Design an accelerator for lightweight cryptography algorithm Ascon

Application for Synopsys Scholarship

Student: Le Duc Duy

1. Introduction

- In the rapidly evolving landscape of smart cities, ensuring data security in Internet of Things (IoT) systems is a critical challenge. IoT devices often have limited memory, computing power, and energy, making it essential to integrate lightweight cryptographic algorithms that maintain high security levels.
- This project focuses on designing and implementing a hardware accelerator for the Ascon lightweight cryptographic algorithm, which has been standardized by NIST for resource-constrained security applications. The accelerator is optimized to enhance processing speed, strengthen data protection in IoT systems.
- By leveraging an optimized hardware architecture, this solution enables IoT devices to perform encryption and authentication efficiently. This not only improves the performance of IoT systems but also enhances resilience against cyber threats, contributing to a safer and more reliable smart city infrastructure.

2. Architecture

- Previous implementations of Ascon acceleration used a single-round Ascon-p permutation core to reduce cycle count. However, encryption still relied on the CPU, limiting resource optimization.
- The proposed hardware architecture fully implements the Ascon algorithm in hardware, with the CPU only handling control, data input, and output. This approach significantly reduces cycle count and enhances system performance.
- The design of Ascon accelerator is based on the document "Ascon-Based Lightweight Cryptography Standards for Constrained Devices", NIST 800-232, published in November 2024. Supports three Ascon cryptographic modes: Ascon-AEAD128, Ascon-Hash256, Ascon-XOF128

2. Architecture
2.1. Problem

- **Permutation:** The Ascon permutation in NIST SP 800-232 follows Ascon v1.2, using 8-round and 12-round permutations. Implementing this in hardware increases area due to additional logic gates connecting single-round Ascon-p blocks.
- **Loop Execution:** Since hardware cannot continuously expand loops, clock cycles are used to execute each iteration, with each loop corresponding to the required cycle count.
- **Endianess:** Ascon is designed for resource-constrained devices and supports little-endian systems. Therefore, input data to the hardware accelerator must also follow the little-endian format.

