	**4,197**	**5,415**
	Full 64-PE	**Prop.**	**286**	**249,760**	**930,112**	**1**	**146,309**	**586**
Virtex 7	Full PE	[26], 2020	181	6,367	25,190	101	917	144
		Prop.	**222**	**2,241**	**14,245**	**64**	**1,775**	**727**
	Full 64-PE	**Prop.**	**222**	**178,518**	**916,745**	**1**	**113628**	**637**
Kintex UltraScale+	No RAM PE	[27], 2019	728	5,314	4,302	65	5,829	1,097
		Prop.	**555**	**2,241**	**14,245**	**64**	**4,444**	**1,907**
	Full 64-PE	**Prop.**	**556**	**145,248**	**905,609**	**1**	**284,448**	**1,958**

operating frequency of 80 MHz are applied.

B. FPGA EXPERIMENT

1) FPGA Synthesis Results

To ensure a fair comparison with other existing SHA-256 architectures such as [11], [19], and [20], we synthesized the proposed SHA-256 circuits on several Xilinx FPGA boards, including Virtex 2, E, 4, 5, 6, 7, and Kintex UltraScale. The comparison factors include the circuit area, processing rate, and hardware efficiency.

The results are shown in Table 2. Note that among the previous works, some reported only the results of the ALU part, some reported the results of full accelerators including ALU, controller, memories, etc., and some reported the full accelerator except the memory that temporarily stores the input/output data. To ensure a fair comparison with other works, we synthesized two versions (1-PE and 64-PEs) of our accelerator with many options such as "only ALU", "no RAM", and "full". In general, both versions 1-PE and 64-PEs of our proposed SHA-256 accelerator outperform the existing SHA-256 architectures in terms of processing rate and hardware efficiency. Some typical numerical results are as follows.

On the Virtex 2 device, the ALU circuit of our accelerator version 1-PE generates 1 hash value every 64 clock cycles, which is equivalent to a processing rate of 1,989 Mbps at

