

Implementing a very high-speed secure hash algorithm 3 accelerator based on PCI-express

Huu-Thuan Huynh, Tuan-Kiet Tran, Tan-Phat Dang

University of Science, Vietnam National University, Ho Chi Minh City, Vietnam

Article Info

Article history:

Received May 6, 2024

Revised Jul 25, 2024

Accepted Aug 12, 2024

Keywords:

Edge computing

Hardware accelerator

KECCAK

Peripheral component

interconnect express

Secure hash algorithm 3

ABSTRACT

In this paper, a high-performance secure hash algorithm 3 (SHA-3) is proposed to handle massive amounts of data for applications such as edge computing, medical image encryption, and blockchain networks. This work not only focuses on the SHA-3 core as in previous works but also addresses the bottleneck phenomenon caused by transfer rates. Our proposed SHA-3 architecture serves as the hardware accelerator for personal computers (PC) connected via a peripheral component interconnect express (PCIe), enhancing data transfer rates between the host PC and dedicated computation components like SHA-3. Additionally, the throughput of the SHA-3 core is enhanced based on two different proposals for the KECCAK- f algorithm: re-scheduled and sub-pipelined architectures. The multiple KECCAK- f is applied to maximize data transfer throughput. Configurable buffer in/out (BIO) is introduced to support all SHA-3 modes, which is suitable for devices that handle various hashing applications. The proposed SHA-3 architectures are implemented and tested on DE10-Pro supporting Stratix 10 - 1SX280HU2F50E1VG and PCIe, achieving a throughput of up to 35.55 Gbps and 43.12 Gbps for multiple-re-scheduled-KECCAK- f -based SHA-3 (MRS) and multiple-sub-pipelined-KECCAK- f -based SHA-3 (MSS), respectively.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Tan-Phat Dang

University of Science, Vietnam National University

Ho Chi Minh City, Vietnam

Email: dtphat@hcmus.edu.vn

1. INTRODUCTION

The increasing demand for massive amounts of real-time data transformations and processing necessitates high-performance data servers. This process acquires data from various sources, such as remote devices via the Internet, and then transmits it to dedicated hardware like graphic processing units (GPU) or hardware accelerators through burst or streaming mechanisms. peripheral component interconnect express (PCIe) has been utilized to enhance the data transfer rate to dedicated hardware [1], [2]. Operating on point-to-point topology, PCIe enables devices to communicate directly with other components without sharing bandwidth with other devices on the bus. PCIe utilizes multiple independent lanes, ranging from one lane to 32 lanes, for data transfer between the host and end device. Each lane comprises two pairs of differential signaling wires, one for transmitting data (Tx) and one for receiving data (Rx). Therefore, the PCIe operation speed can range from 2.5 GT/s to 32 GT/s for Gen1 to Gen5, respectively. Furthermore, the implementation of a direct memory access (DMA) for PCIe to eliminate a central processing unit (CPU) intervention has appeared in earlier works [1], [3], which involves transferring data from the main memory of the host device to a temporary DMA local

register before sending it to the address of the end device.

Regarding the dedicated computing hardware, the field programmable gate array (FPGA)-based hardware accelerators for cryptography have been attractive in research domains [4], [5], because of the increasing need for robust cryptographic algorithms to secure sensitive information and communications. Cryptographic hash functions play a fundamental role in ensuring integrity [6] and authenticity [7], [8]. In recent years, one such hash function that has garnered significant attention and adoption is the SHA-3. Standardized by the National Institute of Standards and Technology (NIST) in 2015, SHA-3 represents the latest iteration in the secure hash algorithm (SHA) family [9]. Unlike its predecessors, SHA-1, which has been susceptible to vulnerabilities and collision attacks [10], SHA-3 offers enhanced security properties and resistance to known cryptographic attacks. SHA-3 is designed to produce fixed-size hash values, or message digests, from input data of arbitrary length.

In high-performance applications, SHA-3 is utilized more and more frequently. For multimedia data, such as image encryption, the SHA-3 algorithm is employed to generate key streams from multiple blocks that are divided from the original images [11], [12]. In the security channel, the transmission of medical data and high-definition images between doctors and patients often necessitates hashing to prevent malicious modification, which is a crucial requirement in the medical field [13], [14]. Moreover, with the increase of internet of things (IoT) devices, the adoption of edge and fog computing has become increasingly common [15]. This trend has led to the emergence of high-performance devices optimized for processing speed, with a particular focus on security, including hash function algorithms [16], [17]. Consequently, there is a growing demand to enhance the performance of cryptographic algorithms to protect the vast amounts of data transmitted between these devices. On the other hand, hash functions like SHA-3 play a crucial role in blockchain technology, ensuring the integrity, security, and transparency of distributed ledger systems. The hash function helps maintain transaction integrity based on the Merkle tree structure [18]. Notably, miners are tasked with validating transactions under consensus mechanisms such as proof of work (PoW) [19]. To be eligible for rewards, miners must quickly generate nonce, underscoring the need for a high-performance hash function [20].

To enhance the performance of SHA-3, various research has been conducted, ranging from software optimizations on GPU to hardware accelerators [21]–[29]. The efficiency of implementing SHA-3 in a GPU environment has been demonstrated in [21]. Parallel Thread eXecution (PTX) is utilized to leverage the parallel permutation capabilities of the SPONGE construction and compute unified device architecture (CUDA) streams are employed to enable GPUs to receive and compute data simultaneously. Moreover, hardware accelerators for SHA-3 on FPGA are more attractive than implementing it in a GPU environment due to reduced technology dependence. A significant number of works aim to enhance the throughput and efficiency of SHA-3 through unrolling, pipelined, and sub-pipelined techniques and optimization of arithmetic cores (KECCAK- f) [23]–[29]. Unlike other works using the hardware description language (HDL), the work in [22] uses open computing language (OpenCL) to implement SHA-3 as a co-processor on FPGA to demonstrate the efficiency of the hardware implementation of SHA-3.

