RECO-HCON: A High-Throughput Reconfigurable Compact ASCON Processor for Trusted IoT

Xiangdong Wei*¶, Mohamed El-Hadedy†‡¶, Sergiu Mosanu§, Zhengping Zhu*, Wen-Mei Hwu‡, Xinfei Guo*‖
*University of Michigan – Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University
†Department of Electrical and Computer Engineering, California State Polytechnic University, Pomona
‡Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
§Department of Electrical and Computer Engineering, University of Virginia
{xiangdong.wei, xinfei.guo}@sjtu.edu.cn, hadedy@illinois.edu, mealy@cpp.edu
¶Both authors contributed equally to this work. ‖Corresponding author.

***Abstract*—Statistics show that in 2030 the number of connected IoT devices will reach 25.44 billion, which can lead to the security breach in the back-end of high-performance computing clusters connected with the same network. Unfortunately, the current security primitives are not suitable algorithms to be implemented on physically constrained devices designed for IoT. Thus, the National Institute of Standard and Technology has announced a worldwide lightweight cryptographic competition (LWC) for securing tiny devices. This paper introduces a flexible, reconfigurable, and energy-efficient crypto-processor running one of the LWC finalist candidates - ASCON, which uses sponge construction that has fewer memory accesses that leads to less power consumption compared to other ones. The proposed processor is reconfigurable in a way both authenticated cipher (Encryption/decryption processes) and hash functions of ASCON are implemented in a six-mode compact fashion, covering a diversity of applications in the IoT spectrum. The design is developed in Chisel and evaluated in 28/32nm technology with commercial EDA tools. Evaluation results show that the proposed processor achieves the highest throughput while consuming 29% less power, operating at over 667 MHz. The design has also been implemented in Skywater 130nm technology node with the latest released OpenLane design flow to ensure an end-to-end open-source delivery of the IP.**

***Index Terms*—LWC, ASCON, FPGA, Reconfigurable computing, ASIC**

I. Introduction

While the conventional cryptographic primitives are designed to serve high-performance computing applications, there are no cryptographic primitives intended for physically constrained Internet of Things (IoT) applications. For this reason, the National Institute of Standard and Technology (NIST) launched a worldwide competition for defining the right security primitives (hash function and authenticated encryption with associated data (AEAD)) that were able to meet the aggressive requirements in terms of power consumption, battery lifetime and limited resources on-chip that IoT devices depending on for providing valuable services. Those security primitives are called lightweight cryptography (LWC) [1]. With extreme low resource demand, applicability to various devices, and protection from typical attacks, lightweight cryptography methods fulfill the need for security in physically constrained applications like embedded systems, Radio-frequency identification (RFID) and other IoT scenarios.

This paper architects a custom reconfigurable processor named RECO-HCON for one of the LWC finalists, ASCON that applies sponge construction. The sponge interface illustrated in Fig. 1 iteratively updates a state memory based on variable-length input and takes a portion of the state memory as output [2], [3]. It not only relieves the pressure on devices with fixed small memory and straightforward XOR operations but also offers extra flexibility to support arbitrary input/output size and multiple functions including hashing, random number generation, and encryption [4]. The proposed processor is designed to support six different modes while still being compact, and thus can be embedded as an IP for SoCs to secure physical and power-constrained devices used in various IoT applications.

The rest of the paper is organized as follows. Section II compares this paper with other state-of-the-art lightweight cryptography hardware and highlights the contributions of our work. A brief introduction to ASCON Cipher Suite is given in Section III. Section IV presents the motivation and implementation details of the proposed RECO-HCON architecture. Section V shows the evaluation methodology and evaluation results on FPGA and ASIC. The performance is compared among instances, with state-of-the-art single-instance designs. Finally, the conclusion and prospective work are discussed in Section VI.