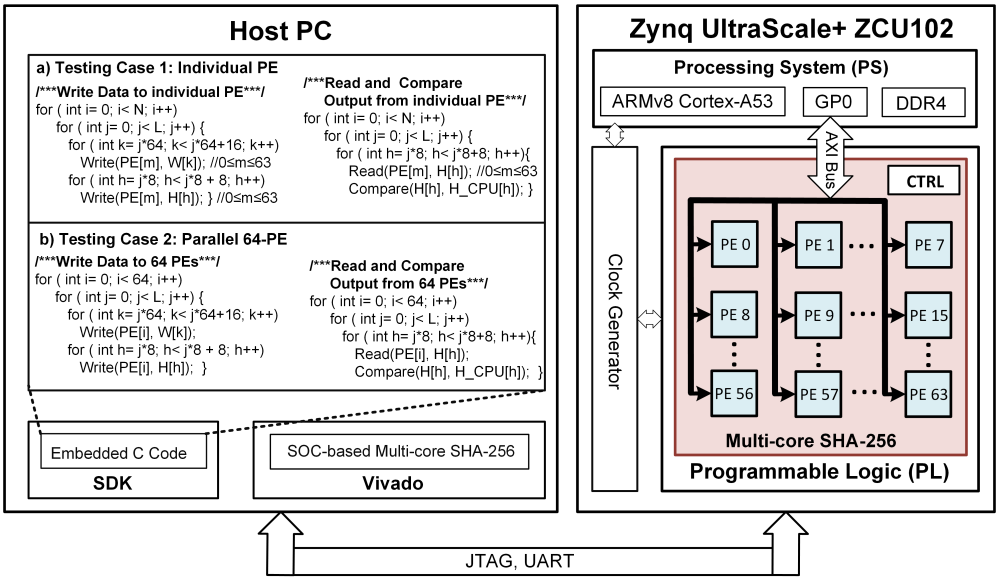

FIGURE 13. SoC-based multimem SHA-256 diagram

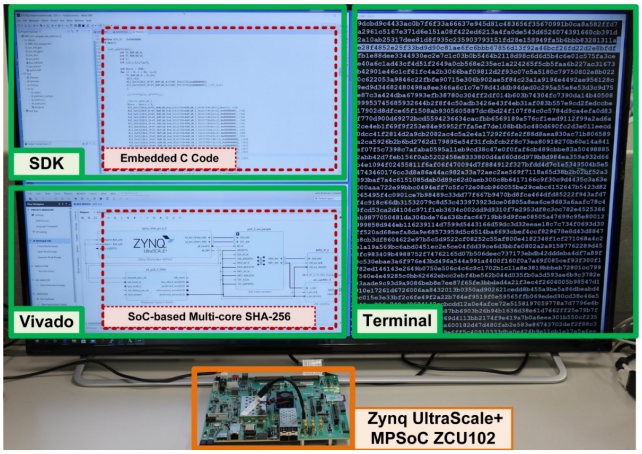

FIGURE 14. SoC-based multimem SHA-256 experiment with real devices

a maximum frequency of 249 MHz. The processing rate of our accelerator is 30.6 times, 2.4 times, 2 times, and 3.7 times better than the architectures in [19], [20], [6], and [21], respectively. The hardware efficiency of our accelerator version 1-PE is 1,864 Kbps/LUT, which is 12.4 times, 2.5 times, 2.5 times, and 1.76 times higher than the architectures in [19], [20], [6], and [21], respectively.

On the Virtex E device, the processing rate and hardware efficiency of the full circuit of our accelerator version 1-PE are 7.6 times (662 vs. 87 Mbps) and 2.7 times (183 vs. 69 Kbps/LUT) higher than those of the architecture in [22], respectively.

On the Virtex 4 device, compared to that of the architectures in [19], [11], [10], and [21], the performance of our accelerator version 1-PE is 34 times, 1.6 times, 2.3 times, and 3.5 times better in terms of processing rate and 13.5 times, 1.4 times, 1.3 times, and 1.6 times better in terms of hardware efficiency, respectively.

On the Virtex 5 device, the processing rate of our architecture version 1-PE is 28.5 times and 3.1 times better than those of the architectures in [19] and [21], respectively. Our circuit hardware efficiency is improved by 5.1 times and 1.1 times compared to that of the architectures in [19] and [21], respectively.

On the Virtex 6 device, the processing rate and hardware efficiency of the ALU part of our accelerator version 1-PE are 2 times and 2.5 times higher than those of the architecture in [25], respectively.

On the Virtex 7 device, the processing rate and hardware efficiency of our accelerator version 1-PE are 1.9 times and 5 times higher than those of the architecture in [26], respectively.

On the Kintex UltraScale+ device, our accelerator version 1-PE achieves almost the same performance as that of the work in [27]. However, our accelerator version 64-PE obtains a significantly better processing rate, 49 times that of the aforementioned study (284,448 vs. 5,829 Mbps).

Overall, our proposed SHA-256 accelerator obtains significantly better processing rate and hardware efficiency and has the ability to scale the trade-off between the processing rate and hardware cost. This means that our accelerator processing rate can be changed according to the requirements of the real system.

2) Functional Verification on a Real Hardware Platform

To prove that the circuit operates correctly not only in the software simulation tool but also on real hardware, we built a system on chip (SoC) platform to execute the proposed SHA-256 circuit including 64 PEs. The SoC platform overview

is shown in Fig. 13, and the real image of the platform is captured in Fig. 14.

The platform consists of two main devices: a host PC and a Zynq Ultrascale+ ZCU102 evaluation board. The two devices connect and exchange data via JTAG and UART cables. The ZCU102 board comprises an ARMv8 Cortex-A53 microprocessor, a programmable logic (PL), and a clock generator. Our multimem SHA-256 is developed in the PL part and controlled by the ARMv8 microprocessor. The clock generator generates a clock speed of 100 MHz to operate the ARMv8 microprocessor and our accelerator embedded in the PL.

The host PC runs two Xilinx tools: a Vivado and a Software Development Kit (SDK). We use Vivado to design and load the SHA-256 circuit onto the Zynq UltraScale+ ZCU102 board. To control the operation of the SHA-256 accelerator, we use the SDK to embed C code onto the ARMv8 microprocessor. The C code runs tasks such as transferring data between the external memory and the SHA-256 accelerator, activating the operation of the accelerator, and verifying the accelerator results.

