CHIMERA: Cryptographic Hardware for Integrated Multipurpose Engine on RISC-V with ASCON

CHIMERA: Cryptographic Hardware for Integrated Multipurpose Engine on RISC-V with ASCON

Alessandra Dolmeta *Student member, IEEE*, Valeria Piscopo *Student member, IEEE*,
Maurizio Martina *Senior member, IEEE*, Guido Masera *Senior member, IEEE*
Department of Electronics and Telecommunications, Politecnico di Torino, Torino, Italia
name.surname@polito.it

***Abstract*—In the context of the growing Internet-of-Things (IoT) ecosystem, where security and privacy concerns are critical due to the limited resources of connected devices, lightweight cryptography plays a vital role. ASCON, a lightweight cryptographic algorithm designed for constrained environments, offers robust security mechanisms such as Authenticated Encryption with Associated Data (AEAD), hashing, Message Authentication Code (MAC) generation and Pseudorandom Functions (PRF). In this work, we introduce CHIMERA, an Application Specific Instruction Set Processor (ASIP) architecture tailored to efficiently compute the ASCON algorithm on 32-bit RISC processors. The ASIP interfaces with the RISC-V core via the Core-V eXtension Interface (CV-X-IF), a novel communication mechanism. CHIMERA functions as a multipurpose coprocessor, supporting AEAD (ASCON-128, ASCON-128a) and hashing (Hash, Hasha). We present two versions of CHIMERA: the Complete Round (CR) version, a tightly coupled accelerator that delivers high performance at a higher hardware cost, and the Bitwise Rotation Unit (BRU) version, an Instruction Set Extension (ISE) offering lower efficiency but minimal area requirements. The design has been implemented on both Zynq Ultrascale+ FPGA and ASIC platforms, with results comparing the two versions and evaluating their performance relative to the state-of-the-art.**

***Index Terms*—Hardware accelerator, RISC-V, Lightweight Cryptography, ASCON, Hash functions and message authentication codes, FPGA.**

I. Introduction

Hardware-based cryptographic accelerators are crucial in various domains, including secure communication for embedded systems, high-performance computing environments like data centers, and secure IoT networks [1]. Lightweight cryptography (LWC) has been introduced to address the unique challenges posed by resource-constrained devices in the rapidly expanding Internet of Things (IoT) and embedded systems. These devices often have limited computational power, memory, energy, and communication capabilities. LWC provides secure communication and data protection in such environments while minimizing resource consumption. The National Institute of Standards and Technology (NIST) initiated the LWC competition, focused on the standardization of algorithms able to support Authenticated Encryption with Associated Data (AEAD) and hashing, specifically for resource-constrained devices. In 2023, the ASCON suite of algorithms [2] won the LWC competition. The sponge-based structure built upon the same lightweight permutation results in provably secure algorithms, which can also be straightforwardly implemented in hardware.

In this paper, we introduce CHIMERA, a RISC-V tightly coupled engine designed to accelerate the ASCON lightweight cryptographic algorithm.
In Marshall et al. [3], guidelines are presented for designing RISC-V instruction set extensions (ISEs) to ensure simplicity, efficiency, and security. These guidelines emphasize the use of general-purpose scalar registers for operands and results, adherence to simple building-block operations with two source registers and one destination register, avoidance of special-purpose architectural or micro-architectural state, and support for data-oblivious execution to prevent timing attacks. By following these principles, ISEs can balance utility, standardization, compatibility, latency, and area efficiency. We adopt these guidelines in our work, proposing two solutions, depending on the design constraints: the Complete Round (CR) version and the Bitwise Rotation Unit (BRU) version. Despite their differences, both versions are integrated in the same way leveraging the innovative Core-V eXtension Interface (CV-X-IF) [4]. This new approach is currently under standardization in the RISC-V community and enables the seamless integration of tightly coupled accelerators with the RISC-V core, preserving the advantages of tightly coupled accelerators while simplifying the overall design process. The CR version is a tightly coupled accelerator designed to minimize memory access by utilizing a dedicated register file, enhancing efficiency at the cost of increased hardware complexity. Conversely, the BRU version is an ISE optimized for extremely low area usage, focusing on accelerating bitwise rotation operations with minimal hardware, though at the expense of overall performance efficiency. Both versions are compared to state-of-the-art implementations, demonstrating superior efficiency in terms of performance. Notably, the BRU version offers a significantly reduced area footprint, making it particularly suited for resource-constrained environments such as IoT devices. The implementations are available open-source. [1]

This work was funded by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. This work received funding from the Key Digital Technologies Joint Undertaking (KDT-JU) under grant agreement No 101095947.