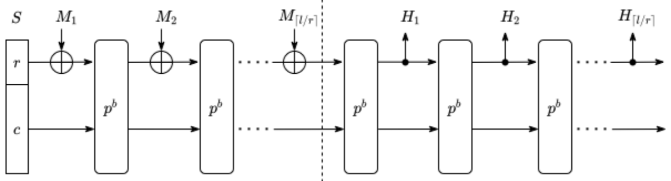

Fig. 1. The Sponge Construction $H_{b,r}(M,l)$

II. Related Work

A few research have been conducted on hardware designs of LWC candidates. A hardware Application Programming Interface (API) is made especially for LWC authenticated ciphers by [5] and assisted the first implementation of SpoC-128 in [6]. [7] proposed a reconfigurable PE micro-architecture, capable of efficiently accelerating multiple LWC ciphers in the S-Box or ARX categories. While some candidates provide authenticated encryption only like GIFT-COFB and Grain-128 implemented in [8], [9], sponge-based algorithms like ASCON and GAGE [10] include both authenticated encryption and hashing functions. All these papers agree on the huge potential of lightweight cryptography methods selected by NIST and the need of bridging the gap between theories and actual hardware. Our work contributes to this area by specifying an innovative reconfigurable system for ASCON.

While [11], [12] implement a single instance of ASCON_128 and ASCON_128a with different optimizations of throughput and power, this paper first integrates ASCON_128, ASCON_128a, and two hash functions in one core. With a small resource overhead, our reconfigurable architecture reaches a similar performance and offers more functionality.

Most ASCON hardware implementations [5], [13] focus on the internal logic of the ASCON core per instance, while the proposed architecture introduces one ASCON core supporting all instances. In addition, the proposed architecture adapts to rate-changing communication using FIFOs to make it quickly adopted by any IoT system.

III. ASCON Cipher Suite

The ASCON cipher suite is based on sponge duplex construction that provides users with both authenticated encryption and hashing functionality. While ASCON can be extended to a spectrum of designs with different parameters, two authenticated encryption instances and two hashing functions are recommended in its submission to NIST [2], [3]. These four instances share the same state memory permutation. In this section, the authenticated encryption instance ASCON_128a is introduced while the other encryption algorithm and hashing functions are illustrated by comparing them with ASCON_128a. Detailed specifications of ASCON instances can be found in [2], [3].

A. ASCON_128a

ASCON_128a encryption centers around permutations of a 320-bit sponge construction state S. A single-round permutation consists of three in-order stages - constant additions, substitution, and linear diffusion. The sponge state is initialized based on instance parameters, 128-bit secret key K and nounce N. The input associated data and plaintext are first padded and divided into blocks with fixed length $r = 128$. Each block is XORed with the most significant bits of the sponge state, the whole state is then permutated for $b = 8$ rounds. The XOR result of each plaintext block and the sponge state is outputted as partial ciphertext. The operation is repeated until associated data and plaintext run out so the final concatenated ciphertext has the same length of plaintext. A 12-round permutation and XOR with the secret key are performed specially for initialization and finalization to increase the complexity and yield the tag.

The decryption process is similar to encryption with only two differences. Ciphertext replaces plaintext without being padded and an additional comparison is needed to verify the correctness of the expected tag.

B. ASCON_128

Compared to ASCON_128a, ASCON_128 has a shorter block size $r = 64$. The iterative permutation has $b = 6$ rounds instead of 8 rounds. The other parts of ASCON_128 are identical to ASCON_128a.

C. Hashing

The two recommended hashing functions ASCON_Hash and ASCON_Hasha can be considered simplified versions of authenticated encryption. The input messages are processed the same way as associated data. There is no plaintext in hashing so each output hash block is a part of the sponge state. The hashing function continues until the length of generated hash text reaches 256 bits. Without key and nonce in hashing, the XOR operation in initialization and finalization is skipped.

In general, instances of authenticated encryption and hashing functions share a similar sponge structure, which lays the theoretical basis for our architecture's integrating all instances. It is also noticed that the main differences between instances are the permutation round, the block size, and the input XOR operand between permutations.