2. Architecture
2.2. Ascon Algorithm (Encrypt & Decrypt)

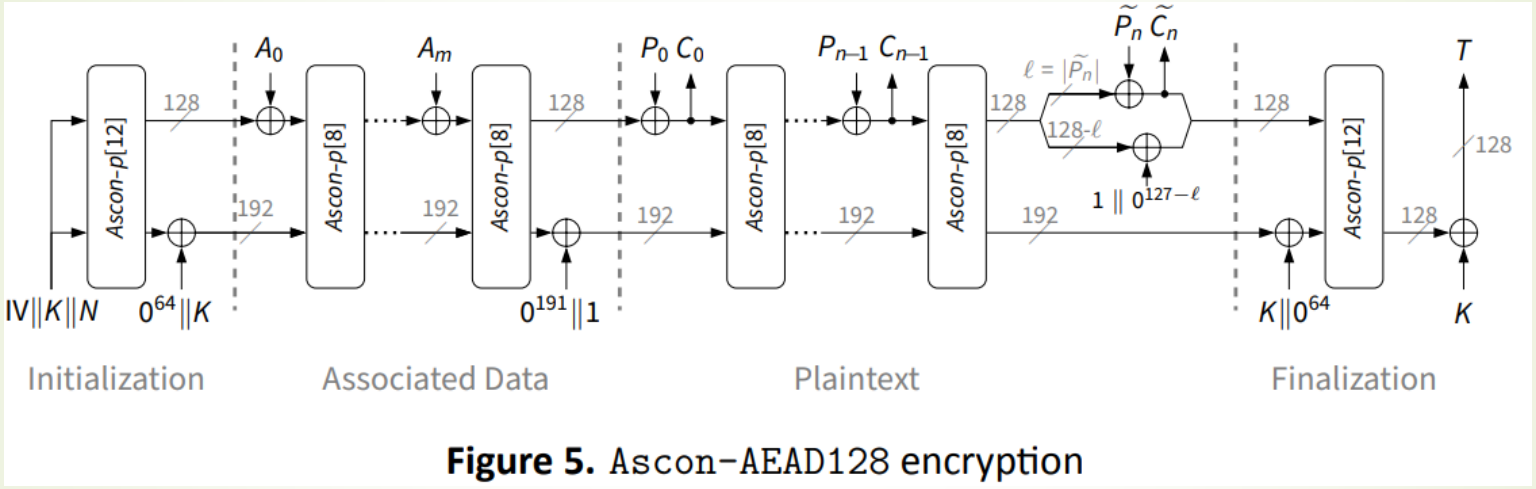


Figure 5. Ascon-AEAD128 encryption

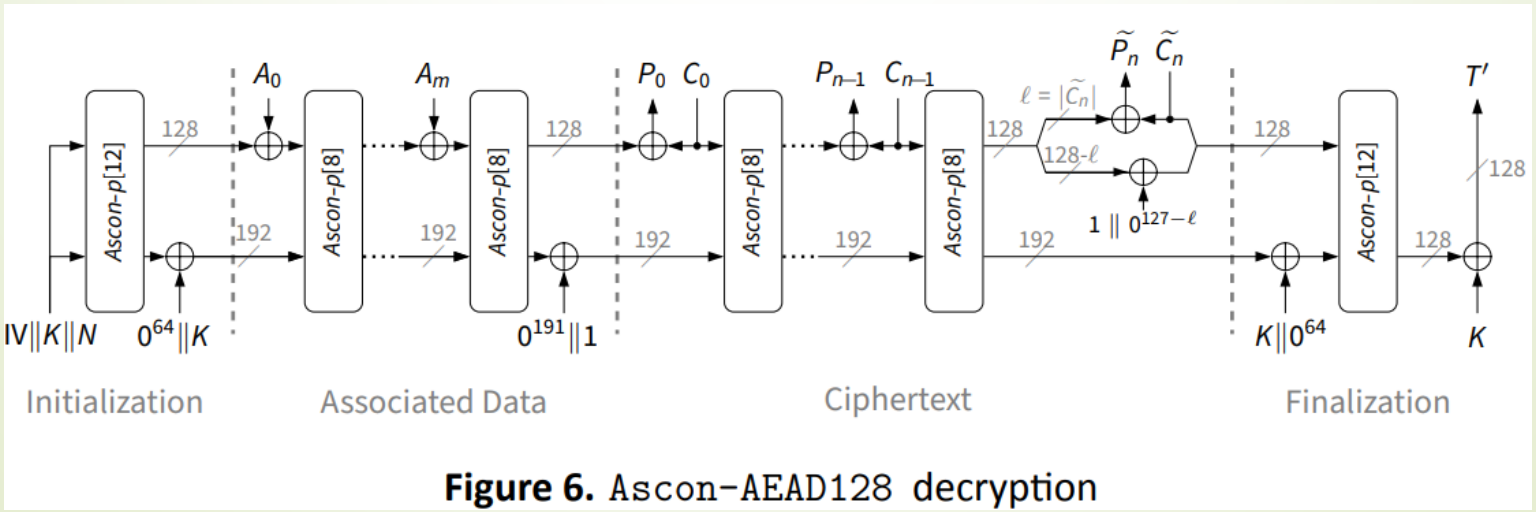


Figure 6. Ascon-AEAD128 decryption

2. Architecture
2.2. Ascon Algorithm (Hash)

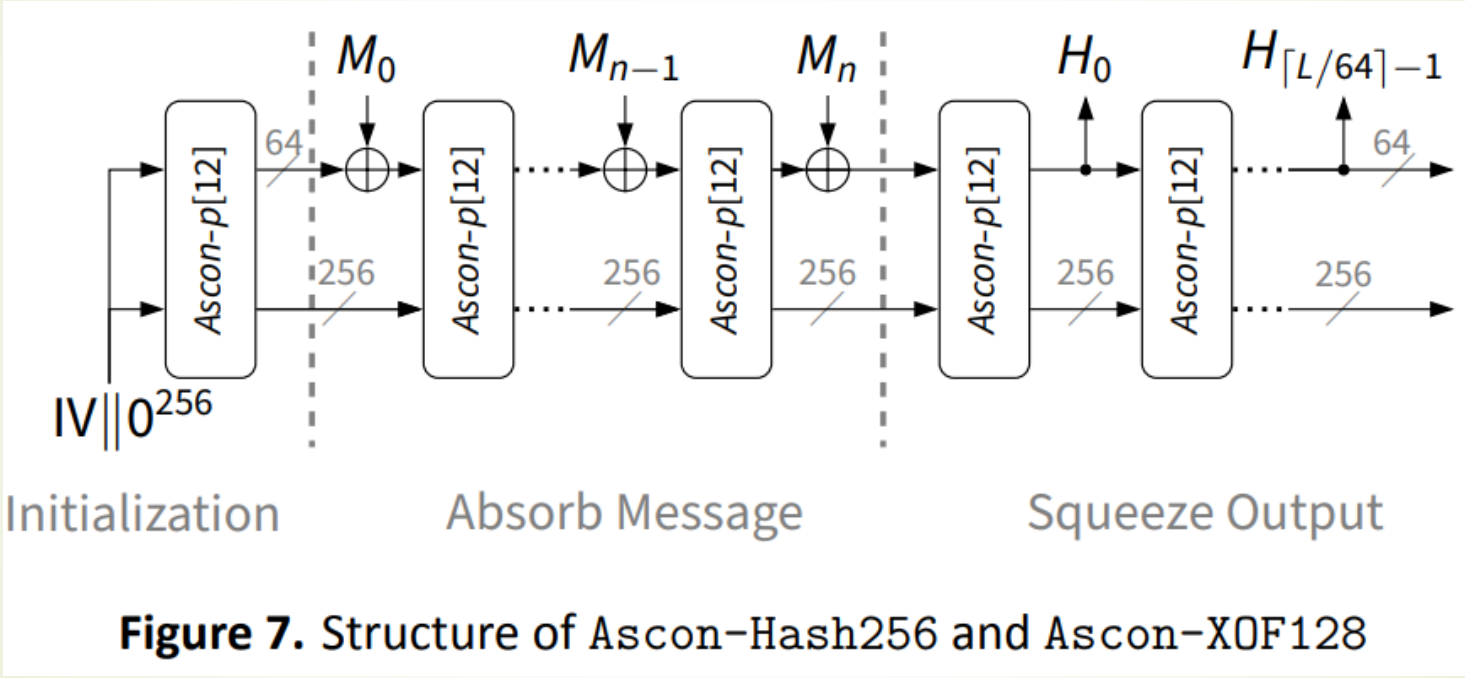


Figure 7. Structure of Ascon-Hash256 and Ascon-XOF128

2. Architecture
2.3.1. Ascon Core

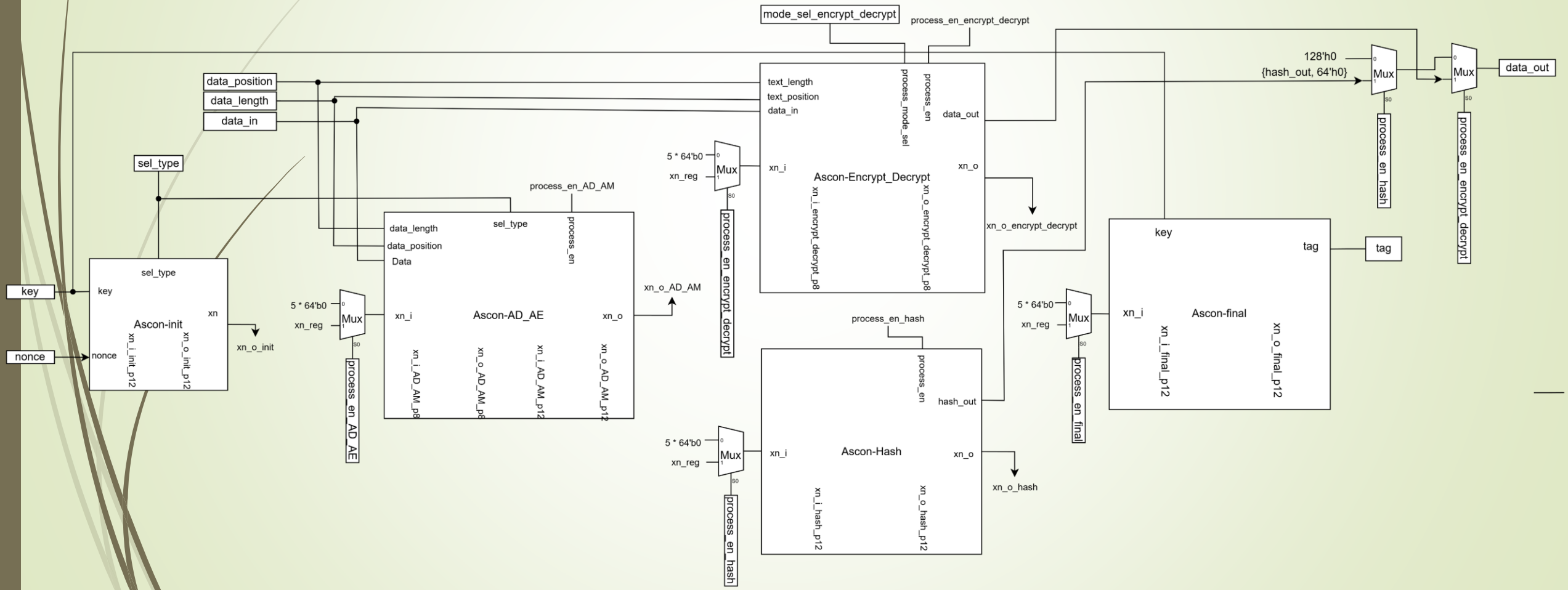
- The Ascon Core module executes the Ascon algorithm based on NIST 800-232, supporting three main cryptographic modes: Ascon-AEAD128, Ascon-Hash256, and Ascon-XOF128. Based on the outlined algorithm model, the essential steps for encryption can be summarized as follows:
- Initialization – Set up initial values.
- Absorption – Process associated data.
- Encryption – Encrypt the main data.
- Authentication (for AEAD only) – Generate a tag to verify data integrity during transmission or storage (Not applicable for hash functions)

