ASCON

Rithvika Pervala

Indian Institute of Technolgy, Bhilai, 11940830, pervalar@iitbhilai

Abstract. In this paper we explore ASCON - its design and hardware implementation as well as its cryptanalysis automated analysis implementation. With regards to design, we focus through the construction of the Mode - Authenticated Encryption and the Permutation Components. In cryptanalysis, we present distinguishers from Zero-Sum, Linear and Differential cryptanalysis - truncated, impossible differentials. The Hardware Implementation of the Permutation is presented in Verilog along with Area and Component Analysis. Automated Analysis using basic MILP as well as including Logical Conditional Modelling is covered in this paper.

Keywords: ASCON · AEAD · Cryptanalysis · Verilog Implementation · Automated Analysis

1 Cipher Construction

ASCON cipher suite consists of authenticated encryption with associated data (AEAD) and hashing functionality. Authenticated encryption consists of the authenticated ciphers Ascon-128 and Ascon-128a, which vary on the number of rounds in the core permutation as well as in the rate and capacity. Apart from that ASCON Cipher suite also provides the hash functions Ascon-Hash and Ascon-Hasha, as well as the Extendable Output Functions (XOFs) Ascon-Xof and Ascon-Xofa. We will however focus mainly on the Authenticated Encryption Mode but more specifically the permutation of the ASCON. All the mode schemes have 128-bit security and the permutation is run over a 320 bit state. Authentication Encryption in specific is also designed to work through single pass - not needing to run twice to get the Authentication Tag as well as online - not needing to know the entire Plaintext or Associated Data to start running the algorithm.

The Permutation is design to have several features and complement over the balance of high security, performance and robustness. Ascon permutation operations are all run over 64 bit words making it fast for 64-bit platforms, while bitsliced implementation of s-box can easily be parallelized and run faster in 32-bit, 16-bit, 8-bit platforms. Ascon uses S-box with low algebraic degree and other less complex factors which allows for a small area usage.

1.1 Cipher Specifications

Recommended parameters for ASCON authenticated encryption

Cipher	Bit size of					Rounds	
	key	nonce	tag	rate	capacity	p^a	p^b
ASCON-128	128	128	128	64	256	12	6
ASCON-128a	128	128	128	128	192	12	8

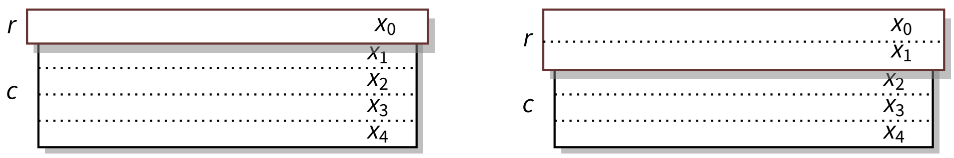
Figure 1: Authentication Encryption Specifications

1.2 State Design

ASCON uses a 320-Bit State which can be divided into 5 - 64-Bit words - x_0, x_1, x_2, x_3, x_4 namely. The Least Significant Bit is the Right most Bit of x_4 and the Most Significant is the Left most bit of x_0. It follows a reverse row indexing of the bits.

Table 1: $S = x_0||x_1||x_2||x_3||x_4$

Apart from that the state can also be divided into Rate Part S_r and the capacity part S_c the following way

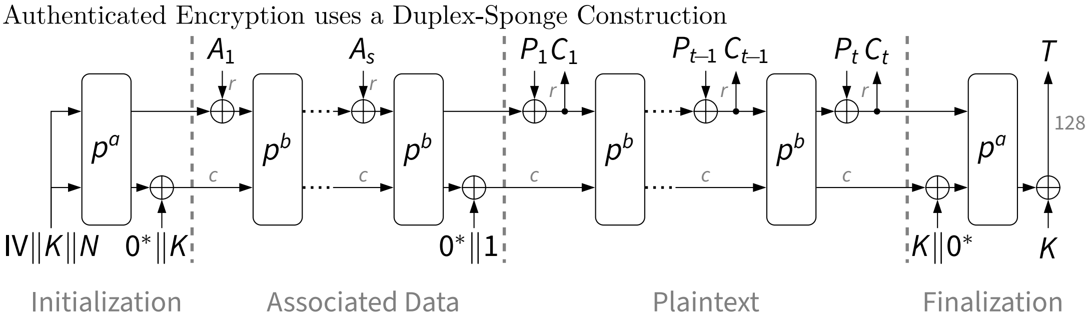


(a) 64-Bit Rate (b) 128-Bit Rate

Figure 2: $S = S_r||S_c$

1.3 Authenticated Encryption

Authenticated Encryption uses a Duplex-Sponge Construction



1.3.1 Initialization

Initialization vectors is of 64 bits and it depends on the key size (k), rate size (r), end permutation rounds (a), and the core permutation rounds (b) - each filling up 8 bits and the rest being padded the following way

$$IV_{k,r,a,b} \leftarrow k||r||a||b||0^{160-k}$$
$$\text{ASCON-128} = -\texttt{80400c0600000000}$$
$$\text{ASCON-128a} - \texttt{80800c0800000000}$$