IV. RECO-HCON Architecture

Since IoT applications cover a wide range of devices and tasks, certain ASCON instances might be only suitable for certain devices. Giving these applications freedom to choose suitable instances is essential to ensuring good compatibility of the hardware. Our architecture is motivated to support all recommended functionality of ASCON with a small overhead over single instance implementation. The reconfigurable architecture uses the same sponge interface and mitigates the differences between instances with padding FIFOs, a parameterized permutation, and an XOR operand selection & shift stage. To help IoT applications decide which instance to use, the differences between four instances and their corresponding configurations are given in Table I. The execution time of each instance at its maximum frequency assuming the same data and plaintext size is plotted in Fig. 2 as well. Although encryption and decryption share the same configuration, their execution time is slightly different due to padding and tag generation. ASCON_128 encryption and decryption time overlap when the message size is larger than 2^3 bytes.

A. ASCON Cipher Processor

The proposed architecture in Fig. 3 includes the ASCON core, FIFOs, and the processor's I/O interface to connect to external devices. The ASCON I/O interface only specifies all the input/output ports and there is no logical unit inside. The input data and text are first buffered in their corresponding 128-bit FIFO. The key, nounce, and target instance mode are recorded when the $Start$ signal is triggered. The ASCON core first selects and reads the Data FIFO until it's empty then switches to the Text FIFO. Reading a plaintext block from the Text FIFO and writing a ciphertext block to the Cipherout FIFO is in the same cycle and controlled by one signal $Push$ inside the ASCON core. The processing results are stored in the Cipherout FIFO and wait for external readouts. The ASCON core remains busy until the ciphertext size reaches the input message size in AEAD mode or 256 bits in hashing

TABLE I
INSTANCE-WISE CONFIGURATIONS OF RECONFIGURABLE CUSTOM ARCHITECTURE

Instance	Size (bits)				Permutation Round		Architecture Configuration		
	Key	Nounce	Tag	Block	Initial/Final	Processing	FIFO (block/entry)	Head Mux	Tail Mux
ASCON_128a	128	128	128	128	12	8	1	Data / Plaintext	Padded Key / Constant
ASCON_128	128	128	128	64	12	6	2	Shifted Data / Plaintext	Padded Key / Constant
ASCON_Hasha	-	-	-	64	12	8	2	Shifted Message / 0	0
ASCON_Hash	-	-	-	64	12	12	2	Shifted Message / 0	0

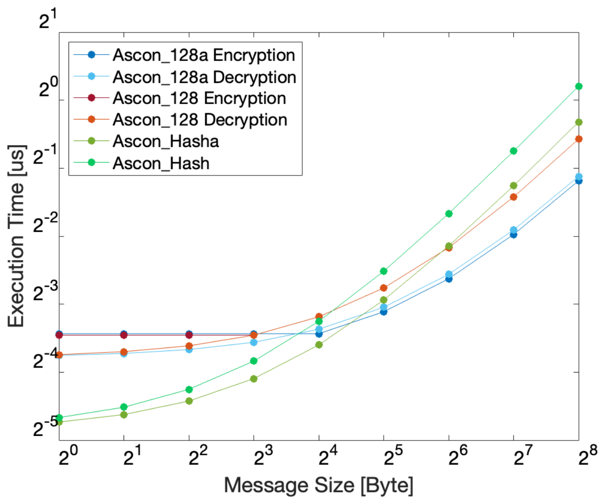

Fig. 2. Execution Time vs. Message Size of Different Instances

mode. Then it raises the $Done$ signal and passes generated tag to Tag FIFO or outputs the tag verification result. Switching instances during processing is not allowed to avoid aliasing. The $Warning$ flag will be raised until the current processing finishes.

The four FIFOs offer important flexibility for communication between the ASCON processor and IoT systems and mitigate block size difference. While the ASCON core runs at a certain frequency, the connected IoT system is not deterministic and may run at a different frequency. FIFOs buffer message transmission and adapt the system to multiple frequency domains. They also allow message transmission over non-constant intervals and control ASCON core implicitly with the number of valid entries. On the other hand, FIFOs are responsible for dividing and padding the input messages into 128-bit entries whose size is identical to ASCON_128a block size. For ASCON_128 and hashing functions, one 128-bit entry is treated as two 64-bit blocks and a different counting logic is applied to fully utilize the space.