2. Architecture
2.3.1. Ascon Core

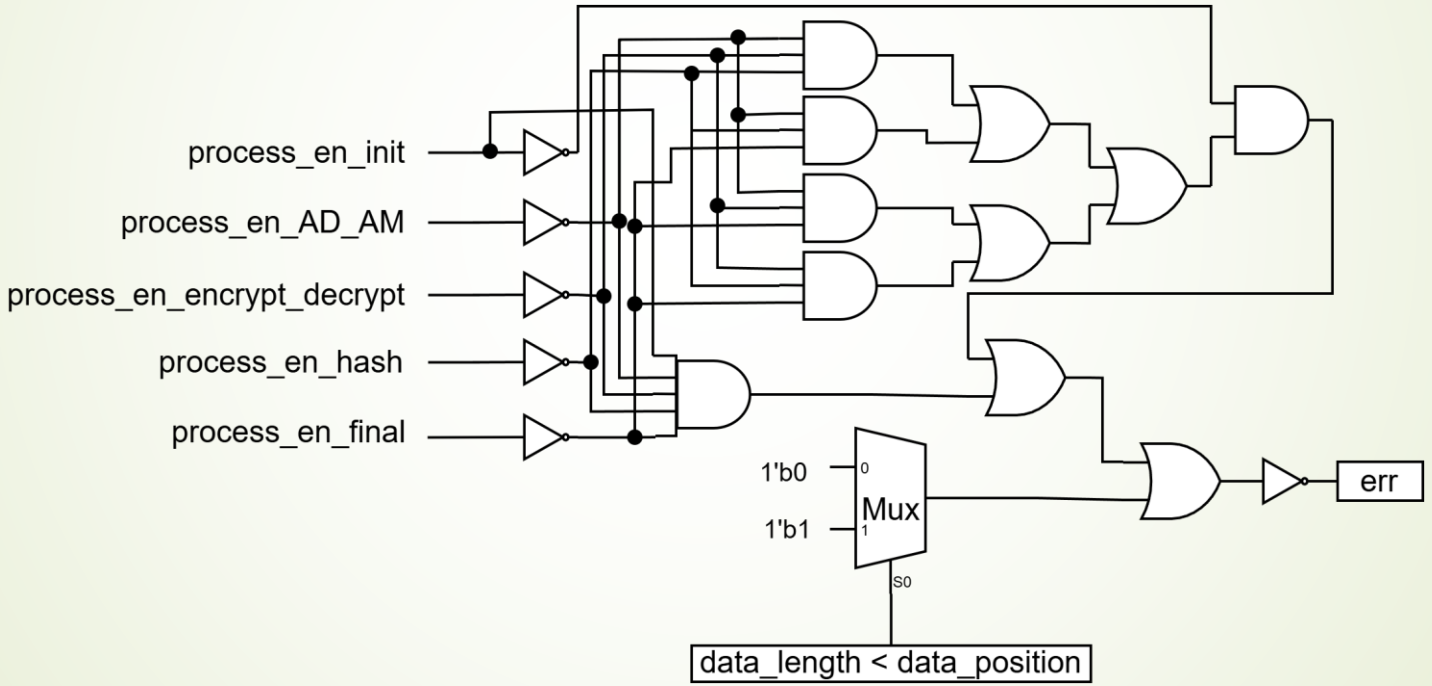
Each encryption step is handled by a dedicated module:

- Ascon-init: Initializes values based on the selected algorithm.
- Ascon-AD_AM: Processes associated data (AEAD) and message absorption (Hash/XOF).
- Ascon-Encrypt_Decrypt & Ascon-Hash: Perform encryption/decryption (AEAD) and hashing operations.
- Ascon-final: Generates authentication tags for AEAD.

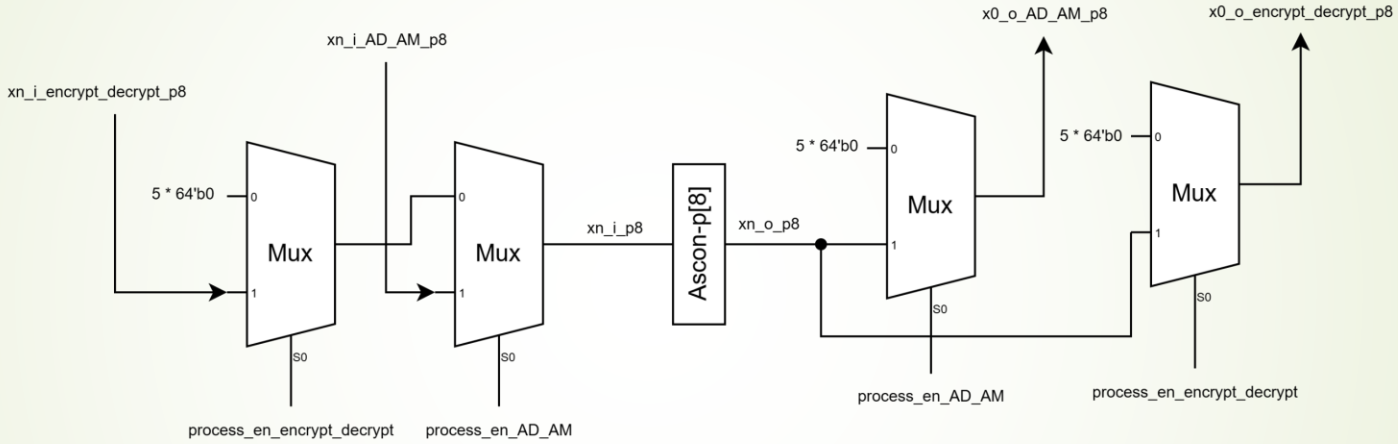
2. Architecture
2.3.1. Ascon Core (Main module)

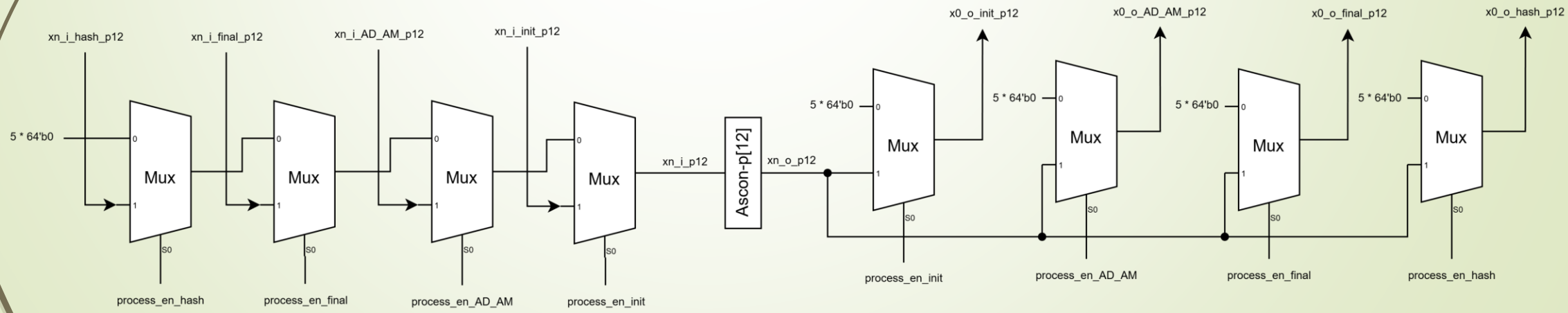


2. Architecture
2.3.1. Ascon Core (Error Handle)



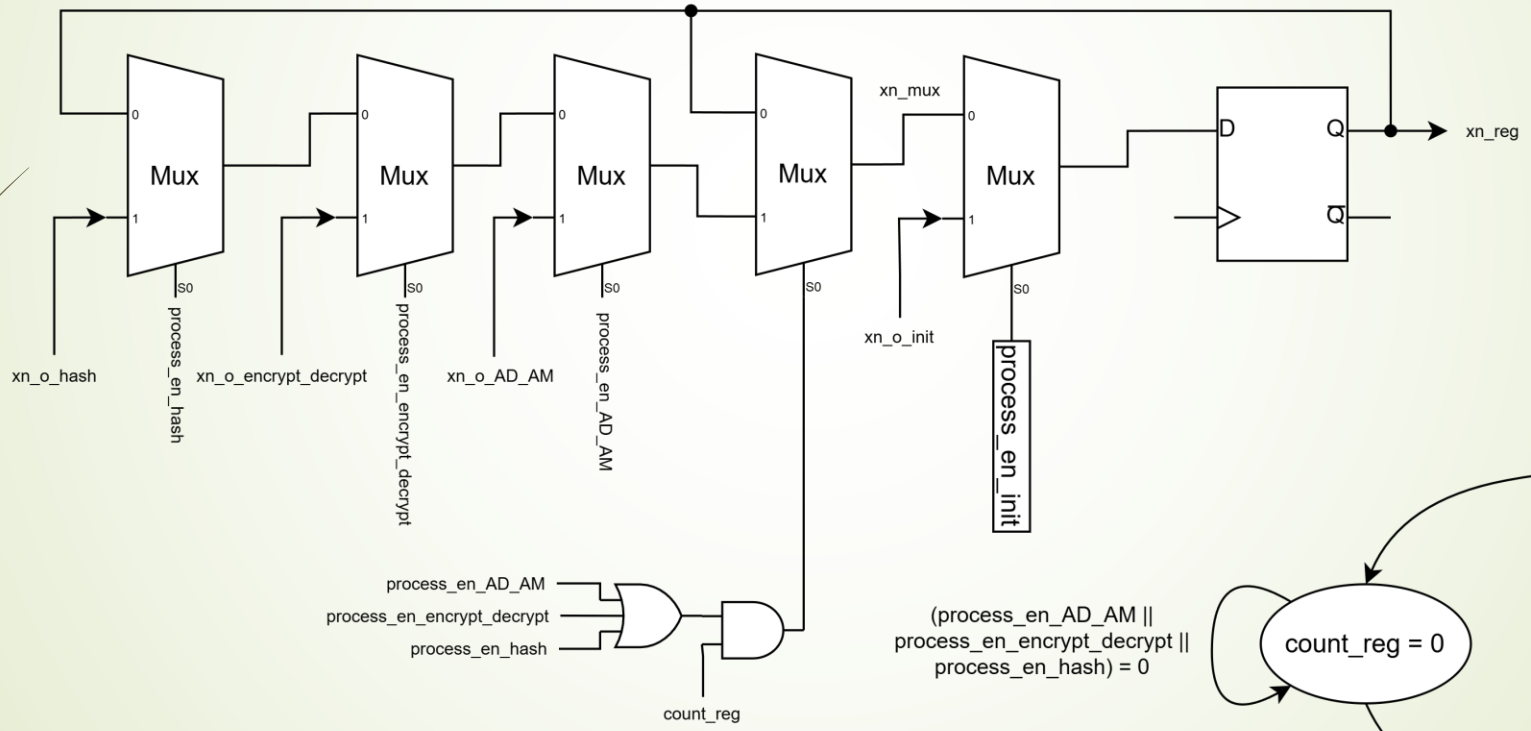
2. Architecture
2.3.1. Ascon Core (Data Path Division)