The State is then filled with the IV in x_0, Key next in x_1, x_2 and the Nonce in the remaining x_3, x_4

$$S \leftarrow IV||K||V$$

The State is then run through the end permutation p^a after being added with a padded key the following way

$$S \leftarrow p^a(S) \oplus \left(0^{320-k}||K\right)$$

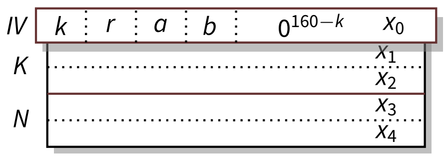


Figure 3: Initialization State

1.3.2 Associated Data Processing

After initialization we process the Associated Data by first padding the Associated Data A and splitting it into s r-Bit Blocks the following way

$$A_1, \ldots, A_s \leftarrow A||1||0^{r-1-(|A| \bmod r)}$$

Then each Associated Data Block A_i is digested by injecting it into the core permutation

$$S \leftarrow p^b\left((S_r \oplus A_i)||S_c\right), \ 1 \leq i \leq s$$

After all the Blocks are digested we add a 1-bit Domain Separation Constant to add security over attacks that swap Associated Data and Plaintext

$$S \leftarrow S \oplus \left(0^{319}||1\right)$$

1.3.3 Encryption

After Associated Data Processing the Plaintext P is processed by first and padding it splitting into t r-Bit Blocks the following way

$$P_1, \ldots, P_t \leftarrow P||1||0^{r-1-(|P| \bmod r)}$$

Then Plaintext Block P_i is injected into the core permutation after which the corresponding Ciphertext Block C_i is extracted from the rate part.

$$C_i \leftarrow S_r \oplus P_i \ \ 1 \leq i \leq t$$
$$S \leftarrow \begin{cases} p^b\left(C_i||S_c\right) & \text{if } 1 \leq i < t \\ C_i||S_c & \text{if } i = t \end{cases}$$

The last Ciphertext C_t is unpadded to have the same size as the last Plaintext Block P_t before padding so that the whole Plaintext and Ciphertext have same size

$$C_t' \leftarrow \lfloor C_t \rfloor_{|P| \bmod r}$$

1.3.4 Finalization

In the finalization phase we first add a Padded Key to State

$$S \leftarrow S \oplus \left(0^r || K || 0^{c-k}\right)$$

It is then run through the End Permutation

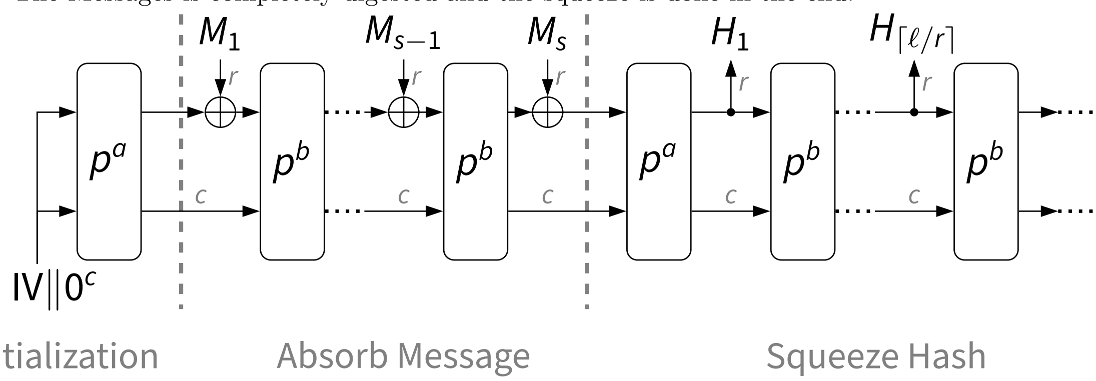
$$S \leftarrow p^a(S)$$

The Tag T is extracted from the Least Significant 128 Bits after adding Key

$$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$$

1.4 Hash

The hash construction is similar to AEAD except it doesn't use Duplex-Sponge Construction. The Messages is completely digested and the squeeze is done in the end.



1.5 Permutation

Permutation consists of three components run one after the other - Round Constant p_C, S-Box p_S, Linear Diffusion Layer p_L

$$p = p_L \circ p_S \circ p_C$$

1.5.1 Round Constant (p_C)

The Round Constant is essentially adding a 64-bit round constant register c_r on x_2. There are 12 registers and the register to be applied at a specific round r depends on the following formula

$$p^a \rightarrow c_r \text{ and } p^b \rightarrow c_{a-b+r}$$

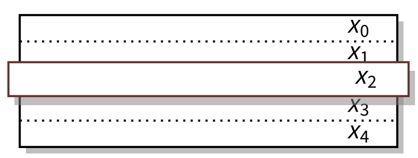


Figure 4: c_r on x_2

Table 2: Round Constant Registers

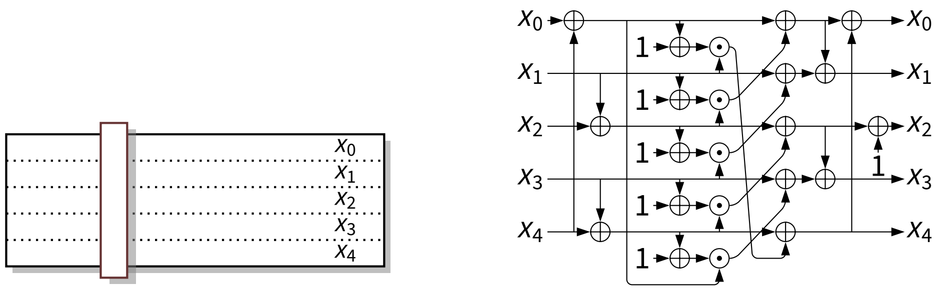
p^{12}	p^8	p^6	Constant	p^{12}	p^8	p^6	Constant
0			000000000000000000f0	6	2	0	00000000000000000096
1			000000000000000000e1	7	3	1	00000000000000000087
2			000000000000000000d2	8	4	2	00000000000000000078
3			000000000000000000c3	9	5	3	00000000000000000069
4	0		000000000000000000b4	10	6	4	0000000000000000005a
5	1		000000000000000000a5	11	7	5	0000000000000000004b

