IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

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

Digital Object Identifier 10.1109/ACCESS.2021.3131558

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

VU TRUNG DUONG LE[1], (Graduate Student Member, IEEE),
THI HONG TRAN[2], (Member, IEEE),
HOAI LUAN PHAM[1], (Graduate Student Member, IEEE),
DUC KHAI LAM[3], AND YASUHIKO NAKASHIMA[1], (Senior Member, IEEE)

[1]Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma, Nara 630-0192, Japan
[2]Graduate School of Engineering, Osaka City University, Osaka 558-8585, Japan
[3]Computer Engineering Department, University of Information Technology, Vietnam National University, Ho Chi Minh City 700000, Vietnam

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

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

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

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

I. INTRODUCTION

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

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

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

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

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

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

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

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

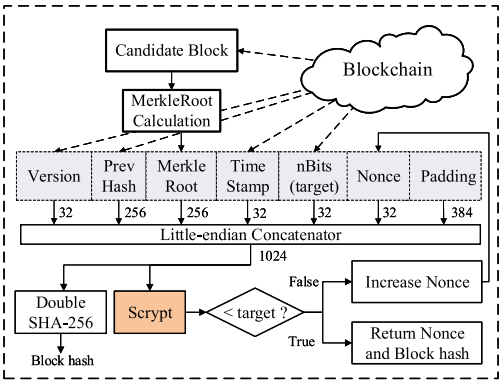

FIGURE 1. Scrypt proof-of-work (PoW) mining system architecture.

II. BACKGROUND

A. PROOF-OF-WORK

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

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

B. SCRYPT

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

TABLE 1. Fields of the block header.

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

Algorithm 1 Out = Scrypt (Blockheader)

Scrypt variables and parameters for cryptocurrency mining:

Block header (B_header) (1024 bits)
Block size factor (r) = 1
Parallelization parameter (p) = 1
CPU/memory cost parameter (N) = 1024
Length of DerivedKey in bits (dklen) = 256

Steps of the algorithm:

1: **P1 = PBKDF2(B_header, B_header, 1024 × r × p)**
2: P1 = LittleEndian32(P1)
3: **RM_out = ROMix(P1, N, r)**
4: RM_out = LittleEndian32(RM_out)
5: **Scrypt_out = PBKDF2(B_header, RM_out, dklen)**
6: Scrypt_out = LittleEndian32(Scrypt_out)
7: **return** Scrypt_out

1) PBKDF2

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

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

```
DK = ""
for i ← 1 to (dklen/256) do
    DK_i = HMAC(message, {salt, i})
    HMAC:
        IPAD = 36363636 ... 36_16 (256 bits)
        OPAD = 5C5C5C5C ... 5C_16 (256 bits)
        KHASH = SHA256(message)
        IXOR = {(KHASH ⊕ IPAD), IPAD}
        OXOR = {(KHASH ⊕ OPAD), OPAD}
        IHASH = SHA256({IXOR, salt, i})
        OHASH = SHA256({OXOR, IHASH})
        DK_i = OHASH
    DK = {DK, DK_i}
end for
return DK
```

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

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

SHA-256 includes three steps: padding, message expansion, and message compression. In the padding step, the message is divided into multiple 512-bit data blocks. The last block of the message is padded with a string of zeros as necessary, and the message length is expressed in bits. For each data block and the previous hash (or initial constants), message expansion and message compression are performed to compute an intermediate 256-bit hash. The hash result of the final block (final digest) is the hash value of the entire message. The reader is referred to [17]–[19] for a better understanding of SHA-256. In PBKDF2, SHA-256 is the most complex process that must be considered when optimizing the hardware.

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

```
for i ← 0 to (N-1) do
    Writing to memory:
    Mem_i = Block
    Block = BlockMix(Block, r)
end for
for j ← 0 to (N-1) do
    Reading from memory:
    j = Block[489:480]
    Block = BlockMix(Block ⊕ Mem_j, r)
end for
return RM_out = Block
```

Algorithm 4 BM_Out = BlockMix(Block, r)

```
X = Block[1023:512] (block's 512 high bits)
for i ← 0 to ((2 × r) − 1) do
    X = X ⊕ Block[511:0] (block's 512 low bits)
    X = X + Salsa20/8(X)
    if (i == 0) then
        OutH = X
    else
        OutL = X
    end if
end for
return BM_out = OutH, OutL
```

2) ROMix

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

the total Scrypt execution time because of the many memory writing and reading loops. Therefore, we propose the Multi ROMix architecture with the main purpose of accelerating the ROMix process.