In this paper, we adopt an FPGA-based hardware design approach for implementing the SHA-3 algorithm using the Verilog language. This choice is motivated by the fact that SHA-3 computations mainly involve permutations using XOR, AND, and NOT gates, as well as inherent parallel processing capabilities. Unlike previous works [23]–[29] that primarily focus on the KECCAK- f function, we also address another core component of the SHA-3 algorithm, such as buffer in and out. Moreover, high-performance applications not only demand high-speed dedicated hardware but also require efficient data transmission. To address this requirement, we utilize PCIe, which enables high-throughput communication and leverages the computational power of the PC for data setup and management via software. The key contributions of our proposed methods are as:

- We present our SHA-3 design implemented on FPGA as a hardware accelerator for PC via PCIe links. DMA read and write is used to accelerate the data transfer rate without the observation of the CPU. In addition, ping-pong memory enables simultaneous computation and data transmission between the PC and our SHA-3 accelerator, thereby maximizing the parallel processing capabilities of SHA-3.
- To support various applications, we introduce configurable buffers that are flexible enough to switch between modes and minimize buffer usage for both input and output data while maintaining flexibility and efficiency.
- Multiple KECCAK- f are introduced to enhance maximum performance. Two architectures for KECCAK- f , including the re-scheduled and sub-pipelined architectures, are presented, contributing to overall performance enhancement in our SHA-3 design.

The remaining sections of the paper are organized as follows. Section 2 provides background information on SHA-3 algorithms. Our hardware design is comprehensively analyzed in section 3, covering the model of the SHA-3 accelerator to PC through PCIe, configurable buffers, and multiple KECCAK- f architecture. Evaluation and comparison of our results with other approaches are presented in section 4. Finally, section 5 concludes the paper.

2. SHA-3 PRELIMINARY

The construction of SHA-3 differs from the Merkle–Damgård design in SHA-1 and SHA-2, instead adopting the SPONGE construction [9], which is comprised of two main phases: absorbing and squeezing, as shown in Figure 1. Prior to the absorbing and squeezing phases, the input message m of arbitrary length undergoes a padding process. This ensures that the message is expanded to a multiple of r bits (1152, 1088, 832, or 576 bits) by appending the pattern "10*1". However, the SHA-3 hash function requires that the message m must append the suffix "01" to support domain separation [9]. Consequently, the pattern "0110*1" is appended to message m , as illustrated in Figure 1. During the absorbing phase, the padded message m is partitioned into several blocks of size r . Each r -sized block is then combined with a capacity c to form a 1600-bit block, which is subsequently processed sequentially by each KECCAK- f function until all blocks are processed. In the squeezing phase, the length of the output d (224, 256, 384, or 512 bits) can vary depending on the selected mode.

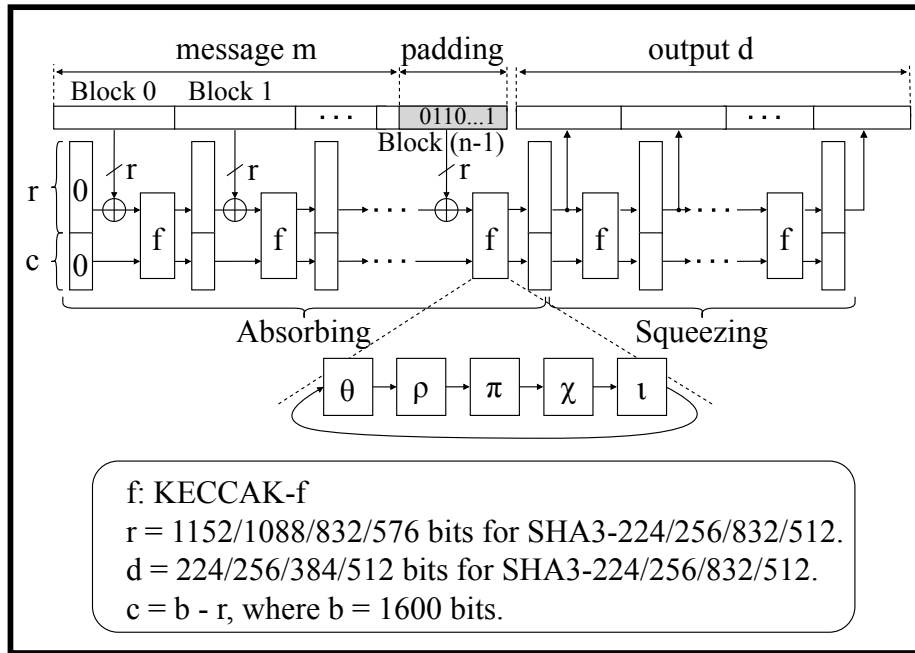


Figure 1. Padding and SPONGE construction

The KECCAK- f function is a fundamental component in SHA-3 and is utilized in both the absorbing and squeezing phases. The input message is converted into a three-dimensional array, denoted by x , y , and z , formatting a $5 \times 5 \times 64$ state array. The KECCAK- f function operates on this state array during 24 rounds of a round function (Rnd), with each round consisting of five step mappings: θ (theta), ρ (rho), π (pi), χ (chi), and ι (iota).

3. DESIGN AND IMPLEMENTATION

In this section, we provide an overview of the proposed SHA-3 architecture at the system level. We analyze the comprehensive data flow, showing the interaction with each component within the system. Next, we discuss the implementation of configurable buffers to support multiple modes. Lastly, our re-scheduled and sub-pipelined techniques are introduced in detail for the KECCAK- f function.

3.1. Overview architecture

System level overall architecture and data flow for SHA-3 accelerator described in Figure 2. The system architecture of the proposed SHA-3 accelerator is depicted in Figure 2(a). The PC serves as a host server, receiving requests from various remote devices and responding to the results. Requests related to the hash function are transmitted to the SHA-3 accelerator via PCIe. In this work, we utilize Intel intellectual property (IP) named Intel L/H-Tile Avalon-MM for PCIe on the DE10-Pro device [30]. Specifically, PCIe Gen 3x8 is employed, operating at a frequency of 250 MHz, allowing for a throughput of up to 63 Gbps. Additionally, this IP serves as a bridge, converting PCIe protocol to the Avalon bus. Before the SHA-3 accelerator begins operation, essential information, such as the size of the processed string, is transferred to its Control/Status Registers block. This is employed through the use of a base address register (BAR) with 32-bit non-prefetchable memory. To enable high-performance transmission, a DMA engine is employed along with separate read-and-write data modules. Additionally, two random-access memories (RAM) operate in a ping-pong manner for both input and output data. After the hash computation is completed, the hash values cannot immediately be sent to the PC; they must wait for a request from the PC. Therefore, a ping-pong RAM Out is utilized to temporarily store the hash values. The ping-pong way ensures that the dedicated hardware accelerator remains fully utilized, minimizing any idle time. Two clock domains are utilized in this system. The first clock operates at a frequency of 250 MHz for the PCIe IP, while the second clock operates at the frequency of the SHA-3 accelerator. This configuration optimizes the throughput of each domain to accelerate the entire system.