1.5.2 S-Box (p_S)

The below 5-Bit S-box is applied in a Bit Sliced Manner across $(x_0, x_1, x_2 x_3, x_4)$ with x_0 being the Most Significant and x_4 being the Least Significant. 64 parallel applications can hence be done over Bit-Sliced S-box

Table 3: 5-Bit ASCON S-Box

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$S(x)$	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17



(a) Bit-Sliced Application

(b) ANF Form

1.5.3 Linear Layer

Linear Layer is applied within each word w_i with different Linear Functions $\Sigma_i(x_i)$ with essentially linearly combines each word with their right circular rotations the following way.

$$x_i \leftarrow \Sigma_i(x_i) \quad 0 \leq i \leq 4$$

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

2 Cryptanalysis

2.1 S-Box Analysis

2.1.1 DDT

The Differential Branch Number is 3 and the Differential Uniformity of ASCON S-box is 8. Hence the maximum Differential Probability of the S-Box is 2^{-2}

Table 4: $\mathrm{DDT}[\Delta_i, \Delta_o] = |\{x : S(x \oplus \Delta_i) \oplus S(x) = \Delta_o\}|$

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	32	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	·	·	·	·	·	·	4	·	4	·	4	·	4	·	·	·	·	·	·	·	·	4	·	4	·	4	·	4	·
2	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	4	·	4	·	4	·	4	·	4	·	4	·	4	·	4
3	·	4	·	·	·	4	·	·	·	4	·	·	·	4	·	·	4	·	·	·	4	·	·	·	4	·	·	·	4	·	·	·
4	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·
5	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	4	·	4	4	·	4	·	4	·	4	·	·	4	·	4
6	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2
7	·	·	4	4	·	·	4	4	·	·	4	4	·	·	4	4	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4
9	·	2	·	2	2	·	2	·	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	·	2	·	2	2	·	2	·
10	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2
11	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2
12	·	8	·	·	·	·	·	·	8	·	·	·	·	·	·	·	8	·	·	·	·	·	·	·	·	8	·	·	·	·	·	·
13	·	2	·	2	·	2	·	2	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	·	2	·	2	·	2	·	2
14	·	4	4	·	4	·	·	4	·	·	·	·	·	·	·	·	·	4	4	·	4	·	·	4	·	·	·	·	·	·	·	·
15	·	·	·	·	·	·	·	·	4	4	·	·	4	4	·	·	·	·	·	·	·	·	·	·	4	4	·	·	4	4	·	·
16	·	·	·	·	·	·	·	·	·	8	·	8	·	·	·	·	·	·	·	·	·	·	·	·	8	·	8	·	·	·	·	·
17	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	8	·	8	·	8	·	8	·	·	·	·	·	·	·	·
18	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·
19	·	·	8	·	8	·	·	·	·	·	8	·	8	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
20	·	·	·	·	4	4	4	4	·	·	·	·	4	4	4	4	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
21	·	·	·	·	·	4	·	4	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	·	·	·	·	4	·	4
22	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
23	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·
24	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2
25	·	·	·	4	·	·	4	·	4	·	·	·	·	4	·	·	4	·	·	·	·	4	·	·	·	·	·	4	·	·	4	·
26	·	2	2	·	·	2	2	·	2	·	·	2	2	·	·	2	·	2	2	·	·	2	2	·	2	·	·	2	2	·	·	2
27	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·
28	·	4	·	4	·	·	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	·	4	·	4	·	·	·	·
29	·	·	·	4	·	4	·	·	4	·	·	·	·	·	4	·	4	·	·	·	·	·	4	·	·	·	·	4	·	4	·	·
30	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2
31	·	·	4	4	4	4	·	·	·	·	·	·	·	·	·	·	·	·	4	4	4	4	·	·	·	·	·	·	·	·	·	·

Table 5: DDT Frequency Analysis

$f(\Delta_i, \Delta_o)$	S
0	707
2	176
4	120
6	0
8	20

2.1.2 LAT

Linear Branch Number is 3 and Maximum Absolute Linear Bias of the ASCON S-Box is 8. Hence the The Maximum Linear bias of the S-box is 2^{-2}

Table 6: $\mathrm{LAT}[\alpha, \beta] = |\{x : \alpha^T \cdot x = \beta^T \cdot S(x)\}| - 16$