3) BlockMix

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

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

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

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

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

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

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

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

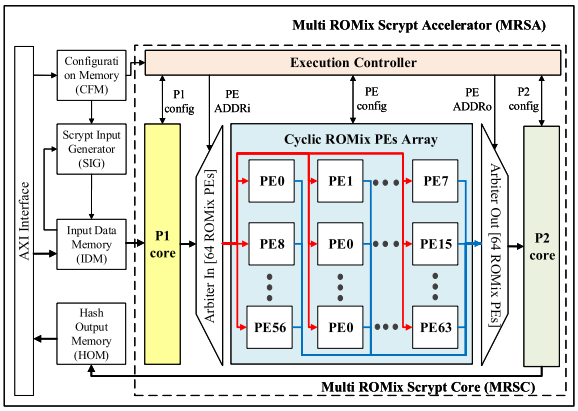

FIGURE 2. The MRSA hardware architecture.

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

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

III. THE PROPOSED MULTI ROMix SCRYPT ACCELERATOR (MRSA)

A. CONFIGURABLE ARCHITECTURE

Scrypt is a parameterizable ASIC-resistant algorithm. Therefore, each Scrypt application in cryptocurrency mining or

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

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

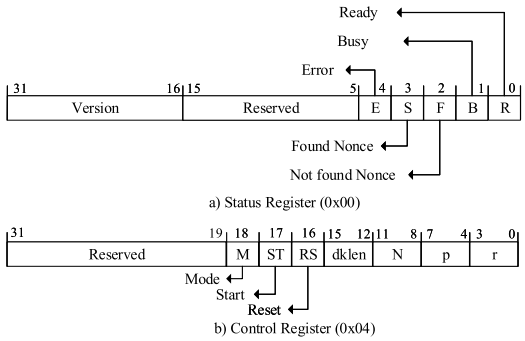

FIGURE 3. Diagrams of the status and control registers.

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

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

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

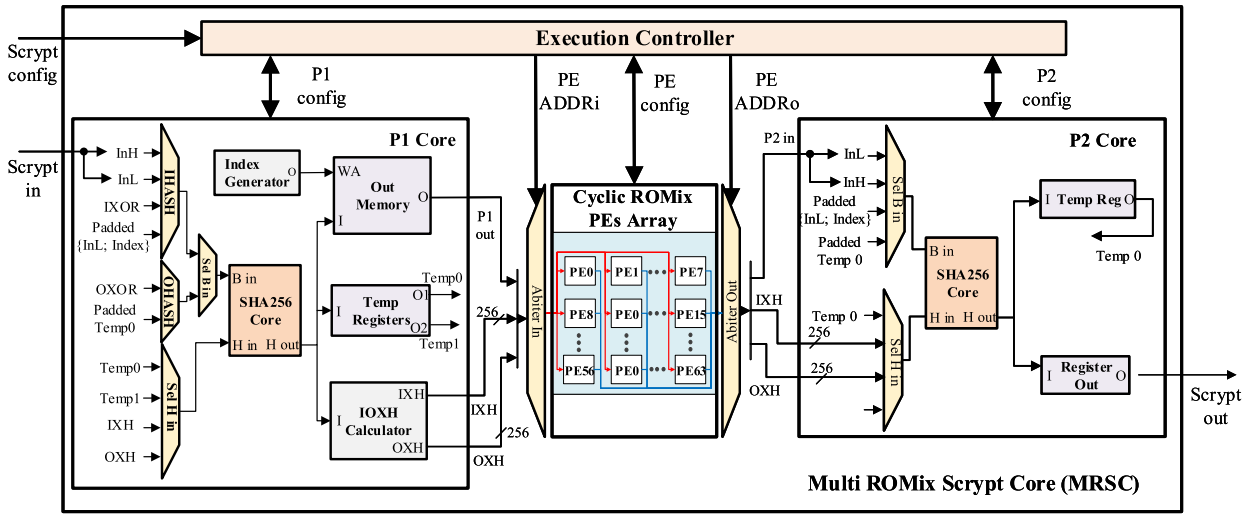

FIGURE 4. The MRSC hardware architecture.