In our experiment, we develop two testing cases named individual PE and parallel 64-PE, as shown in Fig. 13.

In the individual PE testing case, all 64 PEs of the accelerator are programmed to perform the same task. They are all programmed to compute the hash values for 100,000 independent data blocks. The results from each PE are then compared against the correct 100,000 hashing values computed by the Intel Xeon Gold 6144 CPU. Our implementation results have shown that all 64 PEs work correctly to generate hash values matched with those computed by the Xeon CPU. In other words, the correction ratio of our accelerator is 100%.

In the parallel 64-PE testing case, the 64 PEs of the accelerator are programmed to perform different tasks in parallel. The input data of different data blocks are loaded into different PEs. The results are then compared against the corresponding hashing outputs computed by the Intel Xeon Gold 6144 CPU. Again, the implementation results show that all 64 PEs are calculated correctly with a correction ratio of 100%.

V. CONCLUSION

The cryptography hash function SHA-256 is an important process in keeping the decentralized blockchain network secure and preserving data security and integrity in the smart systems of society 5.0. Developing a high-performance SHA-256 circuit compatible with SoC is thus a requirement. Unfortunately, state-of-art SHA-256 architectures have focused only on improving the performance of the SHA-256 core as a standing-alone circuit. The compatibility of the SHA-256 circuit with the SoC as well as the data transfer between the SHA-256 core and other parts of the SoC have not yet been studied. In this paper, we have solved the problem by developing an SoC-compatible SHA-256 accelerator. Several new techniques such as multiple local memory structures (multimem), pipelined ALU, and SBi-SBo have been introduced to achieve this purpose. Our experimental results on both FPGA and ASIC have shown that the proposed accelerator is not only compatible with the SoC via an AXI bus and provides an efficient data transfer mechanism but also achieves significantly better processing rate and hardware efficiency than previous works. The functional behavior of our accelerator has been proven to run correctly on a real SoC FPGA platform Zynq UltraScale+ ZCU102. The maximum processing rate of our accelerator in theory is 284 Gbps. Version 64-PE of our accelerator has been successfully laid out with 180 nm CMOS technology with a chip size of $8.5mm \times 8.5mm$, consumes a power of 1.86 W, and provides a maximum processing rate of 40.96 Gbps at 80 MHz frequency and 1.8 V voltage.

Obviously, the processing rate of an entire SoC embedding SHA-256 accelerator is much more important than that of the SHA-256 accelerator alone. This research has proposed an SoC-compatible SHA-256 accelerator and significantly improved the accelerator processing rate. However, the potential of the accelerator may not be sufficiently utilized due to the bottleneck data transfer rate provided by the SoC platform itself. Therefore, we believe that developing a new architecture and technique to overcome the data transfer rate bottleneck of SoC platforms will be an important research trend of the field in the near future.

ACKNOWLEDGMENTS

This research was supported by the Japan Science and Technology Agency (JST) PRESTO under grant number 2020A031.

REFERENCES

[1] U.S. Department of Commerce. (Jul. 2013). Digital signature standard (DSS). [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[2] National Institute of Standards and Technologies. U.S. Department of Commerce. (Jun. 2015). The Keyed-Hash Message Authentication Code (HMAC). [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf

[3] National Institute of Standards and Technologies. U.S. Department Commerce. (Jun. 2015). Recommendation for Random Number Generation Using Deterministic Random Bit Generators, 800-90a Rev. 1. doi:10.6028/NIST.SP.800-90Ar1.

[4] R. S. Kent, IP Authentication Header, document RFC 2404. [Online]. Available: https://www.ietf.org/rfc/rfc2402.txt

[5] 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," in IEEE Access, vol. 8, pp. 139634-139646, 2020, doi: 10.1109/ACCESS.2020.3012581.

[6] 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 IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI'06), March 2006, pp. 317–322.

[7] 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 Transactions on Reconfigurable Technology and Systems (TRETS), vol. 5, no. 1, pp. 1–28, 2012.

[8] L. Dadda, M. Macchetti, and J. Owen, "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)," in Proceedings Design, Automation and Test in Europe Conference and Exhibition, Feb 2004, pp. 70–75.

