International Journal of Science and Engineering Invention (IJSEI)
Volume 11, Issue 10,October 2025
https://doi.org/10.23958/ijsei/vol11-i10/298

Research Article

ASIC-Oriented RTL Implementation of ASCON-128 for IoT Devices

Pujitha Padmanabha[1], Sriraksha Srinivas[1], Sudipta Sunil[1], and Dr. Yasha Jyothi M. Shirur[2]

[1]Under Graduate Student, Department of Electronics and Communication Engineering, BNM Institute of Technology, Bengaluru, India
[2]Professor and Head, Department of Electronics and Communication Engineering, BNM Institute of Technology, Bengaluru, India

Corresponding Author- Sriraksha, ECE Department, BNM Institute of Technology, Banashankari second Stage, Bengaluru, 560070, India.

E-mail: ***srirakshasrinivas17@gmail.com***

Abstract

Objective: The objective was to observe the behavior, correctness, and functional robustness of the ASCON-128 RTL implementation when verified using NIST test vectors. While the current aim is a general implementation, a synthesizable verified and ASIC-oriented hardware model is aimed to be developed which aligns with the lightweight cryptography standards set by NIST.Design:

Methods: ASCON-128 is an authenticated encryption algorithm which was developed in modular structured Verilog HDL that contained a permutation core, control unit, and state management logic which was complemented with functional verification using Cadence Xcelium along with NIST test vectors for verification of encryption, decryption, and generation of authentication tags.

Design: RTL implementation and verification study based on functional observation and simulation analysis.

Results: For the RTL simulation, the full functional correctness was verified and NIST reference ciphertexts and tags demonstrated exact matches. The design security integrity was validated due to the design robustness against altered inputs.

Conclusion: The RTL design in Verilog illustrates a secure and verified starting point for lightweight cryptography that is appropriate for ASIC implementation in IoT devices. This work paves the synthesis, power optimization and other more complex architectures that can be developed in future.

Key words: ASCON-128, Authenticated Encryption, IoT security, RTL design, Verilog

1. Introduction

The growth of the Internet of Things (IoT) includes an array of connected devices, such as wearables, medical sensors, and industrial nodes. These devices need to have robust data protection systems, yet they are highly constrained in power, memory, and space. Although conventional cryptographic standards such as AES and SHA-2 are secure, they are too hardware intensive to be feasible in ultra-low-power situations [3].This is the gap the National Institute of Standards and Technology (NIST) is attempting to fill with the Lightweight Cryptography (LWC) initiative [1]. This resulted in the selection of ASCON as the standard algorithm in 2023 [1,9].

ASCON-128 presents Authenticated Encryption with Associated Data (AEAD) and thus fulfils the confidentiality and integrity requirements in a single, compact structure [1,5]. It also utilizes a 320-bit permutation-based sponge architecture designed specifically to achieve the best performance in constrained environments [2,7].

The need to achieve a hardware-efficient implementation that balances area, performance, and security is the motivation of this work. This paper documents the design and verification of ASCON-128 hardware, developed in Verilog and verified with official NIST test vectors [1,6].

Table. 1. Comparison of ASCON-128 RTL Outputs with NIST Reference Test Vectors

Test No.	Input Description	Expected Output (NIST Reference)	RTL Output	Result	Interpretation
Test 1	Standard input from NIST test vectors	Ciphertext & Tag as per NIST	Ciphertext & Tag matched	PASS	Encryption and authentication working correctly; design compliant with NIST.
Test 2	Another standard input from NIST test vectors	Ciphertext & Tag as per NIST	Ciphertext & Tag matched	PASS	Confirms correct encryption/authentication; functional correctness verified.
Test 3	Intentionally modified input	Ciphertext & Tag should not match	Ciphertext & Tag did not match	FAIL	Expected failure; shows design correctly rejects invalid data, robust behaviour.

2. Methods

This work presents the implementation of the lightweight authenticated encryption algorithm ASCON-128 at the Register Transfer Level (RTL), focusing on its suitability for IoT systems. The design follows an industry-standard digital design flow up to functional verification, with the intent to extend to ASIC implementation in future phases [3,7]. Guided by the NIST-approved ASCON-128 specification, this implementation offers efficient and secure authenticated encryption, particularly in resource-constrained environments [1,9].