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

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

B. MULTI ROMix SCRYPT CORE (MRSC)

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

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

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

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

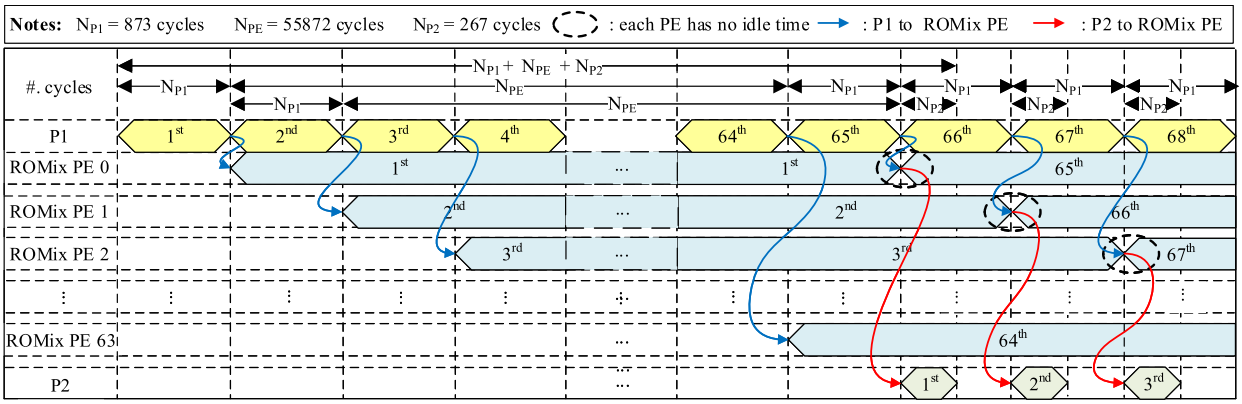

FIGURE 5. The timing chart of the MRSC.

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

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

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

C. RESCHEDULING TECHNIQUE

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

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

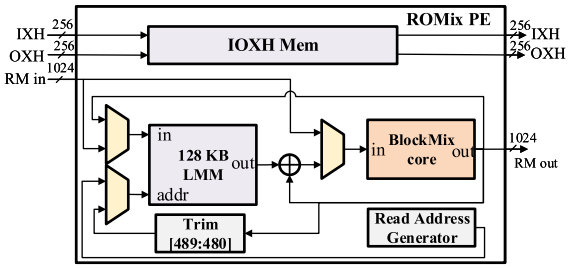

FIGURE 6. The ROMix PE hardware architecture.

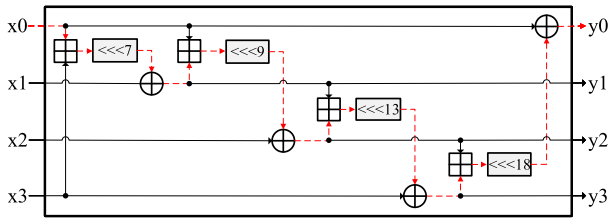

FIGURE 7. The critical path of a ROMix PE.

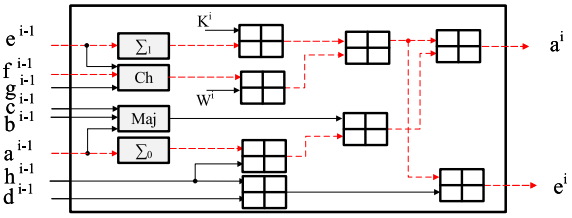

FIGURE 8. The critical path of the P1 and P2 Cores.