[9] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Cost-Efficient SHA Hardware Accelerators," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 8, pp. 999–1008, Aug 2008.

[10] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," in 2017 7th International Symposium on Embedded Computing and System Design (ISED), Dec 2017, pp. 1–4.

[11] Yimeng Chen, Shuguo Li, "A high-throughput hardware implementation of SHA-256 algorithm,"IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-4, 2020.

[12] National Technical Information Service. FIPS 180-2 – secure hash standard, U.S. Department of Commerce/NIST, Springfield, VA; 2002 <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>

[13] T. Ichikura, R. Yamano, Y. Kikutani, R. Zhang, and Y. Nakashima, "EMAXVR: A Programmable Accelerator Employing Near ALU Utilization to DSA," IEEE Symposium on Low-Power and HighS peed Chips, Coolchips2018, 2018.

[14] J. Iwamoto, Y. Kikutani, R. Zhang, and Y. Nakashima, "Daisy-Chained Systolic Array and Reconfigurable Memory Space for Narrow Memory Bandwidth," IEICE Transactions on Information and Systems, 2020, Volume E103.D, Issue 3, pages 578-589, DOI: 10.1587/transinf.2019EDP7144.

[15] D. Phan, T. H. Tran and Y. Nakashima: "SHA-256 Implementation on Coarse-Grained Reconfigurable Architecture", IEEE Symposium on Low-Power and High-Speed Chips 2020 (poster), Apr. 2020, Japan.

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

[17] Helion IP Core Products, Helion Technology, available at http://heliontech.com/core.htm

[18] A. Satoh and T. Inoue, "ASIC hardware focused comparison for hash functions MD5, RIPEMD-160, and SHS," International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, Las Vegas, NV, 2005, pp. 532-537 Vol. 1, DOI: 10.1109/ITCC.2005.92.

[19] R. Garcia, I. Algredo-Badillo, M. Morales-Sandoval, C. Feregrino-Uribe, and R. Cumplido, "A compact FPGA-based processor for the secure hash algorithm SHA-256," Computers and Electrical Engineering 40(1), pp. 194-202, 2014.

[20] I. Algredo-Badillo, C. Feregrino-Uribe, R. Cumplido and M. Morales-Sandoval, "Novel Hardware Architecture for Implementing the Inner Loop of the SHA-2 Algorithms," 2011 14th Euromicro Conference on Digital System Design, Oulu, 2011, pp. 543-549, doi: 10.1109/DSD.2011.75.

[21] M. M. Wong, V. Pudi and A. Chattopadhyay, "Lightweight and High Performance SHA-256 using Architectural Folding and 4-2 Adder Compressor," 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Verona, Italy, 2018, pp. 95-100, doi: 10.1109/VLSI-SoC.2018.8644825.

[22] Mooseop Kim, Jaecheol Ryou, and Sungik, "Efficient Hardware Architecture of SHA-256 Algorithm for Trusted Mobile Computing," Information Security and Cryptology: 4th International Conference, China, Jun. 2009, pp. 240-252, DOI:10.1007/978-3-642-01440-619

[23] Kurt K. TingSteve C. L. YuenK. H. LeePhilip H. W. Leong, "An FPGA Based SHA-256 Processor," International Conference on Field Programmable Logic and Applications, 2002, pp. 577-585.

[24] C. Jeong and Y. Kim, "Implementation of efficient SHA-256 hash algorithm for secure vehicle communication using FPGA," 2014 International SoC Design Conference (ISOCC), Jeju, 2014, pp. 224-225, doi: 10.1109/ISOCC.2014.7087617.

[25] M. D. Rote, Vijendran N and D. Selvakumar, "High performance SHA-2 core using the Round Pipelined Technique," 2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, 2015, pp. 1-6, doi: 10.1109/CONECCT.2015.7383912.

[26] M. Kammoun, M. Elleuchi, M. Abid, and M. S. BenSaleh "FPGA-based implementation of the SHA-256 hash algorithm," IEEE International Conference on Design and Test of Integrated Micro and Nano-Systems, pp. 1-6, 2020

[27] Raffaele Martino, Alessandro Cilardo, "A flexible framework for exploring, evaluating, and comparing SHA-2 designs," IEEE Access 7, pp. 72443-72456, 2019.