2.1 Algorithm Analysis

The ASCON-128 algorithm is centred on a permutation-based sponge construction that provides authenticated encryption with associated data (AEAD) [1,5,9].

Fig. 1. ASCON Operation

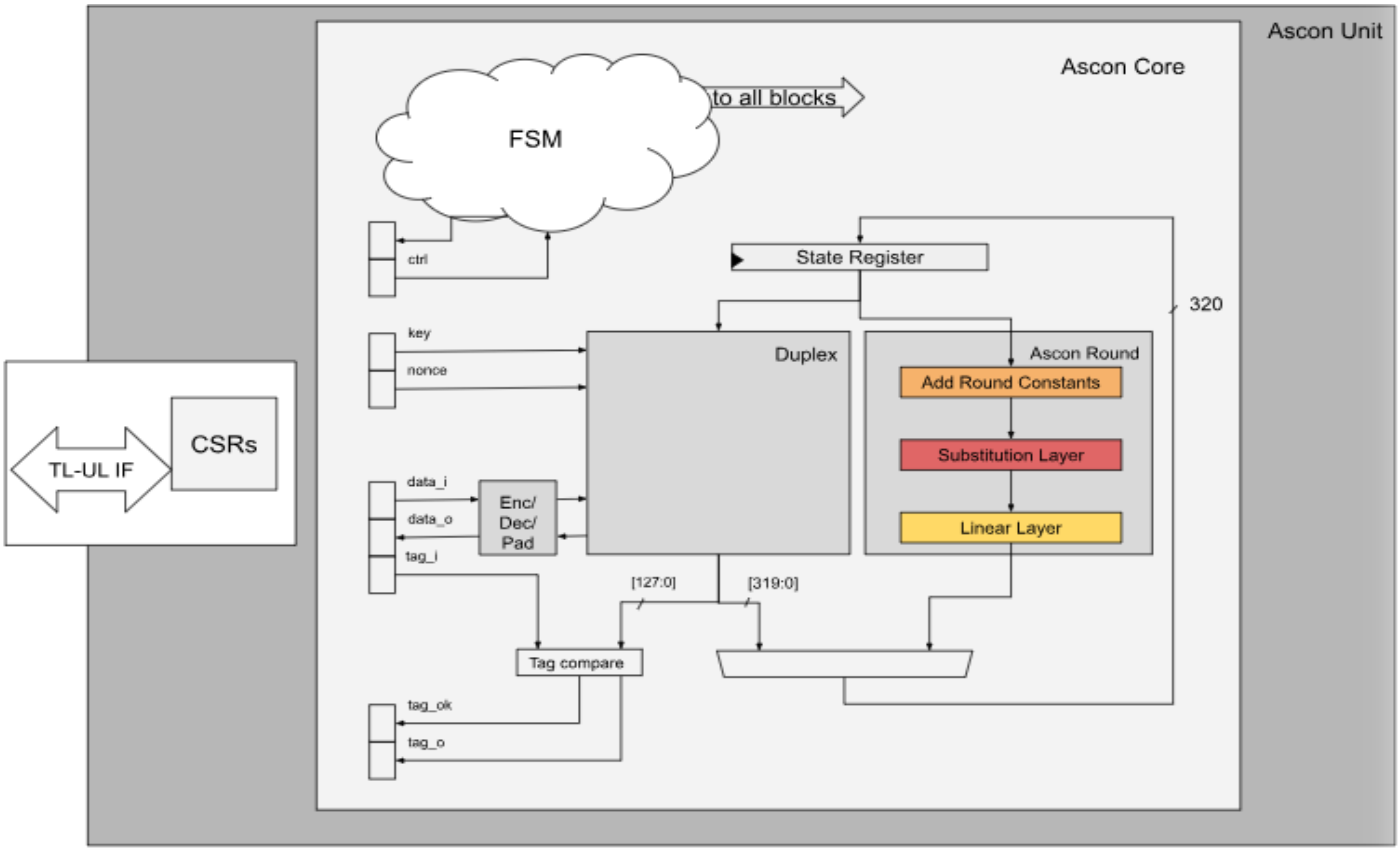

Fig.1 shows the operation of ASCON. It functions with a 320-bit internal state, divided into:

- Rate (r): 64 bits that interact directly with input/output data [1].
- Capacity (c): 256 bits that ensures security through state secrecy [1].