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	·	·	·	8	·	·	4	4	·	·	-4	4	·	·	·	4	4	·	·	4	-4	4	·	-4	·	-4	·	-4	·
2	·	·	·	·	·	·	-8	8	·	·	4	4	·	·	4	4	·	·	4	4	·	·	-4	-4	·	·	·	·	·	·	·	·
3	·	8	·	·	·	·	·	·	·	4	·	4	·	4	·	-4	-8	·	·	·	·	·	·	·	4	·	4	·	4	·	-4	·
4	·	·	·	4	·	-4	·	·	·	·	4	·	·	4	-4	-4	·	·	4	·	-4	·	·	·	·	-8	·	-4	-4	·	4	-4
5	·	·	·	4	·	4	·	·	·	-4	·	·	·	·	·	-4	·	·	·	-4	4	·	-4	-4	4	·	-4	4	·	-8	·	-4
6	·	·	·	4	·	-4	·	·	·	·	·	-4	·	4	·	·	·	·	·	-4	-4	·	-4	-4	·	8	·	-4	-4	·	-4	4
7	·	·	·	-4	·	-4	·	·	·	4	4	4	·	·	-4	·	·	·	-4	·	-4	·	·	·	-4	·	-4	4	·	-8	·	4
8	·	·	·	·	·	·	·	·	·	·	4	4	·	·	-4	-4	·	·	·	·	·	·	·	·	·	8	-4	4	·	8	4	-4
9	·	·	·	·	·	·	·	-8	·	-4	·	4	·	4	·	4	·	·	4	4	·	·	-4	4	4	·	·	4	-4	·	·	4
10	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	4	4	·	·	4	4	·	8	4	-4	·	-8	4	-4
11	·	8	·	·	·	·	·	·	·	-4	4	·	·	-4	-4	·	8	·	·	·	·	·	·	·	4	·	·	-4	4	·	·	4
12	·	·	-8	4	-8	-4	·	·	·	·	·	4	·	-4	·	·	·	·	-4	·	4	·	·	·	·	·	4	·	-4	·	·	·
13	·	·	·	-4	-8	4	·	·	·	4	-4	-4	·	·	-4	·	·	·	·	4	-4	·	-4	-4	4	·	·	·	·	·	4	·
14	·	·	·	-4	8	-4	·	·	·	·	-4	·	·	-4	-4	-4	·	·	·	4	4	·	-4	-4	·	·	4	·	-4	·	·	·
15	·	·	8	-4	-8	-4	·	·	·	-4	·	·	·	·	·	-4	·	·	4	·	4	·	·	·	-4	·	·	·	·	·	-4	·
16	·	·	·	·	·	·	-8	·	·	4	·	-4	-4	·	-4	·	·	·	·	·	4	-4	4	4	4	·	-4	·	-4	·	-4	·
17	·	·	·	·	·	·	·	·	-8	·	-4	4	-4	-4	·	·	·	8	4	-4	-4	-4	·	·	·	·	·	·	·	·	·	·
18	·	-8	·	·	·	·	·	·	·	-4	4	·	-4	·	·	-4	·	·	-4	4	-4	-4	·	·	4	·	4	·	4	·	-4	·
19	·	·	·	·	·	·	-8	-8	·	·	·	·	4	-4	4	-4	·	·	·	·	-4	4	4	-4	·	·	·	·	·	·	·	·
20	·	·	·	4	·	4	·	·	·	4	4	-4	-4	-4	·	-4	·	·	4	·	·	4	-4	4	-4	·	4	4	·	·	·	4
21	·	·	·	4	·	-4	·	·	·	·	·	-4	4	·	-4	4	·	8	·	4	·	4	·	·	·	·	·	4	4	·	-4	-4
22	·	·	·	-4	·	-4	·	·	·	4	·	·	-4	4	4	·	8	·	·	-4	·	4	·	·	4	·	4	4	·	·	·	-4
23	·	·	·	4	·	-4	·	·	8	·	-4	·	-4	·	·	·	·	·	4	·	·	-4	4	-4	·	·	·	4	4	·	4	4
24	·	·	·	·	·	·	·	-8	·	4	4	·	-4	·	·	4	·	·	·	·	4	-4	-4	-4	-4	·	·	-4	4	·	·	-4
25	·	·	·	·	·	·	·	·	·	·	·	·	4	-4	-4	4	·	-8	4	-4	-4	-4	·	·	·	·	4	4	·	·	-4	-4
26	·	8	·	·	·	·	·	·	·	-4	·	-4	-4	·	4	·	·	·	-4	4	-4	-4	·	·	-4	·	·	4	-4	·	·	-4
27	·	·	·	·	·	·	·	·	8	·	-4	4	-4	-4	·	·	·	·	·	·	-4	4	-4	4	·	·	-4	-4	·	·	-4	-4
28	·	·	8	4	·	-4	·	·	·	4	·	·	4	-4	4	·	·	·	-4	·	·	-4	-4	4	4	·	·	·	·	·	4	·
29	·	·	·	-4	·	4	·	·	8	·	4	·	4	·	·	·	·	8	·	-4	·	-4	·	·	·	·	4	·	-4	·	·	·
30	·	·	·	4	·	4	·	·	·	4	-4	4	4	4	·	-4	8	·	·	4	·	-4	·	·	-4	·	·	·	·	·	-4	·
31	·	·	8	4	·	4	·	·	·	·	·	4	-4	·	-4	4	·	·	-4	·	·	4	4	-4	·	·	4	·	-4	·	·	·

Table 7: LAT Frequency Analysis

ϵ	S
-8	18
-4	174
0	647
4	162
8	22

2.1.3 BCT

Boomerang Uniformity is 16

Table 8: $\mathrm{BCT}[\Delta_i, \nabla_o] = |\{S^{-1}(S(x) \oplus \nabla_o) \oplus S^{-1}(S(x \oplus \nabla_o) \oplus \Delta_i) = \Delta_i\}|$