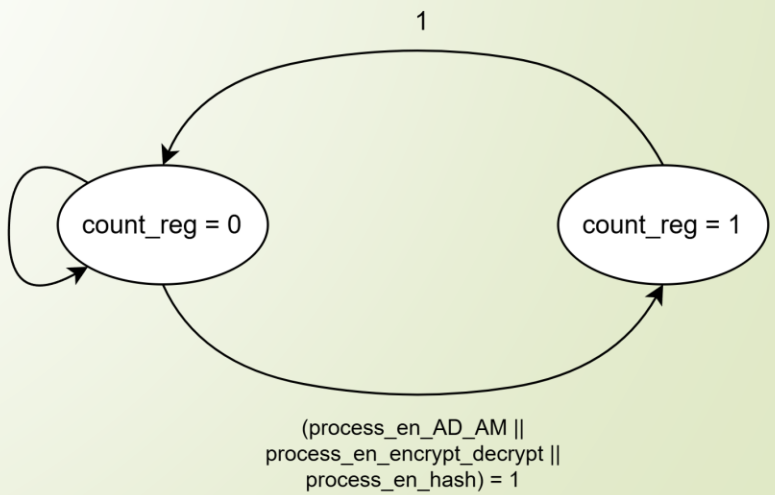




2. Architecture

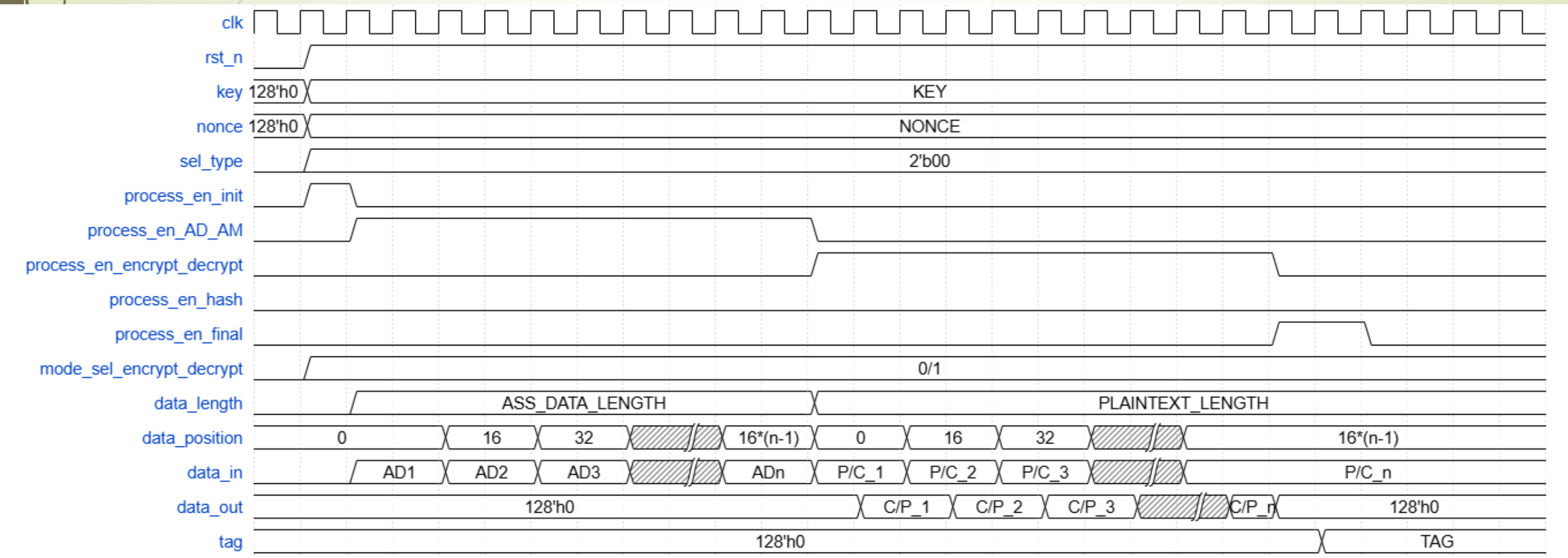
2.3.1. Ascon Core (Process State and save data to register)





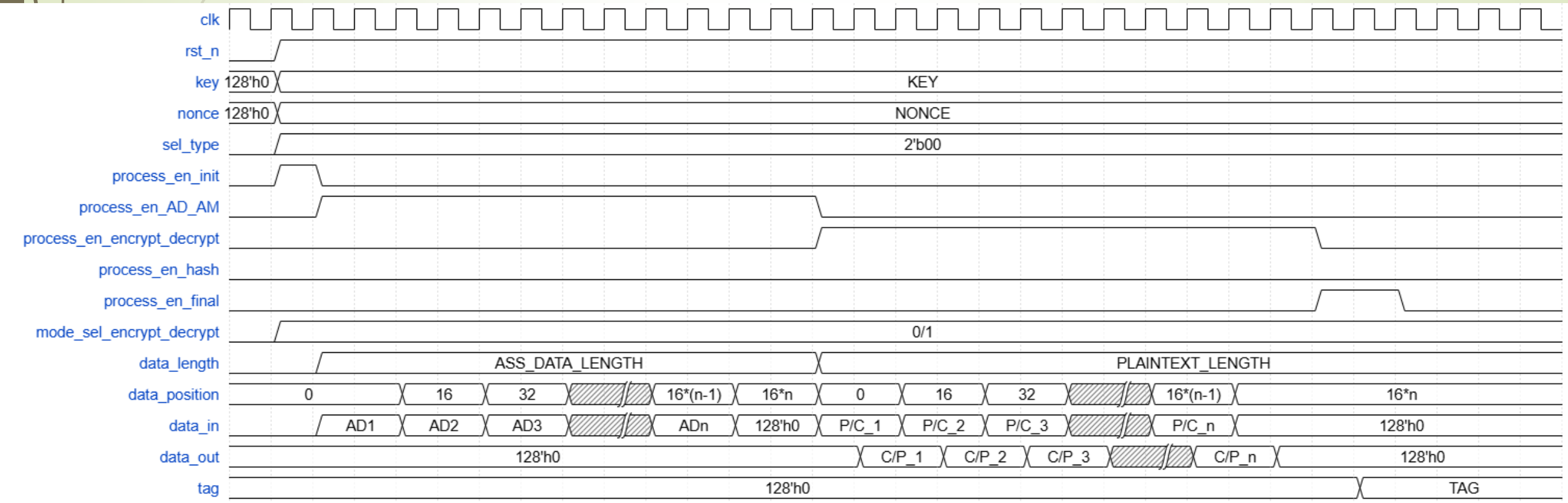
2. Architecture
2.3.1. Ascon Core (waveform – normal case)