The proposed SHA-3 accelerator comprises three main components: padding, buffer including buffer in (BI) and buffer out (BO), and multiple KECCAK- f units. The padding and BI operations are executed concurrently. Once BI accumulates sufficient data serially from RAM In 0/1, the output of this buffer is parallelly combined with the data output from the padding unit through OR operation. This data is then fed into multiple KECCAK- f units, which process multiple data simultaneously. The hash value generated by multiple KECCAK- f units is transferred to BO in parallel. Subsequently, BO serially writes the data to RAM Out.

The data, comprising multiple short and long messages intended for SHA-3 processing, is stored in the system memory of the PC. Under CPU control using Terasic's PCIe driver, this data is continuously transferred from the system memory to the SHA-3 accelerator. In cases where the data size exceeds the capacity of RAM In 0/1, the ping-pong mechanism comes into play. As depicted in Figure 2(b), long data is initially transferred from the system memory to RAM In 0, and subsequently to RAM In 1. Once RAM In 0 is filled with Data 0, the SHA-3 computes the hash value and temporarily stores the results in RAM Out 0. Concurrently, the SHA-3 initiates processing Data 1 from RAM In 1. Once all results of Data 0 are available in RAM Out 0, they are read using DMA read and returned to the PC's system memory. The result of Data 1 is transferred from RAM Out 1 to the system memory, once DMA read for Data 0 is completed. Thus, the ping-pong approach for RAM In and RAM Out facilitates pipeline processing at the system level for increased performance.

Figure 2(c) illustrates three stages for data transfers and hashing computation. In stage 0, the PCIe link connected to the PC retrieves data from the system memory, operating at a speed of 63 Gbps according to Intel specifications [30]. Moving on to stage 1, the PCIe IP utilizes the Avalon-MM master to transfer data to RAM In according to the DMA technique. The DMA process supports burst transfers on a 256-bit interface width with a frequency of 250 MHz. While the throughput of DMA can reach up to 64 Gbps, the burst count is limited to 5 bits. Consequently, when the data size exceeds 8192 bits, the throughput of DMA becomes unstable. Our experiments show that the throughput of the DMA stage fluctuates between approximately 20 Gbps and 55 Gbps, thereby impacting the SHA-3 block. To mitigate this issue, ping-pong memory is employed for RAM In, where one memory receives data while the other provides data for computation. Hash computation is initiated only when one of the two memories is filled, ensuring that DMA does not affect stage 2. In addition, another crucial factor influencing DMA throughput is the size of each RAM In. Small sizes can lead to unstable DMA throughput, while large sizes may result in redundancy. Based on experimental results, a size of 10 KB for each RAM In 0/1 is chosen, as detailed in section 4. Stage 2 relies on the SHA-3 computation rate, which can run up to 43 Gbps.

The detailed SHA-3 architecture is shown in Figure 3, progressing from a coarse to a fine level of granularity. Specifically, the two primary components-the Buffer and Multiple KECCAK- f units, illustrated in Figure 3(a)-are explained in more depth in the subsequent subsections. Additionally, the two optimized KECCAK- f architectures, namely re-scheduled and sub-pipelined, which have the greatest impact on the throughput of the overall design, are depicted in Figure 3(b). Furthermore, a more detailed explanation of the θ and $(\rho - \pi - \chi - \iota)$ stages is provided in Figure 3(c), offering a deeper insight into the micro-architecture of the proposed

design, which will also be elaborated on in the subsequent subsection.

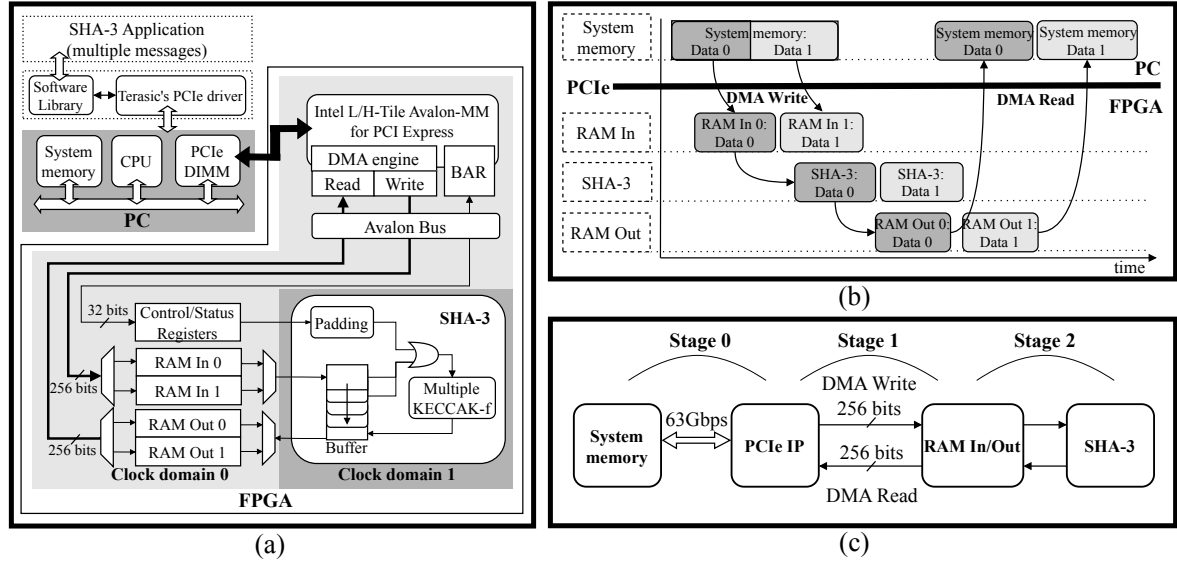


Figure 2. Overall architecture and data flow for SHA-3 accelerator on the system level (a) system architecture, (b) data flow on the system level, and (c) the three stages involve transferring data from the system memory to the SHA-3 accelerator