B. ASCON Core

The ASCON core in Fig. 4 has two stages the selective XOR and parameterized permutation and is configured as ASCON_128a by default. It applies an iterated strategy that once the core is started, the permutation block is always busy. The permutation block requests a new $Message$ from FIFOs every b rounds. Between two iterations of permutation, the current sponge state is XORed with one of the message, secret key, or constant and generates the input of the next iteration. The FSM controller keeps track of the progress based on the FIFO status to select the correct round number and the second XOR operand corresponding to common processing, initialization, and finalization.

The biggest challenge of the ASCON core is supporting multiple instances that have arbitrary round number, variable XOR operand, and block size. Parametrizing the round number is straightforward by having a counter in the permutation block. The current sponge state is split into two parts: the most significant 128 bits as head and least significant 192 bits as tail and got updated in parallel to combine similar patterns of XOR operand and block size and reduce the complexity of instance switching. The head and tail muxes select the correct XOR operand based on the current finite state and instance mode. For example, the tail mux outputs zero for hashing functions and common processing of encryption while the padded secret key for initialization and finalization of encryption. Since the core assumes $r = 128$ to provide enough wires for ASCON_128a, shorter blocks of ASCON_128 and hashing functions are first shifted by 64 bits in the head XOR to align the most significant bit of the sponge state.

Compared to a single ASCON_128a core, the overhead of supporting multiple instances is not significant because most components are reused and expanded. The parametrized round number is required for ASCON_128a to support the special permutation in initialization and finalization while a selection logic before XOR is necessary as the variable operand changes along with the encryption progress. The two major modifications are a wider mux to select more operands based on the instance mode and the shift logic ahead of it to align blocks in different lengths. The added logic is optimized especially to reach the same timing performance as a single instance. The wide mux is separated into two smaller head and tail muxes to reduce the selection latency and there are only two possible shifting distances corresponding to 64 and 128 bits block sizes. To prevent the shift & selection logic between permutations from being the critical path, an extra cycle is assigned to it and the maximum frequency is the same as a single instance.

C. Permutation

The multi-round permutation uses a one-round block with a feedback loop connecting its output to input and a round counter to stop the permutation when the target round number is reached.

The one-round permutation stores the sponge state in five 64-bit registers and passes it through three layers in order. The constant addition layer first selects a constant based on the current round number and XOR a subset of the sponge state with it. The state is then mapped in a 64 5-bit S-box