[1] https://github.com/vlsi-lab/CHIMERA.git

The paper is organized as follows: Sec. II provides background on the working principles of ASCON, while Sec. III presents an overview of the existing implementations. Section IV describes the two different hardware implementations we have developed, and the adopted integration method. Results are presented in Section V. Section VI concludes the paper.

II. ASCON

Developed in 2014 by researchers at Graz University of Technology, Infineon Technologies, and Radboud University [2], ASCON is a family of lightweight authenticated ciphers. It was selected in 2023 as the new standard for lightweight cryptography in the NIST Lightweight Cryptography competition, which ran from 2019 to 2023 and included ten finalist algorithms. Previously, it had also been the primary choice for lightweight authenticated encryption within the CAESAR competition's final portfolio (2014–2019). ASCON is an authenticated encryption algorithm with support for associated data (AEAD), employing a duplex sponge construction. This sponge mechanism enables ASCON to generate an output that is derived from the entire input string, achieved by continually updating a state variable with inputs such as the key, plaintext, and associated data. The encryption process in ASCON involves a series of permutation operations, which are structured in iteration blocks that rely on specific, predefined functions. The initial ASCON submission [2] encompasses two AEAD variants, namely ASCON-128 and ASCON-128a, and two hash functions, ASCON-Hash and ASCON-Hasha, also in the extendable output version, ASCON-XOF and ASCON-XOFa. The ASCON family has been then expanded by also adding Message Authentication Codes (ASCON-MAC) and Pseudorandom Functions (ASCON-PRF, ASCON PRFshort) [5]. The *a* variants (e.g., ASCON-128a, ASCON-Hasha) offer higher throughput by increasing the amount of data processed per step, making them more suitable for performance-critical scenarios. The key parameters are outlined in Table I.

TABLE I: Parameters of the ASCON cipher suite.

		Bit size		Perm. rounds	
Variant	**Algorithm**	**Key *K***	**Rate *r***	p^a	p^b
AEAD	ASCON-128	128	64	12	6
	ASCON-128a	128	128	12	8
	ASCON-80pq	160	64	12	6
Hash	ASCON-HASH	-	64	12	12
	ASCON-HASHa	-	64	12	8
XOF	ASCON-XOF	-	64	12	12
	ASCON-XOFa	-	64	12	8
MAC	ASCON-MAC	128	256/128	12	-
PRF	ASCON-PRF	128	256/128	12	-
	ASCON-PRFshort	128	128	12	-

The size of the cryptographic key K ensures security for AEAD and MAC modes. The rate r specifies the bits processed per permutation step. The number of rounds p^a and p^b define the permutations for associated data and plaintext/ciphertext, ensuring diffusion and security. [2]

ASCON encryption and decryption mechanisms utilize multiple parameters: secret key (K) of k bits, associated data (A), public message number (denoted by nonce, N), Initialization Vector (IV) to encrypt a plaintext (P), authentication tag T, and ciphertext (C). Encryption and decryption can be represented by the following formulae:

$$\mathcal{E}_{a,b,k,r}(K,N,A,P) = (C,T)$$
$$\mathcal{D}_{a,b,k,r}(K,N,A,C) = (P,T)$$

Figures 1 and 2 illustrate ASCON schemes of operations for encryption and decryption.