Key algorithmic phases include:

1. Initialisation: Loading the secret key, nonce, and initialisation vector into the state and then followed by a 12-round permutation.

$$\boldsymbol{S \leftarrow IV, k, r, b \parallel K \parallel N}$$

- IV, k, r, b is an initialization vector that depends on the key size (k), rate (r), and number of bits (b). This is typically a fixed value in ASCON-128.
- K is the 128-bit secret key.
- N is the 128-bit nonce.

The 320-bit state S is represented as five 64-bit words. In ASCON-128, the key and nonce are mapped into specific positions, while the remaining words are initialized to zero. Typically, x_0 contains the IV and part of the key, x_1 includes the remaining key and part of the nonce, and x_2–x_4 hold the nonce and zeros. This structure ensures complete integration of the key and nonce into the internal state.

2. Associated Data Processing: XOR-based absorption of associated data (AD) blocks at rate r and then padding is applied if AD's length isn't a multiple of r.

$$\boldsymbol{S \leftarrow (S_r \oplus ADblock) \parallel S_c}$$

Each block is then is interleaved with 6-round permutations for diffusion.

3. Encryption/Decryption: In Fig. 2 processing message blocks through XOR operations and permutation layers to generate cipher text and recover plaintext.

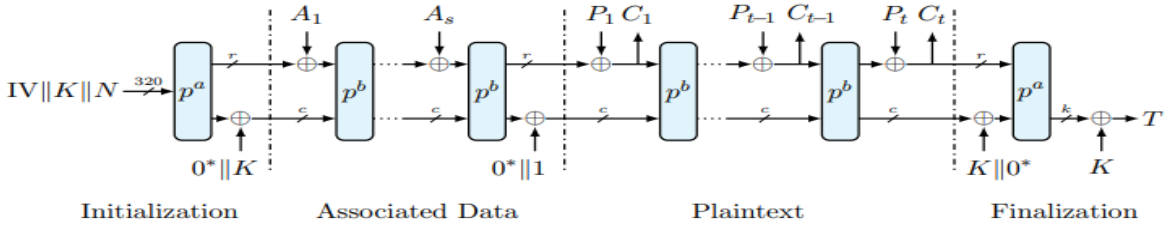

Figure 1: ASCON encryption and tag generation

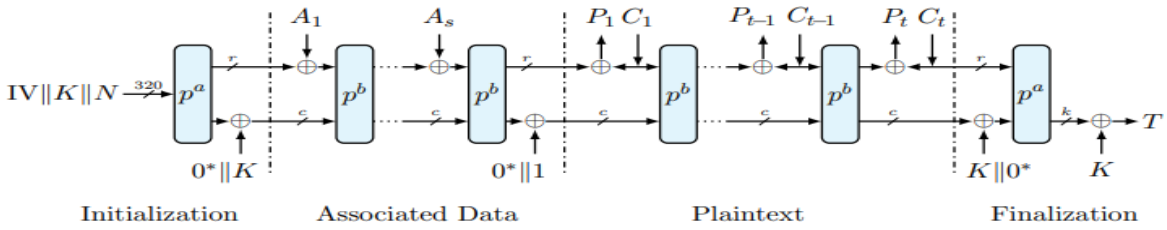

Figure 2: ASCON decryption and tag verification

Fig. 2. Encryption and Decryption in ASCON

For **encryption**, each plaintext block P_i is XO Red with the x_0 state which is the rate part of the state to form the cipher text:

$$\boldsymbol{C_i = P_i \oplus x_0}.$$

The resulting cipher text block replaces the rate portion, and a 6-round permutation $p_b(S)$updates the state. Padding is applied if the final block is incomplete.