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
0	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
1	32	·	·	·	·	·	8	·	·	4	·	4	·	4	8	4	·	4	·	4	4	·	12	·	8	·	8	·	4	4	12	4
2	32	12	4	·	4	·	·	4	8	·	·	·	·	·	·	·	8	8	4	4	4	4	·	8	·	12	·	4	·	4	·	4
3	32	4	8	·	8	4	·	·	·	12	8	8	8	4	·	·	4	·	·	·	4	·	·	·	12	·	8	·	4	·	·	·
4	32	·	·	·	4	8	12	8	·	16	·	16	4	8	12	8	·	16	·	16	4	8	12	8	16	·	16	·	4	8	12	8
5	32	·	·	·	·	·	8	·	·	4	·	4	·	4	8	4	·	4	·	4	4	·	12	·	8	·	8	·	4	4	12	4
6	32	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2
7	32	·	8	4	4	·	4	4	·	8	8	12	4	·	4	4	·	·	4	·	4	·	·	·	8	·	12	·	4	·	·	·
8	32	·	·	4	·	·	8	4	4	·	·	·	·	4	4	4	4	8	·	8	·	12	4	12	·	·	·	4	·	·	8	4
9	32	2	·	2	2	·	2	·	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	·	2	·	2	2	·	2	·
10	32	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2	·	2	2	·	2	·	·	2
11	32	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2	·	·	2	2
12	32	12	16	8	16	8	·	4	12	·	8	·	8	·	4	·	12	16	8	16	8	16	4	16	·	12	·	8	·	8	·	4
13	32	2	·	2	·	2	·	2	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·	·	2	·	2	·	2	·	2
14	32	12	4	·	4	·	·	4	8	·	·	·	·	·	·	·	8	8	4	4	4	4	·	8	·	12	·	4	·	4	·	4
15	32	4	8	4	8	·	·	·	8	4	12	·	12	4	·	·	4	·	4	·	·	·	·	·	4	8	·	4	4	4	·	·
16	32	4	16	4	16	8	16	8	·	12	16	12	16	8	16	8	4	·	4	·	8	·	8	·	12	·	12	·	8	·	8	·
17	32	8	·	8	·	8	16	8	16	4	·	4	·	4	16	4	16	12	·	12	·	12	16	12	·	8	·	8	·	8	16	8
18	32	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	2	·	2	·	2	·	2	·	2	·	2	·	2	·	2	·
19	32	16	12	8	12	8	·	·	16	16	12	8	12	8	·	·	16	·	8	4	8	4	·	·	16	16	8	4	8	4	·	·
20	32	·	·	·	4	4	12	4	·	8	·	8	4	4	12	4	·	·	·	·	·	·	8	·	8	·	8	·	·	·	8	·
21	32	·	·	·	·	4	8	4	·	4	·	4	·	·	8	·	·	12	·	12	·	8	8	8	·	·	·	·	·	4	8	4
22	32	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
23	32	·	8	4	4	·	4	4	·	8	8	12	4	·	4	4	·	·	4	·	4	·	·	·	8	·	12	·	4	·	·	·
24	32	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2
25	32	·	·	4	·	·	8	4	4	·	·	·	·	4	4	4	4	8	·	8	·	12	4	12	·	·	·	4	·	·	8	4
26	32	2	2	·	·	2	2	·	2	·	·	2	2	·	·	2	·	2	2	·	·	2	2	·	2	·	·	2	2	·	·	2
27	32	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·	·	·	2	2	2	2	·	·
28	32	4	8	4	8	·	·	·	8	4	12	·	12	4	·	·	4	·	4	·	·	·	·	·	4	8	·	4	4	4	·	·
29	32	8	·	4	·	4	·	·	12	·	·	·	·	·	4	·	12	8	·	8	·	8	4	8	·	8	·	4	·	4	·	·
30	32	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2	·	·	·	·	·	·	·	·	2	2	2	2	2	2	2	2
31	32	8	12	4	12	4	·	·	8	·	8	·	8	·	·	·	8	·	4	4	4	4	·	·	·	8	·	·	·	·	·	·

Table 9: BCT Frequency Analysis

$f(\Delta_i, \nabla_o)$	S
0	445
2	176
4	150
8	110
12	50
16	30

2.1.4 BDT

Boomerang Differential Table (BDT) combines DDT and BCT to find converging points that adhere to both Differential and Boomerang Conditions.

$$\text{DDT}[\Delta_i, \Delta_o, \nabla_o] = |\{x : S(x \oplus \Delta_i) \oplus S(x) = \Delta_o \& S^{-1}(S(x) \oplus \nabla_o) \oplus S^{-1}(S(x \oplus \nabla_o) \oplus \Delta_i) = \Delta_i\}|$$

Table 10: BDT Frequency Analysis

$f(\Delta_i, \Delta_o, \nabla_o)$	S
0	31624
2	352
4	720
8	40
32	32

2.2 Differential and Linear Cryptanalysis

The Differential and Linear Branch Numbers of the Linear Diffusion Layer Σ_i is 4. The number of active S-box after 3 rounds in both Differential and Linear Trails are 15 and 13 respectively. Using heuristics, one can figure out the minimum number of active S-boxes for both differential and linear trails are calculated for upto 5 rounds and the results are the following

Table 11: Minimum Number of Active Sboxes

Rounds (R)	1	2	3	4	5
Minimum # Active Sboxes (Differential)	1	4	15	≤ 44	≤ 78
Minimum # Active Sboxes (Linear)	1	4	13	≤ 43	≤ 67

We can add additional constraints on the differences, for instance we can constrain the differences - both input and output - to be restricted only to the rate part of the state (S_r). This might be useful in forgery attacks in the Authenticated Encryption Scheme. However, it might not be that fruitful as the Differential Properties of Permutations are quite secure.

$$\Delta_i^r \xrightarrow{p^b} \Delta_o^r$$

2.3 Truncated Differentials

For a specific input difference Δ_i of an S-box, if some bits of the output difference Δ_o^* remain invariant, then we call such bits **undisturbed**.

$$Pr\left[\Delta_i \xrightarrow{S} \Delta_o^*\right] = 1$$

For ASCON S-Box, due to its low algebraic degree ($d(S) = 2$) has found to have around 23 such undisturbed bits which are shown below as follows. The Inverse S-box however has found to have only 2 undisturbed bits as it algebraic degree being higher $\left(d\left(S^{-1}\right) = 3\right)$