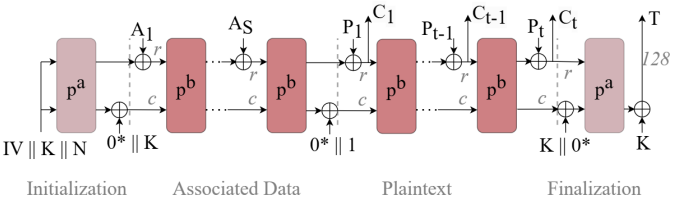

Fig. 1: ASCON Encryption $\mathcal{E}$ scheme. 0* indicates a bitstring of variable length, all 0s; ‖ indicates concatenation.

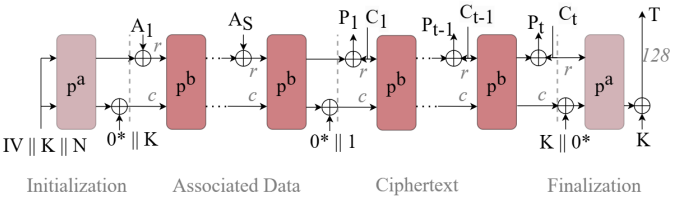

Fig. 2: ASCON Decryption $\mathcal{D}$ scheme.

The ASCON hash scheme is shown in Fig. 3. It is constructed on a sponge mode of operation, and it is composed of three steps: initialization, absorption, and squeezing. It can be considered a simplified version of authenticated encryption, but since there is no plaintext, the hashing continues until the length of the generated hash text reaches 256-bit.

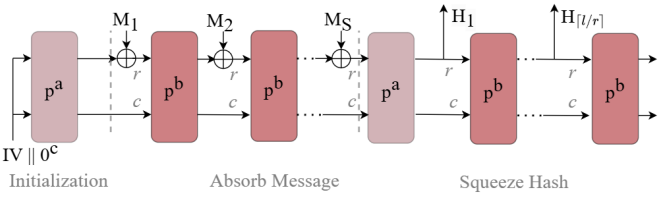

Fig. 3: ASCON Hash scheme.

The permutations in both p^a and p^b involve three in-order stages for each permutation round:

- **constant additions**: an 8-bit constant c_r is summed in each round to the internal-state second word, as in Eq. 1 ($\oplus$: XOR).

$$x_2 \leftarrow x_2 \oplus c_r \tag{1}$$

- **substitution**: 64 parallel operations of the five-bit S-box are performed, as described in Fig. 4. The S-box can be implemented using simple logical operations. This is why is so convenient to be implemented in hardware.
- **linear diffusion**: provides diffusion within each of the 64-bit words, as in Eq. 2 ($\ggg$: shift toward right).

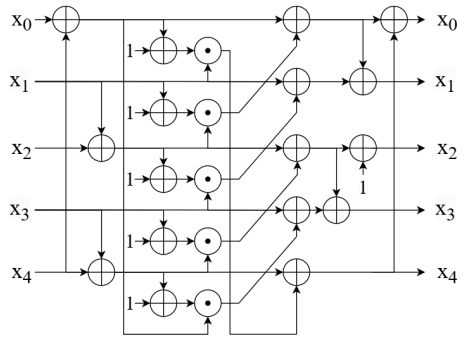

Fig. 4: Substitution layer (S-Box). · indicates bitwise AND.

$$
\begin{aligned}
x_0 &\leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\
x_1 &\leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\
x_2 &\leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6) \\
x_3 &\leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\
x_4 &\leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)
\end{aligned} \tag{2}
$$

Small IoT devices often have constrained resources (e.g., limited processing power, memory, and energy), making certain operations within permutations performance-critical. Nonlinear operations are computationally intensive and can become a bottleneck, especially when dealing with a large internal state that must be frequently stored and retrieved from memory (e.g., ASCON permutations manipulate a 320-bit structure). To address these bottlenecks, the following hardware approaches can be adopted:

- **State Register Bank**: use on-chip registers to store the state and minimize memory access delays, significantly improving efficiency during the full permutation process. This technique is used in the Complete Round (CR) version.
- **Dedicated S-box Modules**: implement substitution layers as combinational logic to reduce latency, optimizing the bitwise rotation operations performed in the permutation. This leads to the version labeled as the Bitwise Rotation Unit (BRU).