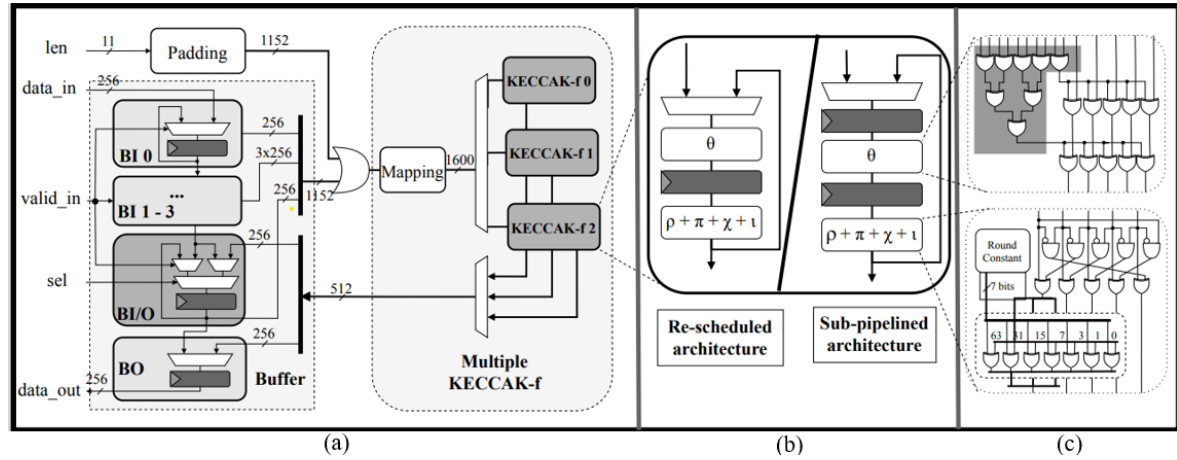


Figure 3. The proposed SHA-3 architecture in detail (a) the buffer and multiple KECCAK- f architectures and (b) the two Rnd architectures: re-scheduled and sub-pipelined ways, and (c) θ and $(\rho - \pi - \chi - \iota)$ architectures

3.2. Buffer

To facilitate flexibility in handling different modes, we introduce a configurable buffer capable of switching between BI and BO based on the selected mode. Each mode needs a distinct block size r , as illustrated in Figure 1. For SHA3-224 mode, with a data width of 256 bits, the input block size is 1152 bits, corresponding to five BIs. Similarly, for SHA3-256/384/512 modes, the required number of BIs is 5/4/3, respectively. Conversely, the output size for SHA3-224/256/384/512 modes in the 256-bit base is 1/1/2/2 BOs. To optimize the buffer utilization, we propose four BIs (BI 0 to BI 3), one BO, and one BIO, as depicted in Figure 3(a). The buffers cascade to each other, with the output of the preceding buffer serving as the input for the subsequent one. BIs only accept new input when the *valid_in* signal is activated; otherwise, they retain

the current value. BIO exhibits slightly more complexity than BI, as it can receive data from the preceding buffer when the *sel* signal is triggered; otherwise, it functions as a BO, receiving hash values from multiple KECCAK-*f* units. BO is responsible for retrieving hash values and serially pushing them to RAM Out. For SHA3-224/256/384/512 modes, it takes 1/1/2/2 clock cycles to complete writing data to RAM Out. To streamline complexity, the receiving process in BI takes 4/4/5/5 clock cycles for SHA3-224/256/384/512 modes, respectively. Each data loaded into BI requires one clock cycle. Therefore, if the modes do not provide sufficient data within those clock cycles, zero inputs are inserted. For example, in the case of SHA3-512 requiring three blocks of 256 bits, the subsequent two blocks consist of zeros.

3.3. Multiple KECCAK-*f*

The multiple KECCAK-*f* module comprises mapping and three KECCAK-*f* instances, as illustrated in Figure 3(a). The 1152-bit data from the preceding phase is fed into the Mapping block, which appends zeros to expand it to a 1600-bit data size. In our design, we opt for three KECCAK-*f* instances to reduce the interval of input data to 8 clock cycles. The output of each KECCAK-*f* instance is 512 bits in size, and depending on the selected mode, truncation is applied to the output data.

In this work, we introduce two architectures for KECCAK-*f*: the re-scheduled and sub-pipelined architectures, depicted in Figure 3(b). In a conventional architecture, the sequence of steps includes $\theta - \rho - \pi - \chi - \iota$, with a register placed at the end of the ι step to indicate the completion of one round [31]. Our re-scheduled architecture reorders these steps to $\rho - \pi - \chi - \iota - \theta$ by inserting a register between the θ and ρ steps. As a result, re-scheduled architecture requires 25 repetitions to complete the hash value, one more compared to the base architecture. During the first repetition, only the θ step is implemented, while the remaining repetitions execute all steps in the sequence of $\rho - \pi - \chi - \iota - \theta$. The re-scheduled architecture offers higher efficiency compared to the conventional architecture. This is proven via synthesis results on the Stratix 10 device, revealing that the re-scheduled architecture achieves a frequency of 336.36 MHz, surpassing the conventional architecture's frequency of 321.85 MHz by 4.31%. Moreover, the re-scheduled architecture utilizes fewer resources, with a reduction in adaptive logic module (ALM) utilization of 16.67% (4214 ALMs compared to 5057 ALMs in the conventional architecture).

Unlike previous works [28], [29], where the sub-pipelined technique typically inserts two registers: one between the π and χ steps or between the θ and ρ steps and another at the end of the ι step, our sub-pipelined architecture uses registers between the θ and ρ steps and another register before the θ step. This decision is based on the observation that the critical path of the θ step is greater than that of the ρ , π , χ , and ι steps. Specifically, the θ step requires at least four XOR gate levels to complete, while the remaining steps need only AND and two XOR gates, as shown in Figure 3(c). By isolating the θ step, we aim to improve the delay for KECCAK-*f*. However, adding the register in the round increases the number of clock cycles required, doubling it to 48 clock cycles. To mitigate this increase in clock cycles, our design is capable of handling two data simultaneously at two different stages. For example, if data 1 is processed in the θ stage, data 2 is processed in the $(\rho - \pi - \chi - \iota)$ stage. In the next clock cycle, data 1 moves to the $(\rho - \pi - \chi - \iota)$ stage while data 2 transitions to the θ stage. Thus, the average time to generate one hash value is reduced to 24 clock cycles. The advantage of our sub-pipelined architecture is that it increases the frequency while maintaining a fixed number of clock cycles at 24, thereby increasing throughput.