Normal case is when length of data need to be is not a multiple of 16 bytes for AEAD mode and 8 byte for Hash mode



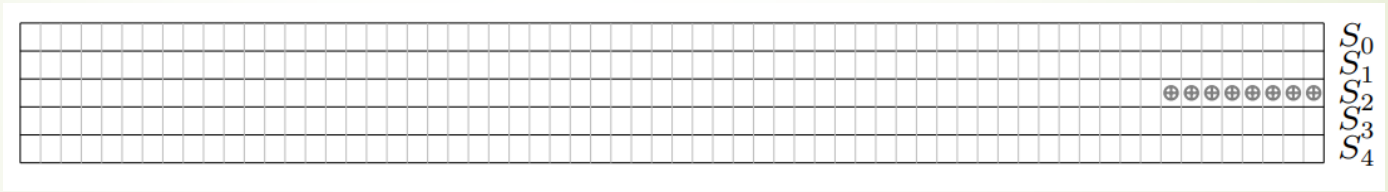
2. Architecture
2.3.1. Ascon Core (waveform – special case)

Special case is when length of data need to be is a multiple of 16 bytes for AEAD mode and 8 byte for Hash mode

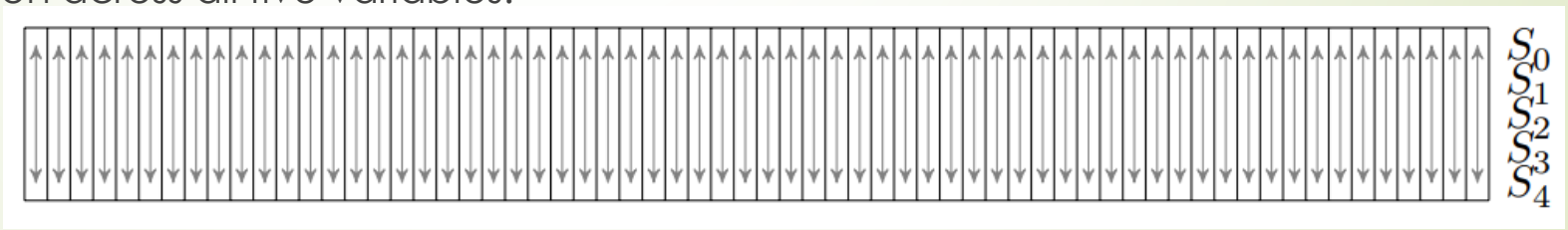


2. Architecture
2.3.2. Ascon permutation (Ascon-p)

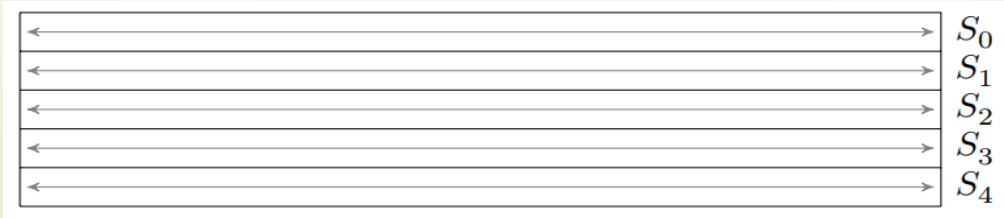
- The Ascon permutation (Ascon-p) is a module that permutes five 64-bit variables, totaling 320 bits. It operates in three main steps:
- **Add round constant** – XOR the round constant with S2.



- **Bitwise permutation across variables** – Apply a specific transformation to bits at the same position across all five variables.

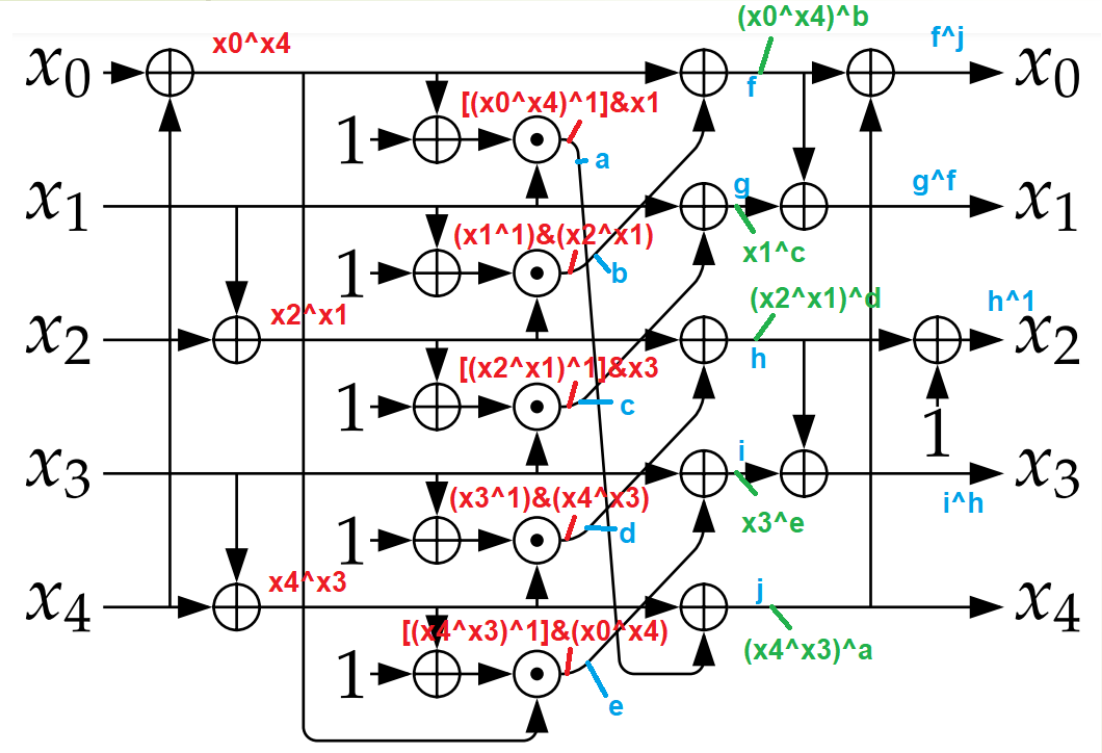


- **Bitwise permutation within a variable** – Rearrange bits within each variable independently.