During decryption, each cipher text block $C_i$is XORed with the x_0 state to recover the plaintext, after which $x_0$is updated with $C_i$and the same 6-round permutation is applied.

$$\boldsymbol{P_i = C_i \oplus x_0},$$

Both operations follow identical state transitions, ensuring synchronization and consistent tag computation. After processing all blocks, the key is reabsorbed, a 12-round permutation $p_a(S)$is applied, and the authentication tag $\boldsymbol{T = (x_3, x_4)}$**i**s generated for verification.

4. Finalisation: Re-absorption of the key and tag generation to ensure authenticity.
The finalization phase securely links the processed state to the secret key, ensuring confidentiality and authenticity. Once all plaintext and associated data blocks are processed, the 128-bit key K is split into two 64-bit halves.

$$\boldsymbol{K = (K_0, K_1)}$$

The halves are XORed into the capacity(c) portion of the state.

$$\boldsymbol{x_3 = x_3 \oplus K_0, x_4 = x_4 \oplus K_1}.$$

The state then goes through a full 12-round permutation $p_a(S)$for complete diffusion. After that, the key is reabsorbed

$$\boldsymbol{x_3 = x_3 \oplus K_0, x_4 = x_4 \oplus K_1}.$$

And the 128-bit authentication tag is extracted as $\boldsymbol{T = (x_3, x_4)}$.

During decryption, the same process is repeated. the recomputed tag is compared with the received tag. If they match, it confirms authenticity, while any mismatch causes the ciphertext to be rejected.

2.2 RTL Design

The RTL was coded in Verilog HDL with modular and synthesizable design principles. The architecture contains a Permutation Core (ARC, S-box, linear diffusion), State Management Logic, and a Control Unit (FSM) [3,7]. A serial plan reduces silicon area and power, making it suitable for lightweight applications.

2.3 Functional Verification

The RTL design is verified using EDA Playground simulations with standard NIST ASCON-128 test vectors [1,6]. The verification process includes:

- Applying standard ASCON-128 test vectors released by NIST.
- Validating the correctness of encryption and decryption outputs.
- Verifying authentication tag generation and validation.

The simulations confirm logical correctness and cycle-accurate behaviour, establishing the baseline implementation as functionally valid and consistent.

2.4 Design Tools and Environment

EDA Playground was used for functional verification and testbench execution, including robustness checks by varying input values [3,6]. Synthesis, timing/power analysis, and backend ASIC steps are planned for future extensions [7].

3. Results

ASCON-128 RTL was verified against the official test vectors of NIST for lightweight cryptography. In this regard, it can be regarded as being fully compliant with functionality in terms of standards.

3.1 Ciphertext and Tag Verificatio

Fig. 3. Ciphertext and Tag Verifying

```
Test 1:
Obtained CT    = 11e9dc9ae53b37d4c03789320cb95efa
Expected CT    = 11e9dc9ae53b37d4c03789320cb95efa
Obtained TAG   = 5a0be2154d882c153fb0f6ae981e23ad
Expected TAG   = 5a0be2154d882c153fb0f6ae981e23ad
Result: PASS
==========================================
==========================================
Test 2:
Obtained CT    = b79c8a40a301d1a642edf7fb4b74114f
Expected CT    = b79c8a40a301d1a642edf7fb4b74114f
Obtained TAG   = 0b4a80e7e32cebc9f00ac448a21f4a15
Expected TAG   = 0b4a80e7e32cebc9f00ac448a21f4a15
Result: PASS
==========================================
==========================================
Test 3:
Obtained CT    = 87cf1dce878f24e193d1ddc079473dcc
Expected CT    = 0123456789abcdef0123456789abcdef
Obtained TAG   = a7220b10a33c96bd019f21f7edc777a5
Expected TAG   = fedcba9876543210fedcba9876543210
Result: FAIL
==========================================
testbench.sv:125: $finish called at 1515000 (1ps)
Finding VCD file...
./dump.vcd
[2025-09-12 08:32:24 UTC] Opening EPWave...
Done
```

- **Test 1 and Test 2:** For these inputs, the ciphertext and authentication tag from our RTL design matched the expected values from the NIST reference. This shows that the encryption and authentication operations are working correctly. Both tests received a PASS.
- **Test 3:** Here, we intentionally modified the inputs to test the design's strength. As expected, the ciphertext and tag did not match the reference outputs. This mismatch shows that the algorithm correctly identifies invalid transformations and does not incorrectly validate wrong data. Therefore, the result was noted as FAIL, which is the expected behaviour in this case.