In both re-scheduled and sub-pipelined architectures, the five steps are consistently grouped into two parts: θ and $(\rho - \pi - \chi - \iota)$. The formulation of the θ step is optimized by combining $C[x]$ and $D[x]$, as indicated by the red area in the θ part of Figure 3(c), denoted as $CD[x]$ in (1). $CD[x]$ serves as the shared element, utilized by $A[x, y]$, and two levels of XOR operation are employed to reduce the delay for the θ step.

$$\begin{aligned}
 CD[x] &= A[x-1, 0] \oplus A[x-1, 1] \oplus A[x-1, 2] \oplus A[x-1, 3] \oplus A[x-1, 4] \oplus \\
 &\quad ROT(A[x+1, 0], 1) \oplus ROT(A[x+1, 1], 1) \oplus ROT(A[x+1, 2], 1) \oplus \\
 &\quad ROT(A[x+1, 3], 1) \oplus ROT(A[x+1, 4], 1) \\
 A[x, y] &= A[x, y] \oplus CD[x]
 \end{aligned} \tag{1}$$

The hardware implementation of $(\rho + \pi)$ steps utilizes a net connection, which requires no additional resources or delay, based on the combination of $(\rho + \pi)$ steps illustrated in [32]. Furthermore, the combination of $(\rho - \pi - \chi - \iota)$ steps is depicted in Figure 3(c). Unlike previous works [24], which utilized 64-bit RC, we have simplified this process by storing only the non-zero bits in RC. Therefore, only the bit positions 0, 1, 3, 7, 15, 31, and 63 are stored, effectively reducing resource usage.

4. EVALUATION AND COMPARISON

This section presents the performance evaluation of MRS and MSS on DE10-Pro, considering several factors that impact the throughput such as the size of each RAM In in the ping-pong way and the number of KECCAK- f instances. Furthermore, a comparison of KECCAK- f computation with previous works [23], [24] is conducted on Virtex 7 to indicate the advantages and limitations of our multiple-re-scheduled-based KECCAK- f (MRK) and multiple-sub-pipelined-based KECCAK- f (MSK) architectures.

4.1. Performance evaluation on DE10-Pro

The SHA-3 hardware accelerator, utilizing the DE10-Pro device, is connected to the PC (Intel® Core™ i5-10400 2.9 GHz) via PCIe Gen 3x8, as illustrated in Figure 2(a). This setup allows for functional testing and performance evaluations. On the PC side, C code manages the transfer of data to RAM In 0/1. Each DMA operation fills one RAM In slot, with subsequent transfers populating the remaining slots in a ping-pong way. Subsequently, the PC promptly reads hash values from RAM Out for comparison with the golden data to verify their accuracy.

The experimental results of the two proposed architectures, MRS and MSS, in relation to factors such as throughput, RAM size, and the number of KECCAK- f units across all modes, are presented in the line charts in Figure 4. These charts visually represent the optimal configurations, highlighting the correlation between these key factors and their impact on the overall performance of the SHA-3 accelerator. Specifically, Figure 4(a) illustrates the relationship between throughput and various RAM In sizes, while Figure 4(c) shows the relationship between throughput and the number of KECCAK- f units. Additionally, Figure 4(b) provides a detailed view of the data flow, contributing to the analysis of bottlenecks and serving as a basis for determining the optimal number of KECCAK- f units for an efficient configuration.

The rate at which data is supplied plays a crucial role in determining the performance of the hardware accelerator. Specifically, in our system, where we employ a ping-pong mechanism, the size of RAM In directly influences performance as mentioned in subsection 3.1. As depicted in Figure 4(a), the relationship between SHA-3 performance and RAM In size was examined across all modes for both MRS and MSS architectures. The MRS and MSS throughput experiences a notable increase within the range of 1 to 8 KB, followed by a gradual rise from 12 to 20 KB. Beyond this point, the throughput saturates for all modes in both MRS and MSS architectures. Consequently, a RAM In size of 20KB (the size of RAM In 0/1 is 10KB) was selected, making a balance between maximizing MRS and MSS throughput and minimizing resource utilization.

Our proposed SHA-3 architecture employs multiple KECCAK- f instances to enhance throughput. However, if too many KECCAK- f instances are utilized, a bottleneck phenomenon arises when the multiple KECCAK- f instances operate faster than the preceding parts, resulting in resource redundancy. Conversely, the architecture becomes inefficient when only a small number of KECCAK- f instances are used. The selection of KECCAK- f instances considers various factors, including preceding block architecture and the algorithmic characteristics of KECCAK- f . As illustrated in Figure 4(b), stage 0 (BI + padding) necessitates a maximum of nine clock cycles, including five cycles for processing data in the worst-case scenarios of SHA3-384/512 modes, three clock cycles for overhead, and one clock cycle for waiting for the ready signal from multiple KECCAK- f . Moreover, given that the KECCAK- f algorithm requires 24 repetitions for one digest value, stage 1 in Figure 4(b) must be completed within approximately eight clock cycles for optimal efficiency. Consequently, three KECCAK- f instances are chosen for the multiple KECCAK- f block. This relationship is further clarified in Figure 4(c), revealing the increasing throughput of the MRS and MSS architectures with one to three KECCAK- f instances. Beyond this range, however, the MRS and MSS throughput saturates with four KECCAK- f instances. Thus, the optimal number of KECCAK- f instances is determined to be three.

Our proposed architectures are evaluated based on throughput and efficiency. Throughput (TP), measured in Gbps, is calculated using (2), where #bit represents the number of bits of input data, Fmax denotes the maximum frequency obtained from synthesis results, and #clock indicates the number of clock cycles elapsed. Efficiency (Eff.), on the other hand, is determined by the ratio of throughput to the utilized resources, such as ALMs for Intel devices or slices for Xilinx devices. The (2) illustrates the calculation of efficiency based on throughput and resource utilization. The evaluation process involves the use of the Intel® Quartus® Prime Pro Edition Design Software Version 19.1 to obtain reports on frequency and resource utilization.

$$TP = \frac{\#bit \times Fmax}{\#clock} \quad (2)$$

$$Eff. = \frac{TP}{Area} \quad (3)$$

The throughput measurement results of the two architectures, MRS and MSS, on DE10-Pro are presented in Table 1. To determine the number of clock cycles (#clock), each architecture is equipped with a counter to record the elapsed clocks, starting immediately when the core begins operation and stopping upon completion of the process. The data size for throughput measurement of the MRS and MSS architectures is tested up to 32 KB. Collaborating with the data length and operating frequencies of MRS and MSS, which are 280 MHz and 380 MHz, respectively, we compute the throughput for each mode. Specifically, the throughput for MRS is 35.55 Gbps, 33.60 Gbps, 27.69 Gbps, and 19.23 Gbps for SHA3-224, SHA3-256, SHA3-384, and SHA3-512 modes, respectively. Similarly, for MSS, the throughput is 43.12 Gbps, 41.20 Gbps, 36.27 Gbps, and 25.11 Gbps for SHA3-224, SHA3-256, SHA3-384, and SHA3-512 modes, respectively. The resources utilized by MRS and MSS are obtained from the Quartus tool, with MSS utilizing 8273 ALMs and 8374 registers, while MSS utilizes 9485 ALMs and 12832 registers, respectively. As a result, the efficiency of MRS and MSS for all modes are as follows: 4.30 Mbps/ALM, 4.06 Mbps/ALM, 3.35 Mbps/ALM, and 2.32 Mbps/ALM for MRS, and 4.55 Mbps/ALM, 4.34 Mbps/ALM, 3.82 Mbps/ALM, and 2.65 Mbps/ALM for MSS, respectively.