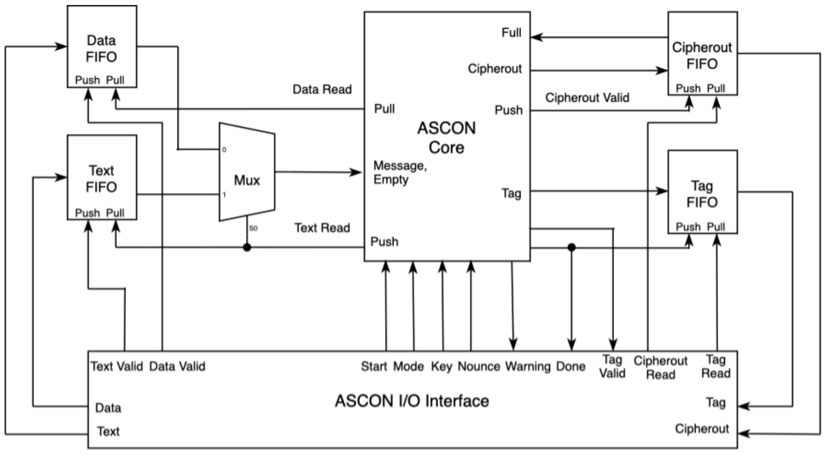

Fig. 3. The Architecture of ASCON Cipher Processor

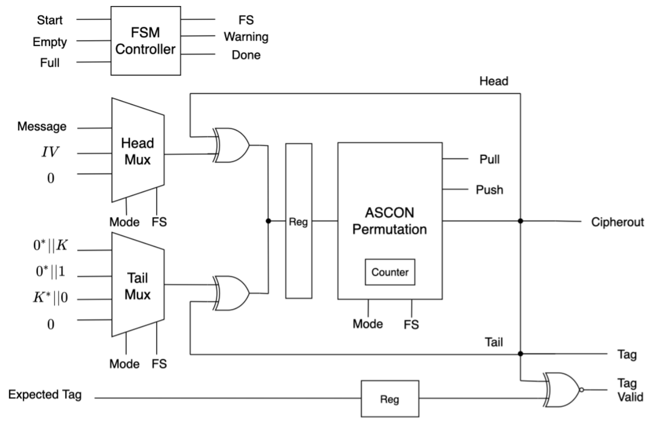

Fig. 4. The Architecture of ASCON Core

followed by the linear diffusion XORing each register with shifted versions of itself. With XOR, look-up table (LUT), and shift operations only, the one-round permutation allows the architecture to achieve a short critical path and moderate resource consumption.

V. Evaluation Results

The following metrics are highly concerned by IoT applications and are thus considered in our evaluation. They are Area/Resources, Frequency, Throughput, Power, and Energy Efficiency. The architecture is implemented with Chisel, a hardware description language supporting object-oriented and functional programming [14]. Compared to traditional hardware design approaches, Chisel offers extra productivity and scalability [15]. The Chisel design is synthesized into RTL source files for further evaluations. Prototype tests and optimizations are run on Basys 3 Artix-7 FPGA and the final design is presented on ASIC in multiple technology nodes.

A. FPGA Results

The proposed ASCON cipher system achieves a maximum frequency 213 MHz and consumes 2733 LUTs and 1462 Flip-flops. A large portion of the critical path and resources comes from FIFOs. While the standalone ASCON core can run at 244 MHz with 1548 LUTs and 1045 Flip-flops.

B. ASIC Results

IoT vendors have been less aggressive in terms of migrating to newer nodes due to high NRE costs. For example, 28nm CMOS remains as a dominant technology for high-end IoT devices. For low-end devices, such as sensing devices, even older (yet cheaper) technologies have been employed. To demonstrate that the proposed ASCON architecture is able to meet requirements for a wide range of IoT scenarios, we implemented the design in both 28/32nm from Synopsys [16] and 130nm technologies from Skywater [17]. The 28/32nm design has been implemented and evaluated with commercial tool suites that include Design Compiler©, IC Compiler©, and Primetime©. As our ultimate goal is to fully open source the design and benefit the community, we also evaluated the design with OpenLane [18], an automated RTL to GDSII flow based on all open-source EDA tools with 130nm open-source PDK. This validation enables an end-to-end delivery of the proposed security IP from the design description (in Chisel) all the way to tapeout-ready layout (in GDS) all through open-source channels.

In both flows, designs have been synthesized and implementations with respective tools, from which the gate-level netlists are fed to the physical implementation tools for floorplanning, place and route (P&R). Based on the P&R results, we extract the area information (e.g. gate count). Static timing analysis (STA) with sign-off requirements is used to estimate the maximum clock frequency. To cover the worst-case scenarios, the design has been synthesized across multiple corners (SS/FF, high/low temperature, and nominal operating voltage) of the corresponding technology nodes. We ensure that the positive slack requirement for both setup and hold time are met. The average power is evaluated with the power analysis tools that are available to each tool suite.

Fig. 5 and 6 show the final layout of the whole ASCON system implemented with 28/32nm and 130nm technology nodes respectively. The major components of the system are highlighted. The compactness of the design is reflected by the total area, which is only $0.068mm^2$ in 28/32nm. For the 130nm version implemented with the OpenLane flow, the proposed design was able to run at 131MHz while consuming only 0.14mW of the total power, which suits the requirements of the low-end IoT devices [10]. Due to page limit, we will mainly discuss the 28/32nm implementation results in the

TABLE II
COMPARISON BETWEEN LWC DESIGNS. THE REPORTED ENERGY IS BASED ON **8192** BITS DIGESTION.