Table 12: Undisturbed Bits

Δ_i	Δ_o^*	Δ_i	Δ_o^*	Δ_i	Δ_o^*	Δ_i	Δ_o^*
00001	*1***	10000	*10**	00110	****1	10111	****0
00010	1***1	10001	10**1	00111	0**1*	11000	**1**
00011	***0*	10011	0***0	01000	**11*	11100	**0**
00100	**110	10100	0*1**	01011	***1*	11110	*1***
00101	1****	10101	****1	01100	**00*	11111	*0***
01111	*1*0*	10110	1****	01110	*0***		

Since there are so many available undisturbed bits with probability 1, we exhaust all of them to find the longest possible Truncated Differential. One such undisturbed bits $\left[\texttt{10011} \xrightarrow{S} \texttt{0***0}\right]$ has found to generate a 3.5 Round Truncated Differential as follows

Table 13: 3.5 Round Truncated Differential Distinguisher

	1000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
I	0000000000000000000000000000000000000000000000000000000000000000
	1000000000000000000000000000000000000000000000000000000000000000
	1000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	*000000000000000000000000000000000000000000000000000000000000000
S_1	*000000000000000000000000000000000000000000000000000000000000000
	*000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	*00000000000000000000000000000000000000*000000000000000000000*00
P_1	**0000*000000000000000000000000000000000000000000000000000000000
	*000000000*000000*0000000000000000000000000000000000000000000000
	0000000000000000000000000000000000000000000000000000000000000000
	**0000*000*000000*000000000000000000000*000000000000000000000*00
	**0000*000*000000*000000000000000000000*000000000000000000000*00
S_2	**0000*000*000000*000000000000000000000*000000000000000000000*00
	**0000*000*000000*000000000000000000000*000000000000000000000*00
	*000000000*000000*000000000000000000000*000000000000000000000*00
	0*00*000*000000**0000*00**0000*0*0**00000*000000000000*00*00
	0*0000*000*00*000000000000000000*00**0000*000*000000*0*00**0
P_2	****00**00***000***0000*000000000000000**0000*000000000000000**0
	000000**00*0***0*00*000*000000*0000*000000000*000000*0000*00
	*000*00*00*00000**000000*0000000000000**0*0000*0000*000000*00*00
	*****0**00***0*0*****00***0***0000*0*0****000**00*0*0000*0*00**0
	*****0**00***0*0*****00***0***0000*0*0****000**00*0*0000*0*00**0
S_3	*****0**00***0*0***0*00**00*000000*0*0****000**00*0*0000*0*00**0
	*****0**00***0*0*****00***0***0000*0*0****000**00*0*0000*0*00**0
	00**00**00*0*****00***0***0000*0*0****000**00*0*0000*0*00**0
	********0***0***0********0
	*****0************************0*0***********0**0****0*0*****0***
P_3	*******************************00**********0****0****00*****0***
	*****0**************************0**********0***0****0*0*****0***
	*********0**0************************0************0***0*********
	0*******
	0*******
S_4	**0*********
	0*******
	0*******

2.4 Impossible Differentials

Using automation tools, 5 Round Impossible Trails of the ASCON Permutation are found. One of which is as follows

Table 14: 5 Round Impossible Trail

x_0	0000000000000000		0100000000100002
x_1	0000000000000000		0000000000000000
x_2	0000000000000000	$\not\to$	0000000000000000
x_3	0000000000000000		0000000000000000
x_4	8000000000000000		0000000000000000

However, for a random permutation the above 5 round impossible differential would require a probability of $p = 2^{320}$. Hence to use it as a distinguisher, a whole codebook is required. It also cannot be used in key recovery or attacks like forgery as the Output Differences are fully specified.

A combination of 3.5 Truncated Distinguisher in the forward direction and a new Impossible Trail can be used in the backward direction however to generate a Truncated Impossible Differential in what is known as **miss-in-the-middle-technique**. To get the Round Differential, we will need to use Inverse S-box and Inverse Linear Diffusion. Using (*Rivest, 2011*) Theorem of finding Inverse of Rotations, we can find inverses of the ASCON linear functions Σ_i as they have odd number of terms ($k = 3$) in the linear layer. Hence using that theorem, here are the terms in both the linear functions Σ_i and their inverses Σ_i^{-1}

Table 15: Linear Function and Inverse Terms

Function	Rotations	# Terms
Σ_0	0 19 28	3
Σ_0^{-1}	0 3 6 9 11 12 14 15 17 18 19 21 22 24 25 27 30 33 36 38 39 41 42 44 45 47 50 53 57 60 63	31
Σ_1	0 61 39	3
Σ_1^{-1}	0 1 2 3 4 8 11 13 14 16 19 21 23 24 25 27 28 29 30 35 39 43 44 45 47 48 51 53 54 55 57 60 61	33
Σ_2	0 1 6	3
Σ_2^{-1}	0 2 4 6 7 10 11 13 14 15 17 18 20 23 26 27 28 32 34 35 36 37 40 42 46 47 52 58 59 60 61 62 63	33
Σ_3	0 10 17	3
Σ_3^{-1}	1 2 4 6 7 9 12 17 18 21 22 23 24 26 27 28 29 31 32 33 35 36 37 40 42 44 47 48 49 53 58 61 63	31
Σ_4	0 7 41	3
Σ_4^{-1}	0 1 2 3 4 5 9 10 11 13 16 20 21 22 24 25 28 29 30 31 35 36 40 41 44 45 46 47 48 50 53 55 60 61 63	35

Using the Inverse we now built an Impossible trail such that it should not match at atleast one bit in S_4 of 3.5 Round Truncated Differential. However since the Inverse S-box has only 2 undisturbed bits, the generated Differential is of length 1.5 Round only. Here is the generated combined 5-Round Differential (3.5 Forward + 1.5 Backward) which only differs in 1-bit of S_4