III. State-of-the-Art ASCON Hardware Implementations

Various literary sources explore the implementation of ASCON in FPGA and ASIC technologies. Most of the works implement a single instance of ASCON and do not support all its instances. [6], [7], and [8] are stand-alone implementations, i.e. external accelerators. [6] analyzes the trade-off between various important performance parameters, combining multiple rounds. While achieving high throughput, this design limits the applicability to ASCON-hash only. [7] optimizes ASCON for different typical applications to fully explore its design space. It also implements threshold implementation (TI) schemes to protect the hardware against side-channel attacks. [8] presents a round-based implementation of ASCON-128, characterized by a high throughput-to-area ratio.
[9] and [10] are standalone implementations, which, however, include interfaces and connections in the design. [9] introduces a flexible, reconfigurable crypto-processor, which supports both authenticated cipher and hash functions of ASCON and that can be connected to external devices with an I/O interface. [10] presents ASCON IP, with two asynchronous FIFOs, a Register File, a Control and Interrupt Generator block, and two interfaces, i.e. AMBA High-performance Bus (AHB) and Advanced Peripheral Bus (APB), which are responsible for the I/O communication with the rest of the system.
[11], [12], and [13] are ASCON implementations integrated into RISC-V. This makes these works the most similar to ours. [11] proposes custom extensions for several LWC algorithms, including ASCON. [12] introduces *Xascon*, with two custom instructions, and evaluates it with software-only and Zbkb-accelerated implementations. The study focuses on the Rocket core, which features five pipeline stages and floating point instructions, and is thus more complex than typical IoT processors. In both studies, the proposed ISEs are compliant with the requirements discussed in [3]. [13] implements ASCON-p as an instruction extension for RISC-V, tightly coupled to the processor register file without any dedicated registers. This approach limits the IP versatility because both the control unit and the internal register file are modified to fulfill the load/store requirements of the p-block. It relies on tightly integrating the core with dedicated resources, such as 10 general-purpose registers for state storage. While this design deviates from the ISE principles outlined by [3], it offers a compelling balance between performance and flexibility, resembling a tightly coupled accelerator rather than a traditional ISE.

IV. Hardware Implementation

Subsect. IV-A details the implementation and internal structures of CHIMERA, explaining the difference between the two approaches. Subsect. IV-B describes the CV-X-IF interface and how CHIMERA is integrated into the overall system.

A. Design

We distinguish the two proposed solutions as tightly coupled accelerator and Instruction Set Extensions, respectively, based on their alignment with RISC-V design principles and their integration approach.

CR Version. The Complete Round (CR) version, depicted in Fig. 5a, operates on ASCON's 320-bit state using a dedicated state register file. This design minimizes memory access by storing intermediate states directly in the custom register file, enhancing performance but increasing hardware complexity. While this approach does not fully align with the ISE definition in [3], as it uses a dedicated register file rather than the scalar register file for operand storage, it adheres to broader RISC-V design principles. Specifically, register file operations involve two source registers and one destination register, maintaining simplicity. Additionally, no special-purpose architectural or micro-architectural states are

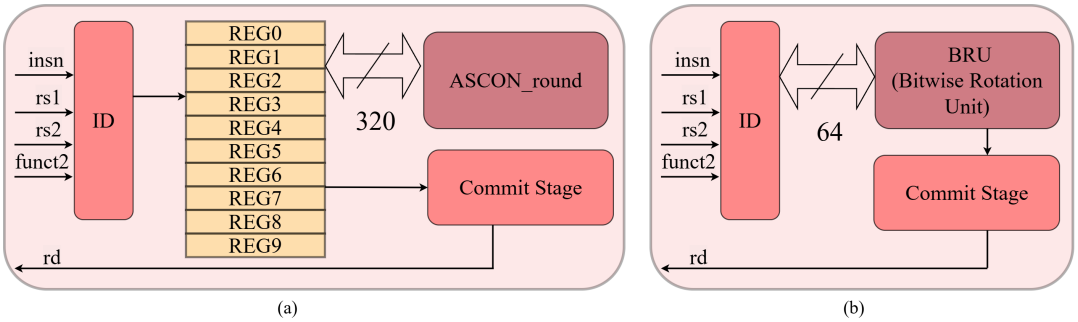

Fig. 5: CHIMERA: (a) CR version, (b) BRU version.