LWC Design	Functionality		FPGA			ASIC						
	AEAD	Hashing	Chip Family	Max.Freq (MHz)	Resource	Tech.	Gate Count (kGE)	Max.Freq (MHz)	Power (μW)	Throughput (Gb/s)	Energy Reported (pJ/bit)	**Energy Scaled to 28/32nm (pJ/bit)**
RECO-HCON (Proposed)	128a		Artix-7	244	LUT:1548 FF:1045	28/32 nm	25.1	667	1990	9.077	0.219	0.219
	128									5.926	0.335	0.335
		Hasha								4.534	0.439	0.439
		Hash								3.160	0.637	0.637
ASCON [11]	128	-	Spartan 6	-	LUT:1297 FF:474	-	-	-	-	2.359	73	-
ASCON [12]	128	-	-	-	-	90 nm	7.08	517	43 @ 1MHz	5.524	-	-
PRO_GAGE [10]	InGAGE	GAGE	Artix-7	250	LUT:436 FF:514	28/32 nm	6.55	909	655	0.181	3.6	3.6
GIFT [8]	64-128	-	-	-	-	90 nm	1.35	543	74@ 10MHz	1.249	216.9	44.89
Grain [9]	128	-	-	-	-	65 nm	2.79	2300	240	1.15	0.209	0.067

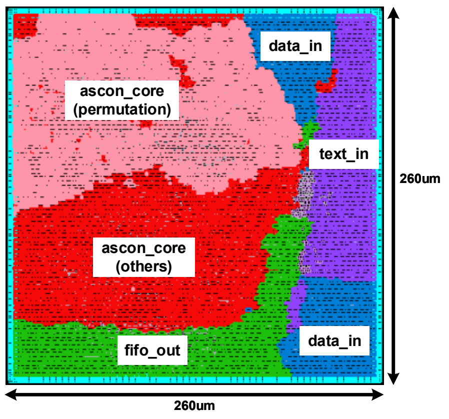

Fig. 5. The final layout of the proposed ASCON Cipher System (implemented with commercial EDA tools in 28/32nm technology node)

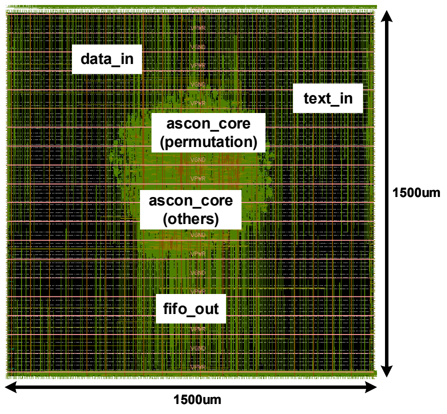

Fig. 6. The final layout of the proposed ASCON Cipher System (implemented with OpenLane flow in Skywater 130nm technology node)

following sections.

C. Comparison Among Different Instances

As different ASCON instances have different block sizes and round number, the latency counted in cycles of each instance mode varies and can be calculated with equation (1), (2), where M is the input message size of hashing and A, P are the input associated data size and plaintext size of AEAD respectively. $Round$ and $Initial\_Final$ are the processing and initial/final permutation round number listed in Table I with one extra cycle added to pipeline the shift & selection logic. The latency of an ASCON AEAD instance's encryption and decryption are the same.