2. Architecture
2.3.2. Ascon permutation (Ascon-p)

Substitution layer (S-box)

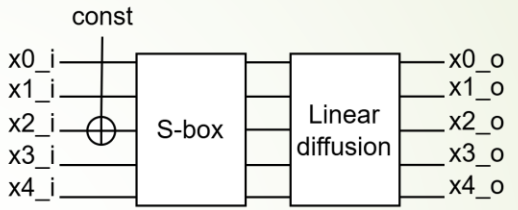


Linear diffusion layer

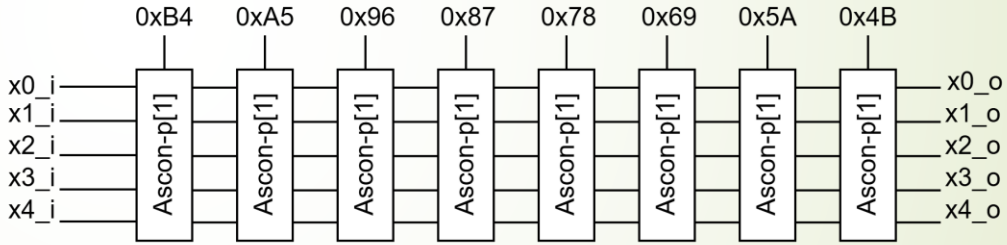
$$x_0 \leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)$$
$$x_1 \leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)$$
$$x_2 \leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6)$$
$$x_3 \leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)$$
$$x_4 \leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)$$

2. Architecture
2.3.2. Ascon permutation (Ascon-p)

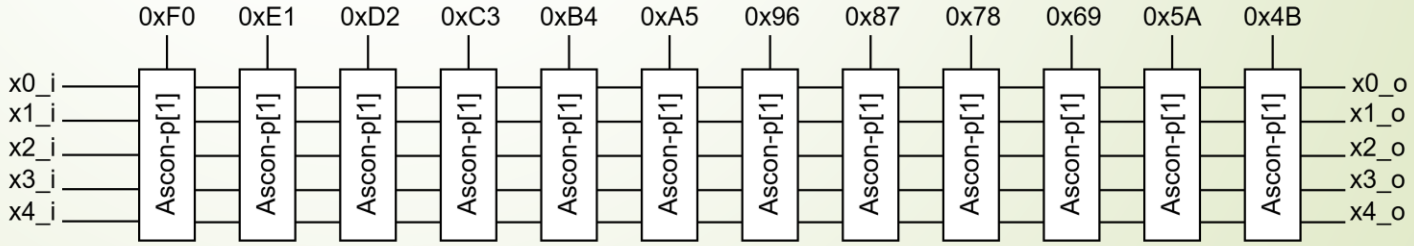
Ascon-p[1]



Ascon-p[8]

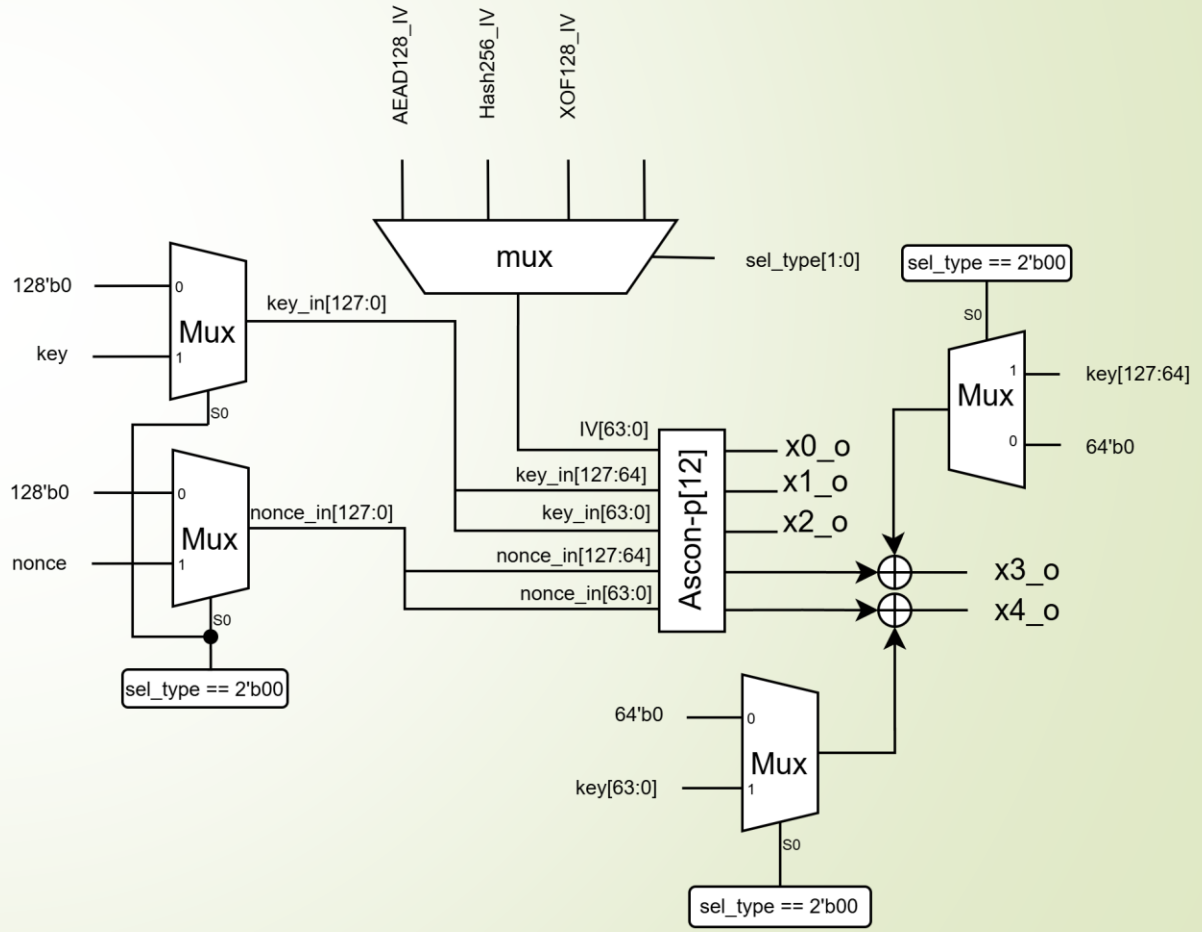


Ascon-p[12]



2. Architecture
2.3.3. Ascon initialization (Ascon-init)

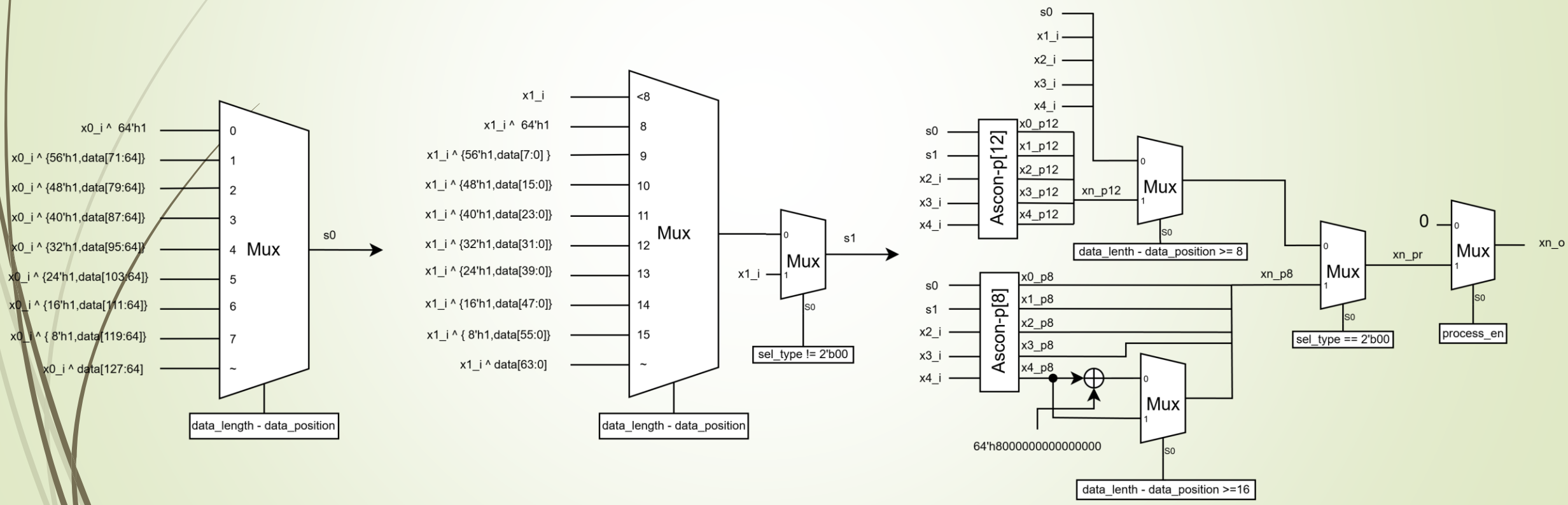
- The idea behind the **Ascon-init** hardware block is to provide the correct initialization signals based on the selected encryption type. Depending on the user's choice, control signals are used to determine the encryption mode, ensuring the hardware generates the appropriate initialization values for the selected algorithm.