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

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

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

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

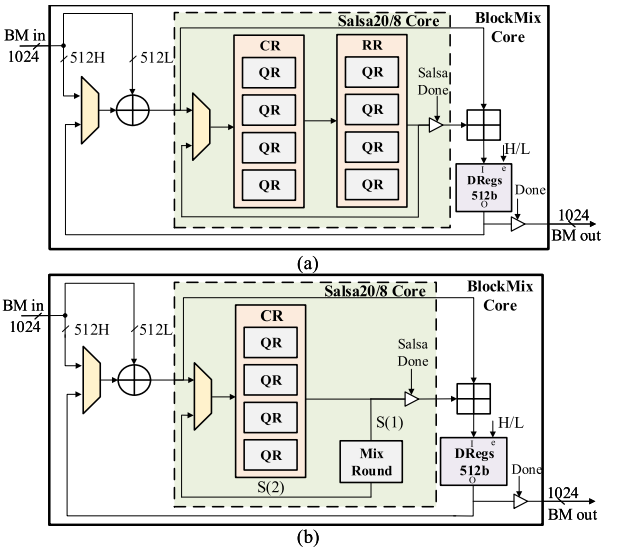

FIGURE 9. The BlockMix core hardware architecture: (a) the conventional BlockMix core; (b) the proposed BlockMix core.

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

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

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

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

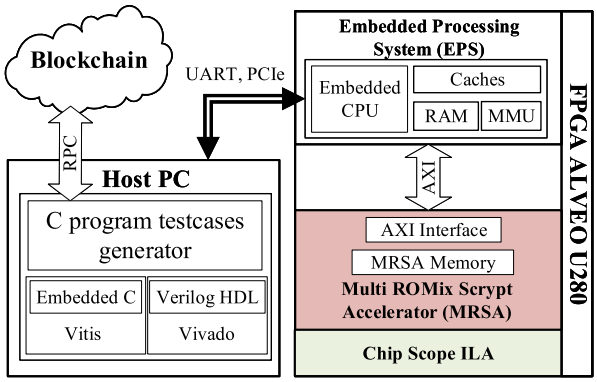

FIGURE 10. The embedded SoC on the Xilinx ALVEO U280 FPGA.

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

IV. EVALUATION AND EXPERIMENTAL RESULTS

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

A. MRSA IMPLEMENTATION AND VERIFICATION ON REAL HARDWARE

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

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

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

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

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

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

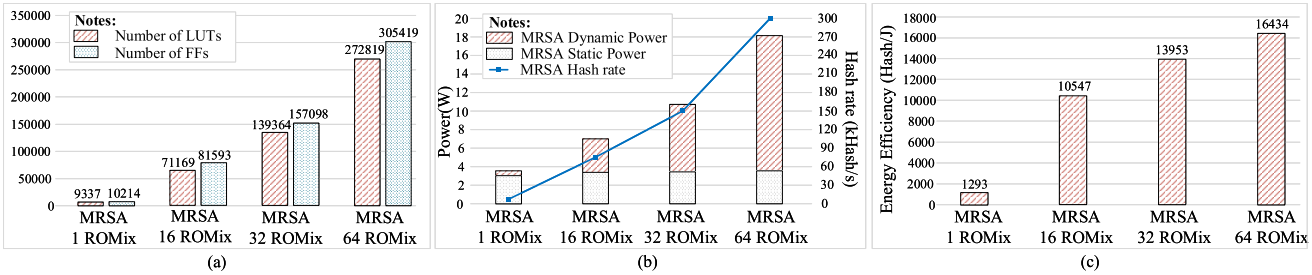

FIGURE 11. Graphs for the quantitative evaluation of the MRSA design on the Xilinx ALVEO U280 FPGA: (a) hardware resources; (b) power consumption and hash rate; (c) energy efficiency.

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

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

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

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

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

1) COMPARISON WITH RELATED FPGA-BASED WORKS

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

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

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

2) QUANTITATIVE EVALUATION ON DIFFERENT FPGAs AND MRSA VERSIONS

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

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

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

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

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

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

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

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

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

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

D. FLEXIBILITY ADVANTAGES OF MRSA

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

1) DYNAMIC CONFIGURATION

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

2) STATIC RECONFIGURATION

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

as well as enhance the circuit flexibility to adopt the future change.

V. CONCLUSION

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

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

APPENDIX

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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