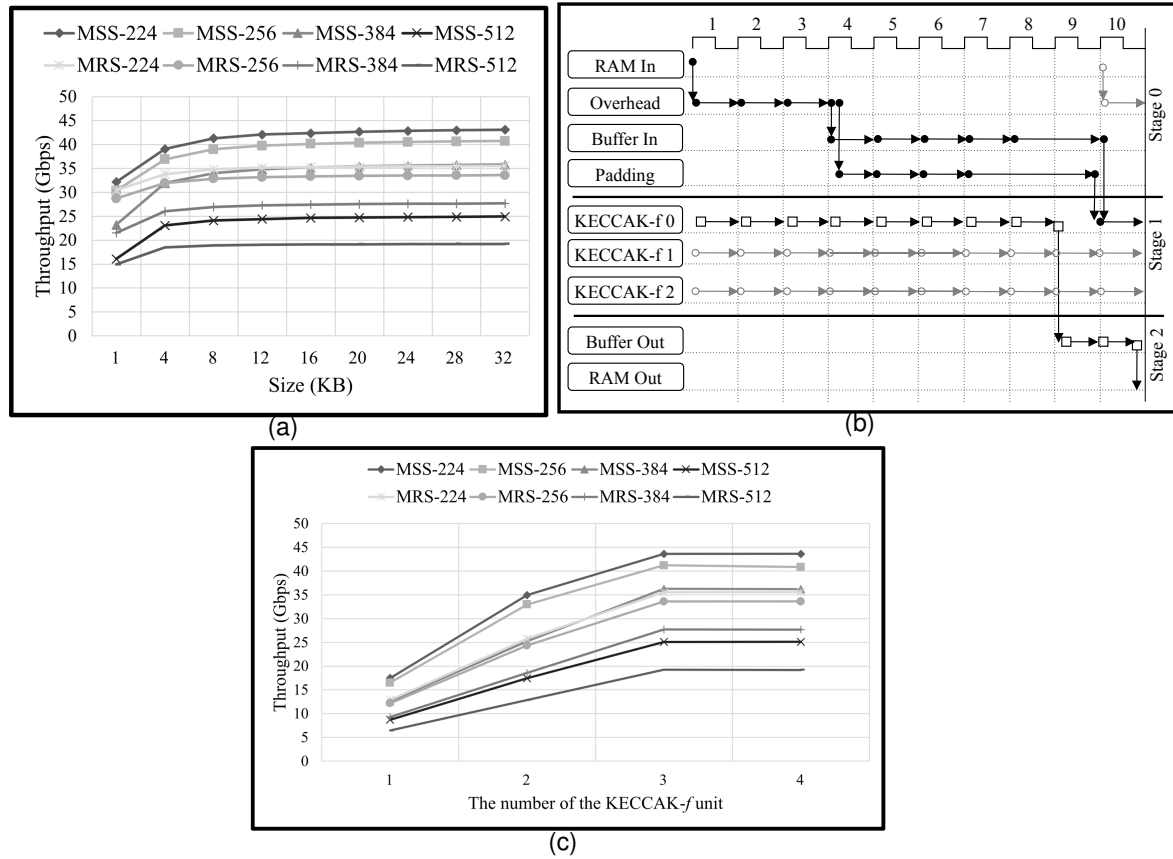


Figure 4. The experiment results of MRS and MSS across all modes in (a) the relationship between throughput and different RAM In sizes, (b) data flow timing chart of multiple KECCAK- f units, and (c) the relationship between throughput and the different numbers of KECCAK- f units

Table 1. The implementation results of the MRS and MSS architectures on DE10-Pro

Architecture	Freq. (MHz)	Area (ALM)	Reg.	TP (Gbps)				Eff. (Mbps/ALM)			
				224	256	384	512	224	256	384	512
MRS	280	8273	8374	35.55	33.60	27.69	19.23	4.30	4.06	3.35	2.32
MSS	380	9485	12832	43.12	41.20	36.27	25.11	4.55	4.34	3.82	2.65

4.2. Comparative analysis

For a fair evaluation and comparison between our proposed architectures (MRK and MSK) and previous ones [23], [26], we synthesize designs on Virtex-7 XC7VX485T using the Vivado 2020 tool. Since the previous works [23], [26] focused solely on the KECCAK- f computation, Table 2 displays the synthesis results of the KECCAK- f computation only. Given that all modes utilize the same KECCAK- f computation architecture, Table 2 only presents the results for the SHA3-512 mode for comparison. Additionally, it facilitates comparison with the proposal in [23] because the design only supports the SHA3-512 mode.

Table 2. The comparison of KECCAK- f computation architectures between our proposals and FPGA-based works on Virtex 7

Reference	[23], 2022	[26], 2023	Our proposed architecture	
Approach	Dual Rnd	Unrolling factor of 2	MRK	MSK
Fmax (MHz)	-	378.73	380.95	485.67
Area (Slice)	1521	1375	3203	2917
Register	-	-	4831	9669
#clock/hash	12	12	8	8
TP (Gbps)*	22.90	18.18	27.43	34.97
Eff. (Mbps/slice)*	15.11	13.22	8.56	11.99

*For SHA3-512 mode

Our MRK architecture requires 3203 slices and 4831 registers, operating at a maximum frequency of 380.95 MHz and achieving a throughput of 27.43 Gbps and an efficiency of 8.56 Mbps/slice. Conversely, the MSK architecture, aimed at reducing the critical path of Rnd, utilizes more registers than MRK (9669 > 4831). However, MSK outperforms MRK in terms of both throughput and efficiency, achieving 34.97 Gbps and 11.99 Mbps/slice, respectively.