2. Architecture
2.3.3. Ascon Asocciated Data – Ascon absorb Message (Ascon-AD_AM)

- Ascon Associated Data (AD) is used to ensure data integrity without encryption, such as headers or filenames. Meanwhile, Ascon Absorb Message (AM) processes input data for hashing.
- Since the processing steps for AD and AM are similar, differing only in input data size and the final step, they can be merged into a single hardware block. The AD process handles 128-bit data per cycle, while the AM process handles 64-bit.
- To accommodate varying data lengths, the hardware is designed to process a fixed amount of data per clock cycle, depending on the encryption type, ensuring efficient and scalable processing.

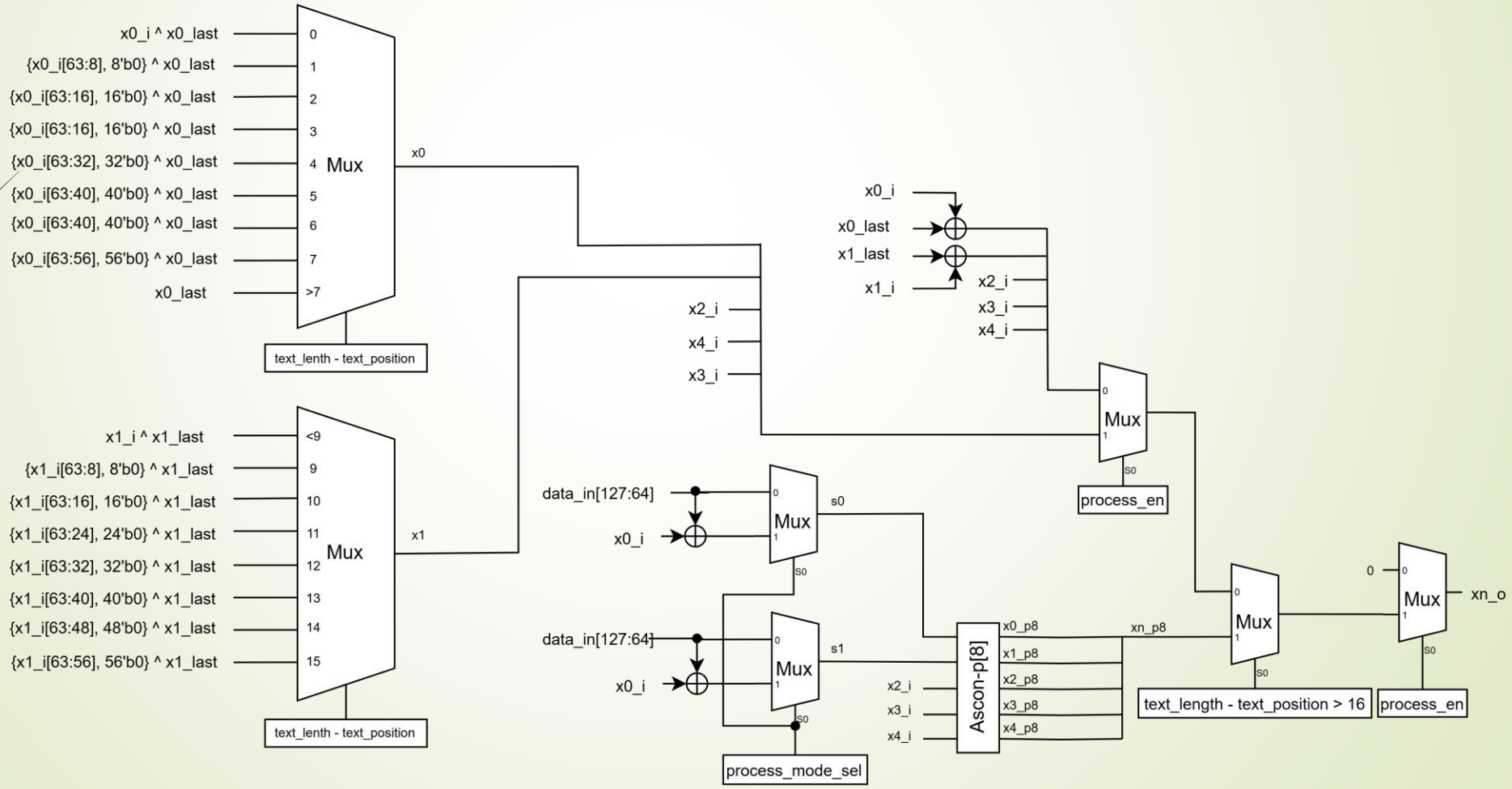
2. Architecture
2.3.3. Ascon Asocciated Data – Ascon absorb Message (Ascon-AD_AM)



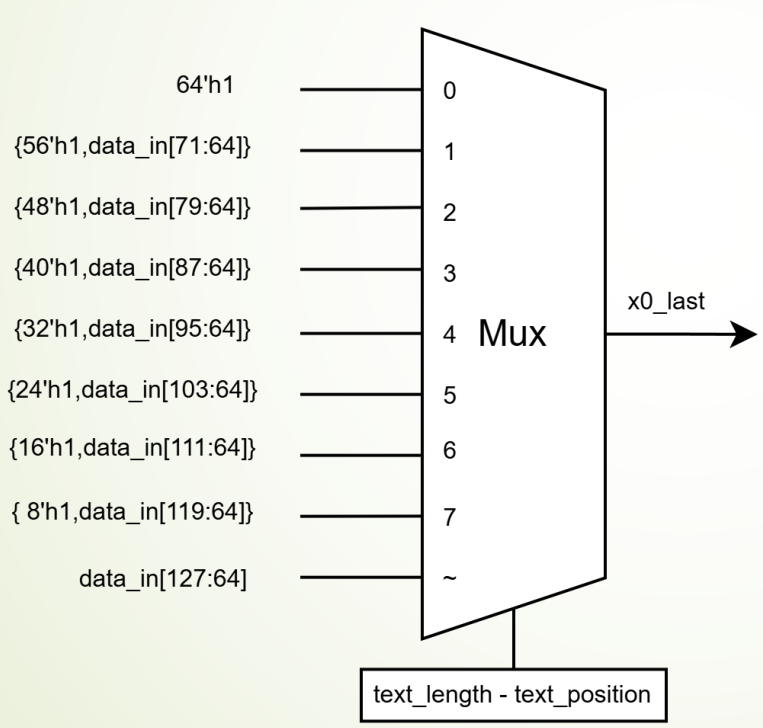
2. Architecture
2.3.3. Ascon Encrypt Decrypt (Ascon-Encrypt_decrypt)

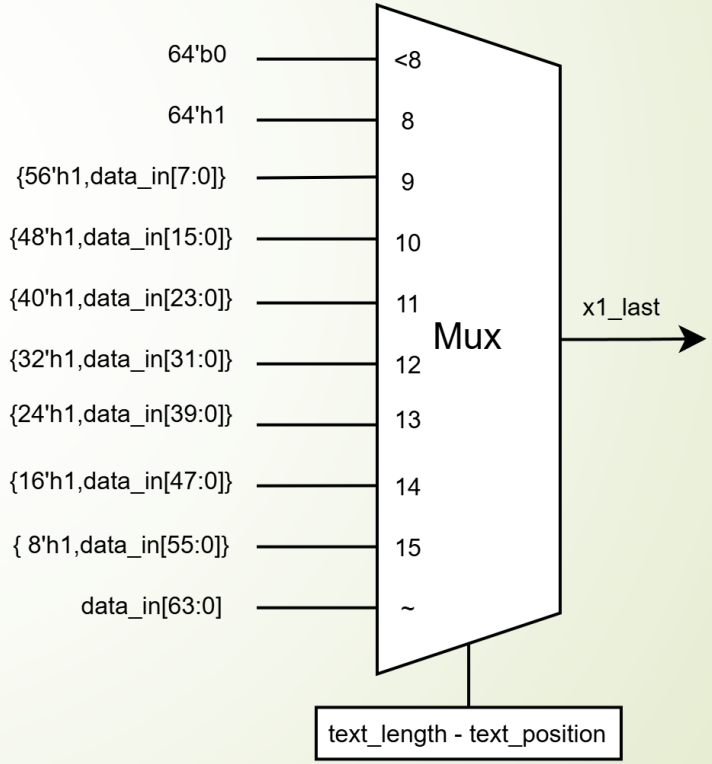
- The Ascon Encryption-Decryption block is responsible for encrypting or decrypting data in the Ascon-AEAD128 function. Its iterative processing approach is similar to the Ascon-AD_AM block.
- Since the encryption and decryption algorithms share significant similarities, they can be combined into a single hardware block. In this design, the data_in for encryption is the plaintext, and the data_out is the ciphertext. Conversely, for decryption, the data_in is the ciphertext, and the data_out is the plaintext.