introduced, ensuring modularity within the RISC-V ecosystem. To support this implementation, the CR version introduces new custom instructions. A dedicated instruction is used to initiate the permutation, while additional instructions enable efficient loading and storing of state values in the custom register file. This approach streamlines communication with the accelerator, ensuring seamless operation within the microcontroller environment.

BRU Version. The BRU version, shown in Fig.5b, is a compact ISE specifically optimized for resource-constrained environments such as IoT devices. It focuses on accelerating the bitwise rotation operations within ASCON's Linear Diffusion Layer, offering an extremely low area footprint. Unlike the CR version, the BRU implementation aligns perfectly with RISC-V ISE requirements, as it uses the scalar register file for operand storage and adheres to the standard two-source and one-destination register format. This version introduces six custom instructions tailored to optimize the rotation operations. Five instructions implement the same function but with fixed shifting constants, which are hardcoded to save area by avoiding parameter passing during execution. A sixth instruction is used to read the 32 most significant bits (MSB) of the BRU result, as the 64-bit output is split into two separate 32-bit reads to align with the 32-bit architecture of the RISC-V processor. This modular approach allows efficient integration of the BRU accelerator while maintaining flexibility and compatibility with the microcontroller.

B. Integration

CHIMERA has been integrated into X-HEEP (eXtendable Heterogeneous Energy-Efficient Platform), an open-source RISC-V microcontroller described in System Verilog [14]. The system is depicted in Fig.6.

The CV-X-IF interface bridges loosely and tightly coupled accelerators in RISC-V systems, allowing direct connectivity to the CPU core without altering the core's pipeline or toolchain. This dedicated interface supports efficient offloading of instructions to various accelerators, using structured signals to manage both the instruction dispatch and data write-back seamlessly. Originally introduced in the CV32E40X core [15], CV-X-IF has since been integrated into the CV32E40P core variant (CV32E40PX, [16]), where a dispatcher directs essential signals to the interface. The CV-X-IF decoder identifies custom instructions, activates the accelerator, and ensures data flow between the core and the accelerator via dedicated signals. Once processing is complete, valid results are written back to the core's register file. Further information on the protocol of this interface can be found in the original repository[2].

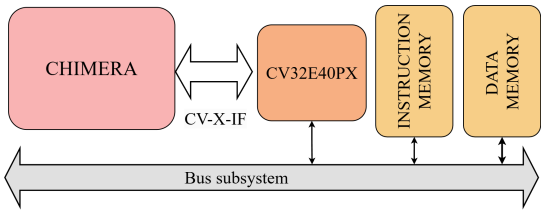

Fig. 6: CHIMERA integration with CV-X-IF.

V. Results and Comparisons

The module is written in System Verilog and implemented in ASIC (CMOS 65 nm) and FPGA (Zynq Ultrascale + ZCU104, device XCZU7EV-2FFVC1156) technologies.
Subsect. V-A reports our results and the comparison between CR and BRU implementations, while Subsect. V-B shows comparisons with the state-of-the-art. Speed-up and Improvement values of Tables II and III are evaluated as the ratio between Original Values and CR/BRU ones.

A. Our work

Table II reports the performance results in terms of clock cycles for the Hash, XOF, and AEAD ASCON applications. The Original column refers to the original software-only applications, whereas the CR and BRU columns contain the results for the two optimized implementations.
The CR version consistently outperforms the baseline, with speed-ups ranging from 2.91× to 6.04×, particularly excelling in algorithms like `asconhashav12` (5.79×) and `asconxofav12` (5.74×), due to its efficient use of the State Register Bank that reduces memory access. In contrast, the BRU version shows more modest speed-ups, between 1.41× and 1.56×, with the highest achieved by `asconxofav12` (1.56×). The lower speed-up of the BRU version is a result

[2]https://github.com/openhwgroup/core-v-xif/tree/main

of its focus on optimizing bitwise rotations rather than full permutation acceleration, offering lower area requirements at the cost of performance.