Table 16: 5 Round Impossible Truncated Distinguisher

3.5 Truncated Differential Distinguisher	
S_4	**0********* **0********* **0********* **0********* **0*********
1.5 Round Impossible Backward Differential	
S_4	** ** ** 1110000101100010110011111011111101010100101000110100011010100101 **
P_4	0**0*0**0*00*0000**00****0****0***0***00*0*0*00***000*0000*00*0* 0**0*0**0*00*0000**00****0****0***0***00*0*0*00***000*0000*00*0* 0**0*0**0*00*0000**00****0****0***0***00*0*0*00***000*0000*00*0* 0110101101001000011001111011110111011100101010011100010000100101 0**0*0**0*00*0000**00****0****0***0***00*0*0*00***000*0000*00*0*
S_5	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0110101101001000011001111011110111011100101010011100010000100101 0000000000000000000000000000000000000000000000000000000000000000
P_5	0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000 1000000000000000000000000000000000000000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000

2.5 Zero-Sum Distinguisher

Zero-sum distinguishers over round-reduced versions for the Keccak permutation have been analysis across multiple research analyses. Since ASCON's Sbox is based on Keccak S-box χ, we will be using similar methods used in Keccak Distinguisher especially that of Boura et al. Using similar techniques, we are able to construct 12 round distinguishers (essentially the whole end permutation).

Keccak S-box χ covers most of the ASCON S-box Search Criteria, however it has low Differential and Linear branch numbers of 2 whereas ASCON requires 3, a Fix Point at 0 whereas ASCON requires no Fix Points, and all Outputs depend on just 3 Inputs whereas ASCON requires atleast 4 Inputs. Hence basing on χ, ASCON S-box is employs few lightweight affine transformations to the input and output of to check through the search criteria. However, since its just affine transformation, they algebraic properties like algebraic degree of ASCON stays the same as that of χ which is 2. Hence for the permutation the algebraic degree is atmost 2. The inverse S-box, however, has alegraic degree 3 making reverse permutations degree at most 3. We will be exploiting that property of Ascon S-box to yield a Zero-Sum Distinguisher.

$$d(S) = 2 \rightarrow d(p) \leq 2 \rightarrow d(p^r) \leq 2^r$$

$$d(S^{-1}) = 3 \rightarrow d(p^{-1}) \leq 3 \rightarrow d\left((p^{-1})^r\right) = 3^r$$

Basic 12 Round Distinguisher

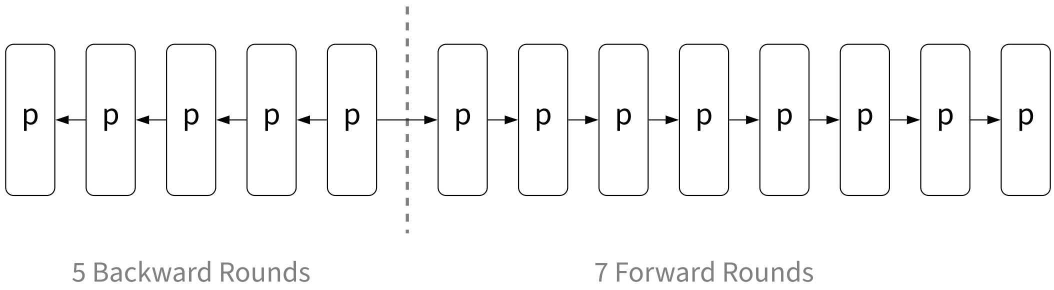


Figure 6: Basic 12 Round Distinguisher

In this we start at an intermediate state after Round 5. Hence there will be 5 backward (inverse) rounds and 7 forward rounds. For the forward 7-rounds, the upper bound degree will be $d\left(p^7\right) = 2^7 = 128$. For the backward 5 rounds of inverse permutation the upper bound will be $d\left((p^{-1})^5\right) = 3^5 = 243$. Hence with $d = \max\{2^7, 3^5\} + 1 = 244$, we start out with $320 - 244 = 76$ of constant bits and 244 variable bits in the intermediate state to create a set of $2^2$44 input intermediate states. For each of these input intermediate states, we calculate 7 rounds forward and 5 rounds backward. The sum of all the resulting input and output states should be equal to 0.