2. Architecture
2.3.3. Ascon Encrypt Decrypt (Diagram[1])

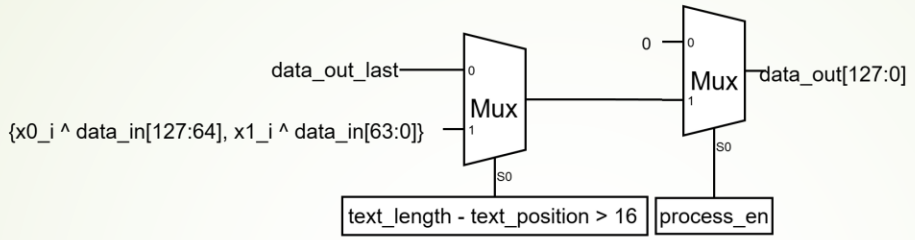


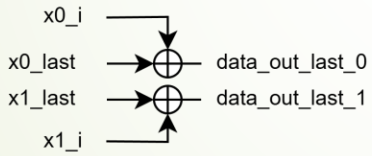
2. Architecture
2.3.3. Ascon Encrypt Decrypt (Diagram[2])

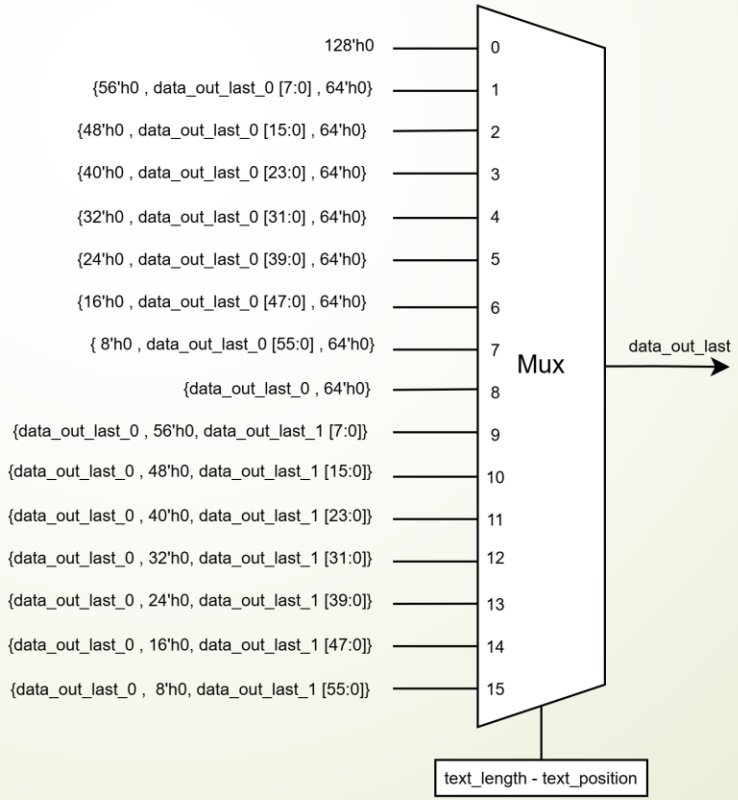




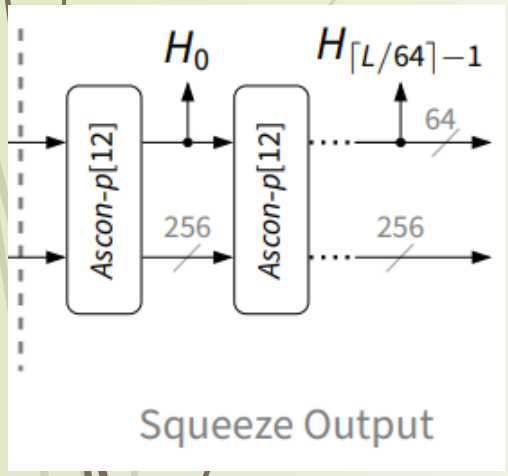
2. Architecture
2.3.3. Ascon Encrypt Decrypt (Diagram[3])







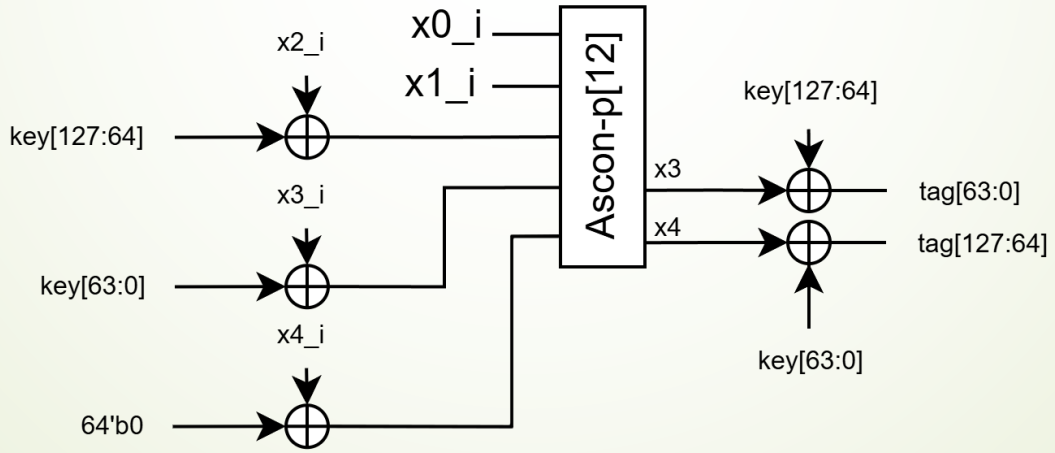
2. Architecture
2.3.3. Ascon Hash(Ascon-Hash)



- The Ascon Hash block processes data hashing, producing an x0 output of 64 bits after each Ascon-p[12] permutation.
- For Ascon-Hash256, hashing is performed four times, each producing 64-bit outputs, resulting in a total 256-bit (32-byte) hash. For Ascon-XOF128, the number of hashing iterations depends on how many times the CPU requests, with each iteration producing 64 bits.
- Since the encryption and decryption algorithms share significant similarities, they can be combined into a single hardware block. In this design, the data_in for encryption is the plaintext, and the data_out is the ciphertext. Conversely, for decryption, the data_in is the ciphertext, and the data_out is the plaintext.

2. Architecture
2.3.3. Ascon Finalization(Ascon-final)

- The Finalization module generates the Tag, which ensures data integrity during encryption, decryption, and verification. It requires a secret key shared between both processes. This block is only used in AEAD mode.



3. Result

- The cryptographic processing speed can reach 0.125 cycles/byte in AEAD mode and 0.25 cycles/byte in Hash mode.

Data length (byte)	AEAD mode	Hash mode
64	15	27
512	71	139
1K	135	267
1M	131079	262155