TABLE II: Performance and Speed-up Evaluation for ASCON algorithms (128B plaintext for AEAD schemes, 32B output message for Hash/XOF schemes).

Algorithm		Clock Cycles			Speed-up	
		Original	CR	BRU	CR	BRU
asconhashav12		13,164	2,273	8,533	5.79x	1.54x
asconhashv12		15,774	2,613	11,057	6.04x	1.43x
asconxofav12		12,070	2,102	7,751	5.74x	1.56x
asconxofv12		14,136	2,416	10,007	5.85x	1.41x
ascon128av12	enc	18,067	6,072	12,705	2.98x	1.42x
	dec	18,380	6,318	12,934	2.91x	1.42x
ascon128v12	enc	22,393	6,971	15,806	3.21x	1.42x
	dec	22,744	7,118	16,062	3.20x	1.42x
ascon80pqv12	enc	22,490	6,972	15,883	3.22x	1.42x
	dec	22,748	7,217	16,068	3.15x	1.42x

Table III compares the code size of ASCON algorithms in their original version and with two optimized implementations, CR and BRU, all compiled with the GCC `-O2` flag. Both optimizations significantly reduce code size, with the CR version achieving improvements of up to 2.35× (e.g., `asconhashv12`), while the BRU version follows closely with improvements of up to 2.31×. The differences between CR and BRU are minimal, reflecting their shared use of optimized custom instructions, while CR's dedicated State Register Bank slightly influences the code footprint.

TABLE III: Code-size and Improvement Evaluation for ASCON algorithms [Bytes] (128B plaintext for AEAD schemes, 32B output message for Hash/XOF schemes).

Algorithm	Code Size			Improvement	
	Original	CR	BRU	CR	BRU
asconhashav12	21,890	13,382	13,528	1.64x	1.62x
asconhashv12	31,474	13,378	13,600	2.35x	2.31x
asconxofav12	21,978	13,458	13,608	1.63x	1.62x
asconxofv12	31,574	13,450	13,676	2.35x	2.31x
ascon128av12	32,996	16,548	16,368	1.99x	2.02x
ascon128v12	28,072	15,646	15,464	1.79x	1.82x
ascon80pqv12	28,068	15,742	15,480	1.78x	1.81x

ASIC synthesis was performed using Synopsys Design Vision with CMOS 65nm technology, at 285 MHz. While the full HEEP system was not evaluated, the core and two coprocessors, BRU and CR, were analyzed. The core's area is approximately 33.8 kGE, with BRU and CR adding 4.15 kGE and 10.73 kGE, increasing the core area by 12.3% and 31.8%, respectively.

The FPGA synthesis and implementation are performed using Xilinx Vivado, which has a global clock of 300 MHz. The system's top-level frequency is set to 50 MHz, so binary applications can be loaded via JTAG and executed using GDB.

Table IV highlights the area differences between the CR and BRU versions of the SoC regarding LUTs and Registers. The CR version shows higher area utilization, particularly due to its dedicated Register File and more complex CHIMERA implementation, which includes additional logic for memory reduction. In contrast, the BRU version is designed for minimal area, achieving lower utilization in LUTs and Registers.

TABLE IV: Resource utilization for CR and BRU versions (ZCU104, 50 MHz).

Component	LUT	Registers
SoC (**CR-version**)	31,252	27,841
◇ Chimera Wrapper	1,321	810
○ XIF Controller	408	123
○ Chimera	913	687
○ Ascon	913	335
○ Register File	0	320
SoC (**BRU-version**)	30,518	27,170
◇ Chimera Wrapper	573	109
○ ID-Stage	272	71
○ Chimera	0	32
○ Commit-Stage	302	1

Notably, the CHIMERA Wrapper in the BRU version is significantly smaller, as its internal structure focuses on efficient bitwise operations without the added overhead of a dedicated Register File.