Sravani and Durai [23] proposed the dual Rnd architecture, which utilizes one Rnd consisting of five steps ($\theta - \rho - \pi - \chi - \iota$) and registers cascading another Rnd and register to halve the number of clock cycles ($\#clock/hash = 12$), achieving a throughput of 22.90 Gbps. However, the throughputs of our two architectures, MRK and MSK, are 1.20 times (27.43 vs. 22.90) and 1.53 times (34.97 vs. 22.90) higher than that achieved by the dual Rnd architecture. While our architectures prioritize high performance, their efficiency is slightly lower compared to the dual architecture of Sravani and Durai [23] with MRK being 0.56 times (8.56 vs. 15.11) and MSK being 0.79 times (11.99 vs. 15.11). However, despite the lower efficiency, the throughput acceleration of our MSK architecture (53%) surpasses the efficiency acceleration of their dual Rnd architecture (26%).

When comparing our proposals with that of Sideris *et al.* [26], who implemented an unrolling factor of 2 to halve the number of clock cycles ($\#clock/hash = 12$), we observe significant improvements in throughput for both our MRK and MSK architectures. Specifically, our MRK architecture achieves a throughput 1.51 times higher (27.43 vs. 18.18), while our MSK architecture achieves a throughput 1.92 times higher (34.97 vs. 18.18) than the proposal of Sideris *et al.* [26]. However, despite these substantial throughput improvements, our efficiency is slightly lower, with MRK being 0.65 times lower (8.56 vs. 13.22) and MSK being 0.91 times lower (11.99 vs. 13.22), respectively. Nonetheless, this decrease in efficiency is not considered significant when compared to the notable throughput accelerations of 51% and 92% for MRK and MSK, respectively.

5. CONCLUSION

The demand for high-performance hash functions for modern applications has emerged, especially for the latest hashing version, SHA-3. The improvement of SHA-3 throughput is proposed in this paper. Specifically, full SHA-3 architecture is present from buffers and arithmetic core like KECCAK- f to integration at the system level. The proposed architectures are designed on an FPGA platform, which is connected to a PC via PCIe. PCIe boosts the data transfer rate, which is used popularly in modern applications. The issue of data transfer in PCIe's DMA is analyzed and resolved through the implementation of ping-pong memory and the selection of appropriate memory sizes. Furthermore, the configuration BIO is presented to support multiple SHA-3 modes and minimize the number of buffer instances. This feature benefits modern applications which require various output lengths of the hash values. The proposed architectures, like MRS and MSS, achieve a high throughput of up to 35.55 Gbps and 43.12 Gbps, respectively, thanks to the multiple KECCAK- f combined with one of the re-scheduled and sub-pipelined architectures. MSS demonstrates greater efficiency compared to MRS, for instance, with 4.55 Mbps/ALM > 4.30 Mbps/ALM for the SHA3-224 mode. In addition, our

MRK and MSK achieve 27.43 Gbps and 34.97 Gbps for SHA3-512 mode when implemented on Virtex 7, respectively.