The throughput can be calculated with the latency and $667MHz$ maximum frequency and the energy per bit is acquired by dividing the operation power with the throughput of each instance (3), (4). The throughput and energy consumed to process 8192 bits assuming $A = 0$ are listed in the first row of Table II. ASCON_128a has the highest energy efficiency mainly because its block size is twice the block size of other instances. Hashing instances have lower throughput and energy efficiency because their permutation round number is larger than AEAD's round number.

$$L_{Hashing} = (\frac{M}{BlockSize} + 3) \cdot Round + 2 \cdot Initial\_Final \tag{1}$$

$$L_{AEAD} = \frac{A + P}{BlockSize} \cdot Round + 2 \cdot Initial\_Final \tag{2}$$

$$Throughput = \frac{MaxFreq \cdot DigestBits}{L} \tag{3}$$

$$Energy/Bit = \frac{Power}{Throughput} \tag{4}$$

D. Comparison with State-of-the-art Designs

The full evaluation results and comparisons against other state-of-the-art LWC designs are listed in Table II. To make a fair comparison, the energy efficiency of state-of-the-art designs in different technologies have all been scaled to 28/32nm based on equation (5), (6) in [19]. The ratio of energy per bit of two technologies is equal to the ratio of their energy factors EF. The energy factor can be calculated with both energy coefficients a_e given by [19] and the supply voltage. Since the supply voltages are not specified in the compared designs, the standard V_{DD} of each technology node reported by ITRS is used instead [19]. Note that we tried our best to normalize the results based on the state-of-the-art scaling techniques for comparison purposes, but the actual results can differ based on actual implementations of the targeted technology nodes.

$$EF = a_{e2}V^2 + a_{e1}V + a_{e0} \quad (5)$$

$$Energy/Bit_y = \frac{EF_y}{EF_x} * Energy/Bit_x \quad (6)$$

An example of scaling the energy efficiency of design [9] from 90nm to 28/32nm is given below.

$$EF_{28/32nm} = 0.8367 \cdot 0.97^2 - 0.4341 \cdot 0.97 + 0.1701 = 0.536$$
$$EF_{90nm} = 4.762 \cdot 1.1^2 - 4.781 \cdot 1.1 + 2.092 = 2.59$$
$$Energy/Bit_{scaled} = \frac{0.536}{2.59} \cdot 0.209 = 0.067$$

While most LWC designs offer one functionality only, the proposed RECO-HCON offers two AEAD and two Hashing instances, twice the number of the second flexible PRO_GAGE [10]. Supporting multiple instances offers flexible options for IoT devices and better compatibility. RECO-HCON also features the highest throughput which is critical for establishing gigabit wireless connection between IoT devices. Although some designs have smaller power consumption, their throughputs are much lower. RECO-HCON achieves the second smallest energy per bit for both reported and scaled comparisons.

It should be noticed that the proposed design also has the best power, throughput, and energy performance among state-of-the-art ASCON hardware implementations [11], [12]. The RECO-HCON processor supports more modes (six) than other implementations that support only a single mode. In terms of area, the proposed processor is about 3.5x larger than [12]. However, a $0.068mm^2$ area is still compact enough to fit in most IoT devices. The area overhead mainly comes from the additional computation and control logic to support multiple instances such as XOR operand selection, block aligning, and programmable permutation round number. Optimizations are made to improve throughput and energy efficiency which are two biggest concerns of IoT systems at the cost of reasonable resources and latency increase. In summary, RECO-HCON offers better throughput and energy efficiency while supporting all four recommended instances, which makes it more available to a wider range of IoT devices.

E. IoT Applications with RECO-HCON Processors

The lifetime of batteries is one of IoT systems' biggest concerns when applying a cryptographic primitive. The number of days a Nickel Cadmium battery (1200 mAh capacity at 1.2 V) lasts can be calculated with the following equation.

$$Days = \frac{1200mAh \cdot 1.2V}{Power \cdot 24h} \quad (7)$$

Given the power of the proposed processor $1990\mu W$ (in 28/32nm), a Nickel Cadmium battery can power the RECO-HCON processor to run standalone without being idle for about 30 days.