A final comparison between the CR and BRU approaches is summarised in Table V. The *Throughput/Area* column suggests that the two versions are comparable, with CR performing better than BRU for the Hash schemes and vice-versa for the AEAD ones. However, it should be noted that this metric is obtained by dividing the throughput by the number of LUTs, without taking into account the number of registers, which is 8 times higher for CR.

TABLE V: Design comparison. ⋆ 128 B plaintext encryption throughput.

Design	Functionality		FPGA Resources	Throughput [Mb/s]	Thr./Area [Mb/(s·LUTs)]
	AEAD	Hashing			
CR	128		LUTs: 1321	7.34⋆	0.00556
		Hash	FFs: 810	19.59	0.01483
BRU	128		LUTs: 573	3.24⋆	0.00565⋆
		Hash	FFs: 109	1.16	0.00223

B. Comparisons

Our work focuses on integrating ASCON into RISC-V, with a specific emphasis on efficiency and versatility in IoT-focused applications. Therefore, as mentioned in Section III, we compare our work with [11], [12], and [13] (Table VII).

Area. Considering the relative area increase, the accelerator in [13] expands the simple core by approximately 10.31% (from 45.6 kGE to 50.3 kGE). In [12], the synthesis of *Xascon* on an ST 28nm library increases the core size by 23.7% in total area (μm^2) and by 12% when measured in equivalent NAND2 gates. Similarly, [11] reports a 28.2% area increase when synthesizing on a Kintex-7 FPGA. In comparison, our BRU and CR coprocessors result in area increases of 12.3% and 31.8% respectively for the ASIC synthesis, and 9% and 20.8% for the FPGA implementation. While the BRU demonstrates a competitive area efficiency, the CR shows a higher relative area increase compared to other works.

Speed-up. As can seen in Table VII, [13] achieves the best performance by storing the 320-bit ASCON state in 10 general-purpose registers, allowing concurrent processing of multiple values with a toggle logic to switch between the registers and the accelerator. While similar to our CR version,

TABLE VI: Design comparison.
⋆ 128 B plaintext encryption throughput.

Design	**Functionality**		**FPGA**			**Throughput**	**Thr./Area**
	AEAD	Hashing	Device	Frequency [MHz]	Resources	[Mb/s]	[Kb/(s·LUTs)]
[11]	128		Kintex-7	50	LUTs:931	3.05⋆	32.8⋆
OURS - BRU	128		Zynq Ultrascale+	50	LUTs: 573 FFs: 109	3.24⋆	56.5⋆
	128a					4.03⋆	70.3⋆
	80pq					3.22⋆	56.3⋆
		Hash				1.16	20.2
		Hasha				1.50	26.2
		XOF				1.28	22.3
		XOFa				1.65	28.8

TABLE VII: Performance and Speed-up Evaluation for ASCON Algorithms. † 128 B plaintext. ‡ 64 B output message.

Reference	Algorithm	Clock Cycles		Speed-up
		Original	ISE	
[11]†	`aead_encrypt`	43,005	16,775	(2.56×)
	`aead_decrypt`	43,414	17,159	(2.53×)
[12]	`Hash/XOF`	1,610	799	(2.16×)
[13]‡	`Hash`	19,642	295	(66.5×)

this approach does not comply with RISC-V ISE requirements, as it bypasses the traditional load/store mechanism and does not respect the two-register source constraint. In contrast, our design adheres to ISE compliance, introducing the overhead of loading two registers at a time into the register file to maintain simplicity and modularity within the RISC-V framework.
In terms of speed-up, our CR version significantly outperforms both the BRU version and other implementations ([11] and [12]), as shown in Table II and Table V. The CR design achieves a speed-up ranging from 5.74 to 6.04 across various ASCON operations, demonstrating a considerable performance boost. The BRU version, while providing more modest improvements (speed-ups ranging from 1.41 to 1.56), still offers reasonable performance gains.

In Table VI, we conduct a fair comparison of throughput and area between our design and the one in [11]. Although only the ASCON-128 scheme is reported, this work is the most similar to our BRU version. In this comparison, our design excels with a throughput-to-area ratio of 56.5, compared to their 32.8, demonstrating the efficiency of our design across both performance and area considerations.