REFERENCES

- [1] L. Rota, M. Caselle, S. Chilingaryan, A. Kopmann, and M. Weber, "A PCIe DMA architecture for multi-gigabyte per second data transmission," *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 972–976, 2015, doi: 10.1109/TNS.2015.2426877.
- [2] J. Liu, J. Wang, Y. Zhou, and F. Liu, "A cloud server oriented FPGA accelerator for lstm recurrent neural network," *IEEE Access*, vol. 7, pp. 122 408–122 418, 2019, doi: 10.1109/ACCESS.2019.2938234.
- [3] H. Kavianipour, S. Muschter, and C. Bohm, "High performance FPGA-based DMA interface for pcie," *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 745–749, 2014, doi: 10.1109/RTC.2012.6418352.
- [4] J.-S. Ng, J. Chen, K.-S. Chong, J. S. Chang, and B.-H. Gwee, "A highly secure fpga-based dual-hiding asynchronous-logic aes accelerator against side-channel attacks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 9, pp. 1144–1157, 2022, doi: 10.1109/TVLSI.2022.3175180.
- [5] M. Zeghid, H. Y. Ahmed, A. Chehri, and A. Sghaier, "Speed/area-efficient ECC processor implementation over gf (2 m) on FPGA via novel algorithm-architecture co-design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 8, pp. 1192–1203, 2023, doi: 10.1109/TVLSI.2023.3268999.
- [6] S. Shin and T. Kwon, "A privacy-preserving authentication, authorization, and key agreement scheme for wireless sensor networks in 5g-integrated internet of things," *IEEE access*, vol. 8, pp. 67 555–67 571, 2020, doi: 10.1109/ACCESS.2020.2985719.
- [7] S. Jiang, X. Zhu, and L. Wang, "An efficient anonymous batch authentication scheme based on hmac for vanets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2193–2204, 2016, doi: 10.1109/TITS.2016.2517603.
- [8] L. Zhou, C. Su, and K.-H. Yeh, "A lightweight cryptographic protocol with certificateless signature for the internet of things," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 3, pp. 1–10, 2019, doi: 10.1145/3301306.
- [9] Federal Information Processing Standards Publication, "SHA-3 standard: permutation-based hash and extendable-output functions," Aug. 2015, doi: 10.6028/NIST.FIPS.202.
- [10] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full sha-1," in *Advances in Cryptology—CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings*, Springer, 2017, pp. 570–596, doi: 10.1007/978-3-319-63688-7_19.
- [11] X. Zhang, Z. Zhou, and Y. Niu, "An image encryption method based on the feistel network and dynamic DNA encoding," *IEEE Photonics Journal*, vol. 10, no. 4, pp. 1–14, 2018, doi: 10.1109/JPHOT.2018.2859257.
- [12] C. Zhu and K. Sun, "Cryptanalyzing and improving a novel color image encryption algorithm using rt-enhanced chaotic tent maps," *IEEE Access*, vol. 6, pp. 18 759–18 770, 2018, doi: 10.1109/ACCESS.2018.2817600.
- [13] W.-K. Lee, R. C.-W. Phan, B.-M. Goi, L. Chen, X. Zhang, and N. N. Xiong, "Parallel and high speed hashing in GPU for telemedicine applications," *IEEE Access*, vol. 6, pp. 37 991–38 002, 2018, doi: 10.1109/ACCESS.2018.2849439.
- [14] M. Sravani and S. A. Durai, "Bio-hash secured hardware e-health record system," *IEEE Transactions on Biomedical Circuits and Systems*, 2023, doi: 10.1109/TBCAS.2023.3263177.
- [15] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, IoT, edge, and fog," *IEEE Access*, vol. 7, pp. 150 936–150 948, 2019, doi: 10.1109/ACCESS.2019.2947652.
- [16] T.-Y. Wu, Z. Lee, M. S. Obaidat, S. Kumari, S. Kumar, and C.-M. Chen, "An authenticated key exchange protocol for multi-server architecture in 5g networks," *IEEE Access*, vol. 8, pp. 28 096–28 108, 2020, doi: 10.1109/ACCESS.2020.2969986.
- [17] W.-K. Lee, K. Jang, G. Song, H. Kim, S. O. Hwang, and H. Seo, "Efficient implementation of lightweight hash functions on GPU and quantum computers for iot applications," *IEEE Access*, vol. 10, pp. 59 661–59 674, 2022, doi: 10.1109/ACCESS.2022.3179970.
- [18] Z. Liu, L. Ren, Y. Feng, S. Wang, and J. Wei, "Data integrity audit scheme based on quad merkle tree and blockchain," *IEEE Access*, 2023, doi: 10.1109/ACCESS.2023.3240066.
- [19] S. Islam, M. J. Islam, M. Hossain, S. Noor, K.-S. Kwak, and S. R. Islam, "A survey on consensus algorithms in blockchain-based applications: architecture, taxonomy, and operational issues," *IEEE Access*, 2023, doi: 10.1109/ACCESS.2023.3267047.
- [20] H. Cho, "Asic-resistance of multi-hash proof-of-work mechanisms for blockchain consensus protocols," *IEEE Access*, vol. 6, pp. 66 210–66 222, 2018, doi: 10.1109/ACCESS.2018.2878895.
- [21] H. Choi and S. C. Seo, "Fast implementation of sha-3 in GPU environment," *IEEE Access*, vol. 9, pp. 144 574–144 586, 2021, doi: 10.1109/ACCESS.2021.3122466.
- [22] H. Bensalem, Y. Blaqui re, and Y. Savaria, "An efficient opencl-based implementation of a SHA-3 co-processor on an fpga-centric platform," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 3, pp. 1144–1148, 2022, doi: 10.1109/TC-SII.2022.3223179.
- [23] M. M. Sravani and S. A. Durai, "On efficiency enhancement of SHA-3 for FPGA-based multimodal biometric authentication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 4, pp. 488–501, 2022, doi: 10.1109/TVLSI.2022.3148275.
- [24] S. El Moumni, M. Fettach, and A. Tragha, "High throughput implementation of sha3 hash algorithm on field programmable gate array (FPGA)," *Microelectronics journal*, vol. 93, p. 104615, 2019, doi: 10.1016/j.mejo.2019.104615.
- [25] B. Li, Y. Yan, Y. Wei, and H. Han, "Scalable and parallel optimization of the number theoretic transform based on FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023, doi: 10.1109/TVLSI.2023.3312423.
- [26] A. Sideris, T. Sanida, and M. Dasygenis, "Hardware acceleration design of the SHA-3 for high throughput and low area on FPGA," *Journal of Cryptographic Engineering*, pp. 1–13, 2023, doi: 10.1007/s13389-023-00334-0.
- [27] H. E. Michail, L. Ioannou, and A. G. Voyiatzis, "Pipelined SHA-3 implementations on FPGA: architecture and performance analysis," in *Proceedings of the Second Workshop on Cryptography and Security in Computing Systems*, 2015, pp. 13–18, doi: 10.1145/2694805.2694808.
- [28] G. S. Athanasiou, G.-P. Makkas, and G. Theodoridis, "High throughput pipelined FPGA implementation of the new SHA-3 cryptographic hash algorithm," in *2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP)*. IEEE, 2014, pp. 538–541, doi: 10.1109/ISCCSP.2014.6877931.




- [29] M. M. Wong, J. Haj-Yahya, S. Sau, and A. Chattopadhyay, "A new high throughput and area efficient SHA-3 implementation," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351649.
- [30] Intel, *L-Tile and H-Tile Avalon® Memory-Mapped Intel® FPGA IP for PCI Express* User Guide (version 23.4)*, 2024, 2024. [Online]. Available: <https://www.intel.com/content/www/us/en/docs/programmable/683667/23-4/introduction.html>, (accessed Apr. 10).
- [31] H. Mestiri and I. Barraj, "High-speed hardware architecture based on error detection for keccak," *Micromachines*, vol. 14, no. 6, p. 1129, 2023, doi: 10.3390/mi14061129.
- [32] A. Arshad, D.-e.-S. Kundi, and A. Aziz, "Compact implementation of SHA3-512 on FPGA," in *2014 Conference on Information Assurance and Cyber Security (CIACS)*, 2014, pp. 29–33, doi: 10.1109/CIACS.2014.6861327.

BIOGRAPHIES OF AUTHORS






Huu-Thuan Huynh    received the B.S., M.S., and Ph.D. degrees in radio physics and electronics from the University of Science, Ho Chi Minh City (HCMUS), in 1997, 2001, and 2010, respectively. Since 2006, he has been with the Faculty of Electronics and Telecommunications (FETEL), HCMUS. His current research interests are SoC FPGA-based real-time digital signal processing. He can be contacted at email: hhthuan@hcmus.edu.vn.



Tuan-Kiet Tran    received a bachelor of science (B.Sc.) degree and a master of science (M.S.) degree in electronics telecommunications engineering and electronics engineering, awarded by the University of Science, Ho Chi Minh City (HCMUS) in 2017 and 2019, respectively. He currently serves as a faculty member at the Faculty of Electronics and Telecommunications (FETEL) at HCMUS, Vietnam. His research interests include hardware accelerators for cryptography and parallel processing in AI. He can be contacted at email: trtkiet@hcmus.edu.vn.



Tan-Phat Dang    received a bachelor of science (B.Sc.) degree and a master of science (M.S.) degree in electronics telecommunications engineering and electronics engineering, awarded by the University of Science, Ho Chi Minh City (HCMUS) in 2018 and 2023, respectively. Presently, he is an active member of the Faculty of Electronics and Telecommunications (FETEL) at HCMUS, Vietnam. His primary focus lies in FPGA-based hardware accelerators for cryptography, digital signal processing, and video compressing. He can be contacted at email: dtphat@hcmus.edu.vn.