5.2 Waveform Analysis

Fig. 4 simulation results show that the ASCON-128 RTL design is functionally correct and robust against altered inputs. The correct ciphertext and tag generation in Tests 1 and 2 confirm compliance with the official NIST specification. The failure in Test 3 ensures that the design does not falsely authenticate invalid data. These results strongly indicate that the implementation accurately represents the ASCON-128 algorithm and is suitable for lightweight cryptographic applications.

4. Discussion

The work establishes a robust, verified RTL model of ASCON-128 that fulfils the lightweight cryptography requirements for IoT devices, as observed in Table. 1 ensuring secure data transmission with minimal hardware overhead.

Compared with traditional algorithms like AES, ASCON-128 offers smaller area, reduced power, and lower computational complexity. The modular RTL architecture enables adaptation to different performance needs — serial versions prioritize minimal resources, while unrolled variants can achieve higher throughput.

In conclusion, in order to meet the lightweight cryptography requirements for Internet of Things devices, the work develops a reliable, validated RTL model of ASCON-128, guaranteeing secure data transmission with little hardware overhead.

Acknowledgements

The authors — Pujitha Padmanabha, Sriraksha Srinivas, Sudipta Sunil, and Dr. Yasha Jyothi M. Shirur — thank BNM Institute of Technology and Visvesvaraya Technological University (VTU) for providing the tools, resources, and support essential for this work.

References

[1] M. S. Turan, K. A. McKay, D. Chang, J. Kang, and J. Kelsey, Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions, NIST Special Publication 800-232, 2025.

[2] M. El-Hadedy, R. Hua, K. Yoshii, and W.-M. Hwu, Optimizing ASCON Permutation in Multi-Clock Domains with Chisel: Resource Efficiency and Critical Path Reduction, 2025. [Conference/Journal info not available]

[3] K.-D. Nguyen, T.-K. Dang, B. Kieu-Do-Nguyen, D.-H. Le, C.-K. Pham, and T.-T. Hoang, ASIC Implementation of ASCON Lightweight Cryptography for IoT Applications, IEEE Trans Circuits Syst I: Regular Papers, vol. 72, no. 5, pp. 1234–1245, May 2025.

[4] H. Groß, E. Wenger, C. Dobraunig, and C. Ehrenhofer, Suit up!—Made-to-Measure Hardware Implementations of ASCON, 2025. [Conference/Journal info not available]

[5] R. Weatherley, Additional Modes for ASCON, Version 1.1, Southern Storm Software, 2023. Available: https://github.com/rweather/ascon-suite

[6] C. Dobraunig, H. Groß, and M. Eichlseder, ASCON Lightweight Authenticated Encryption and Hash Function Reference

Implementation in C, 2023. Available: https://github.com/ascon/ascon-c

[7] A. Malal, High-Performance FPGA Implementations of Lightweight ASCON-128 and ASCON-128a with Enhanced Throughput-to-Area Efficiency, IACR Cryptology ePrint Archive, Preprint, 2025. Available: https://eprint.iacr.org/2025

[8] S. H. Prasad, F. Mendel, M. Schläffer, and R. Nagpal, Efficient Low-Latency Masking of Ascon without Fresh Randomness, IACR Cryptology ePrint Archive, 2023. Available: https://eprint.iacr.org/2023

[9] ASCON v1.2 Submission to NIST (official spec document), ASCON: Submission to NIST Lightweight Cryptography Standardization, 2023. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/ascon-spec-final.pdf

[10] M. Mirigaldi, V. Piscopo, M. Martina, and G. Masera, The Quest for Efficient ASCON Implementations: A Comprehensive Review of Implementation Strategies and Challenges, Chips, 2025. MDPI.