VI. Conclusions

We propose two design approaches for a lightweight ASCON implementation. The CR version provides higher speed-ups due to its use of state registers, while the BRU approach offers a compact ISE, striking a balance between performance and resource efficiency with respect to prior works. By targeting IoT processors with limited complexity, our implementation provides a versatile and practical solution for lightweight cryptographic applications. Future work may involve further optimizing throughput and extending support for additional cryptographic primitives.

References

[1] J. Xie, W. Zhao, H. Lee, D. B. Roy, and X. Zhang, "Hardware Circuits and Systems Design for Post-Quantum Cryptography—A Tutorial Brief," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 3, pp. 1670–1676, 2024.

[2] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," *Journal of Cryptology*, vol. 34, no. 33, 2021.

[3] "The design of scalar AES instruction set extensions for RISC-V," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, p. 109–136, Dec. 2020.

[4] "CV-X-IF eXtension Interface." https://docs.openhwgroup.org/projects/openhw-group-core-v-xif/en/latest/x_ext.html. Accessed: April 3, 2024.

[5] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon PRF, MAC, and short-input MAC." Cryptology ePrint Archive, Paper 2021/1574, 2021.

[6] S. Khan, W.-K. Lee, A. Karmakar, J. M. B. Mera, A. Majeed, and S. O. Hwang, "Area-time Efficient Implementation of NIST Lightweight Hash Functions Targeting IoT Applications." Cryptology ePrint Archive, Paper 2022/1716, 2022.

[7] H. Gross, E. Wenger, C. Dobraunig, and C. Ehrenhöfer, "Ascon hardware implementations and side-channel evaluation," *Microprocessors and Microsystems*, vol. 52, pp. 470–479, 2017.

[8] K. Raj and S. Bodapati, "FPGA Based Light Weight Encryption of Medical Data for IoMT Devices using ASCON Cipher," in *2022 IEEE International Symposium on Smart Electronic Systems (iSES)*, pp. 196–201, 2022.

[9] X. Wei, M. El-Hadedy, S. Mosanu, Z. Zhu, W.-M. Hwu, and X. Guo, "RECO-HCON: A High-Throughput Reconfigurable Compact ASCON Processor for Trusted IoT," in *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, pp. 1–6, 2022.

[10] G. S. Athanasiou, D. Boufeas, and E. Konstantopoulou, "A Robust ASCON Cryptographic Coprocessor for Secure IoT Applications," in *2024 Panhellenic Conference on Electronics & Telecommunications (PACET)*, pp. 1–6, 2024.

[11] H. Cheng, J. Großschädl, B. Marshall, D. Page, and T. Pham, "RISC-V Instruction Set Extensions for Lightweight Symmetric Cryptography." Cryptology ePrint Archive, Paper 2022/1697, 2022.

[12] C. Gewehr, L. Luza, and F. G. Moraes, "Hardware Acceleration of Crystals-Kyber in Low-Complexity Embedded Systems With RISC-V Instruction Set Extensions," *IEEE Access*, vol. 12, pp. 94477–94495, 2024.

[13] S. Steinegger and R. Primas, "A Fast and Compact RISC-V Accelerator for Ascon and Friends," in *Smart Card Research and Advanced Applications: 19th International Conference, CARDIS 2020, Virtual Event, November 18–19, 2020, Revised Selected Papers*, (Berlin, Heidelberg), p. 53–67, Springer-Verlag, 2020.

[14] S. Machetti, P. D. Schiavone, T. C. Müller, M. Peón-Quirós, and D. Atienza, "X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller for the Exploration of Ultra-Low-Power Edge Accelerators," 2024.

[15] O. Group, *CV32E40X User Manual*, 2024. Accessed: 2024-06-11.

[16] ESL-EPFL, "cv32e40px: RISC-V Core with Extensions for Cryptography and Machine Learning." https://github.com/esl-epfl/cv32e40px, 2024. Accessed: 2024-10-11.