VI. Conclusions

To overcome the essential need of adding security primitives on constrained physical devices in IoT applications, we proposed a reconfigurable processor that runs six different modes (encryption, decryption, hash function with variants of data size) based on one of the LWC finalists (ASCON). The proposed processor has been evaluated using 28/32nm (with the commercial flow) and the 130nm technologies (with the Openlane flow). The total area of the 28/32 nm version is only $0.068mm^2$ (25.1k gates), while operating at 667 MHz and consumes $\sim$ 2mW of the power. The 130nm version runs at 131MHz while consuming only 0.14mW of the total power. Compared to the state-of-the-art ASCON implementations which focus on single instance only, the proposed reconfigurable processor achieves the highest throughput and is more energy efficient. The source code of the proposed processor and detailed physical implementation scripts with Openlane will be released on GitHub.

Acknowledgement

This work has been funded by the Air Force Research Laboratory Academy Center for Cyberspace Research (ACCR) Directorate, through the Air Force Office of Scientific Research Summer Faculty Fellowship Program®, Contract Numbers FA8750-15-0-6003 and FA9550-15-0001, a NASA M-STAR Grant# 80NSSC20K1859, a SJTU Explore-X Research Grant D6040004/038, a Huawei Research Grant SA3700121 and a UM-SJTU JI startup fund.

References

[1] W. J. Buchanan, S. Li, and R. Asif, "Lightweight cryptography methods," *Journal of Cyber Security Technology*, vol. 1, no. 3-4, pp. 187–201, 2017.
[2] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," *Journal of Cryptology*, vol. 34, no. 3, p. 33, 2021.
[3] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2," Submission to NIST, 2021. [Online]. Available: https://ascon.iaik.tugraz.at/files/asconv12-nist.pdf
[4] B. Guido, D. Joan, P. Michaël, and V. Gilles, "Cryptographic sponge functions," 2011.
[5] Athena, "Caesar hardware api implementation," accessed: 2021-02-28. [Online]. Available: https://cryptography.gmu.edu/athena/index.php?id=CAESAR_source_codes
[6] A. Sathvik, T. Rahul, A. Baksi, and V. Pudi, "Hardware implementation of spoc-128," Cryptology ePrint Archive, Report 2022/119, 2022, https://ia.cr/2022/119.
[7] J. Anderson, Y. Alkabani, and T. El-Ghazawi, "Recpe: A pe for reconfigurable lightweight cryptography," in *2021 IEEE 34th International System-on-Chip Conference (SOCC)*. IEEE, 2021, pp. 176–181.
[8] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift: A small present," Cryptology ePrint Archive, Report 2017/622, 2017, https://ia.cr/2017/622.
[9] J. Sönnerup, M. Hell, M. Sönnerup, and R. Khattar, "Efficient hardware implementations of grain-128aead," in *Progress in Cryptology – INDOCRYPT 2019*, F. Hao, S. Ruj, and S. Sen Gupta, Eds. Cham: Springer International Publishing, 2019, pp. 495–513.
[10] M. El-Hadedy, M. Margala, S. Mosanu, D. Gligorosk, and W.-M. Hwu, "Pro-gage: A high performance compact gage hash function processor for small space technology," in *2021 IEEE Space Computing Conference (SCC)*, 2021, pp. 9–16.
[11] M. Fivez, "Energy efficient hardware implementations of caesar submissions," Master's thesis, KU Leuven, 2015.
[12] 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.
[13] H. Gross, "Caesar hardware api reference implementations," accessed: 2021-02-28.
[14] M. Schoeberl, "Digital design with chisel," 2019.
[15] X. Guo, M. El-Hadedy, S. Mosanu, X. Wei, K. Skadron, and M. R. Stan, "Agile-aes: Implementation of configurable aes primitive with agile design approach," *Integration*, 2022.
[16] Synopsys, "32/28nm Generic Library for Teaching IC Design."
[17] Google and SkyWater, "The SkyWater Open Source PDK." [Online]. Available: https://github.com/google/skywater-pdk
[18] A. Ghazy and M. Shalan, "Openlane: The open-source digital asic implementation flow," in *Proc. Workshop on Open-Source EDA Technol.(WOSET)*, 2020.
[19] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of cmos device performance from 180nm to 7nm," *Integration*, vol. 58, pp. 74–81, 2017.