Free Middle Round Distinguisher

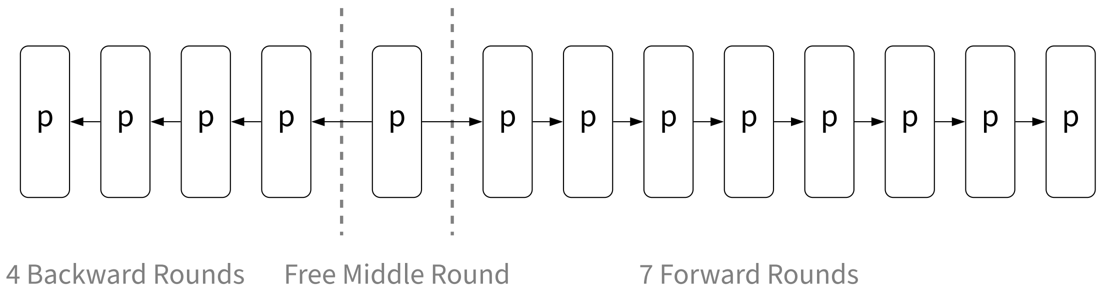


Figure 7: Free Middle Round Distinguisher

In this we start similarly as above at an intermediate state after Round 5. However this time we start out with a degree that is a multiple of 5-bit Sbox and we fill the d variable bits such that Any S-box that is filled should be completely filled. Hence, the remaining constant bits fill out S-boxes completely as well. After application of S-box as well the variable bit and constant bit positions and number will be same. The starting Point of backward trail can be start of S-box of Round 5 whereas the starting point of Forward Rounds can be the Output of S-box of Round 5. The Linear Layer won't have much of an affect hence we get an entire free round without increase in algebraic degree. Hence there will be 4 backward (inverse) rounds and 7 forward rounds. For the forward 7-rounds, the upper bound degree will be $d\left(p^7\right) = 2^7 = 128$. For the backward 4 rounds of inverse permutation the upper bound will be $d\left((p^{-1})^4\right) = 3^4 = 81$. Hence with $d = \max\{2^7, 3^4\} + 1 = 129$, Since we need d to be multiple of 5 our degree can be 130. we start out with $320 - 130 = 190$ of constant bits and 130 variable bits in the intermediate state to create a set of $2^1$30 input intermediate states. For each of these input intermediate states, we calculate 7 rounds forward and 4 rounds backward. The sum of all the resulting input and output states should be equal to 0.

3 Verilog Implementation

3.1 S-Box

The S-box is implemented is 3 ways - using a Look-Up Table (LuT), Algebraic Normal Form (ANF) and Karnaugh Map (K-Map). The ASIC Area Analysis using Genus and the FPGA Area Analysis using Vivado can be found below. It is to be noted that in ASIC K-Maps have lower areas as they are essentially run on AND and OR Gates only which are cheaper compared to XOR. However as the algebraic degree of ASCON is 2, even ANF has lower area compared to LuT.

Table 17: ASIC Area Analysis - Genus

LuT	ANF	K-MAP
43.920	42.480	39.240

Table 18: FPGA Area Analysis - Vivado

Component	Slice LuT (Logic)	OBUF	IBUF	LUT5	LUT4
Number Used	3	5	5	3	2

3.1.1 Schematics

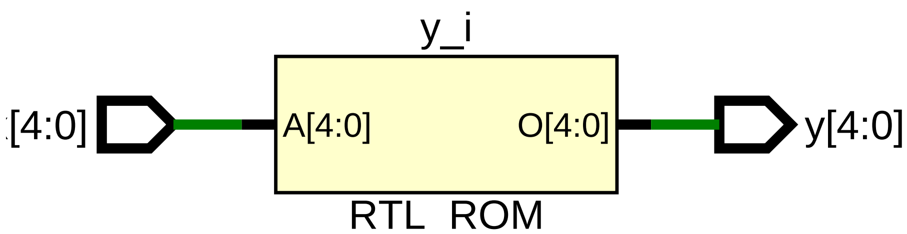


Figure 8: LuT

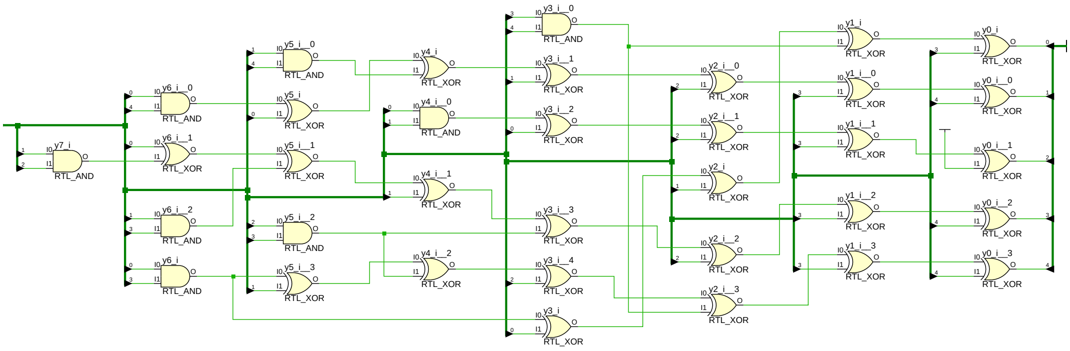


Figure 9: ANF

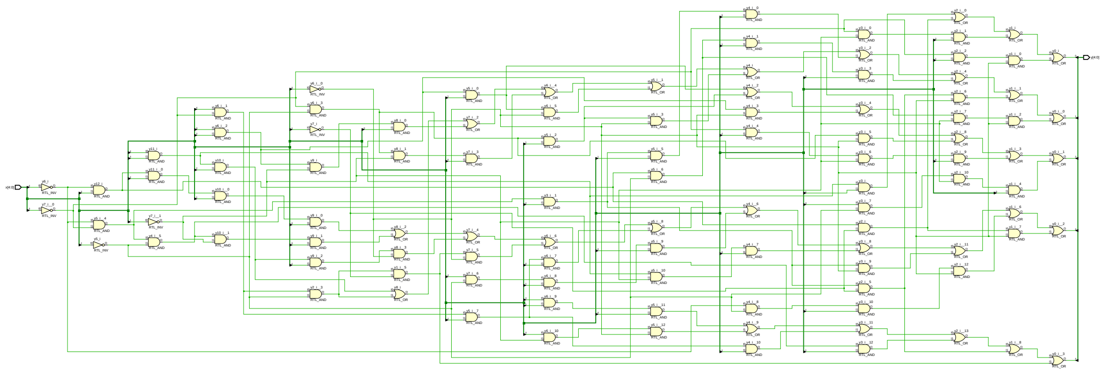


Figure 10: K-Map

3.2 Linear Layer

The Linear Layer takes in 320 Bits Input State and returns back 320 Bit Output State after applying the 5 Linear Functions Σ_i on each word x_i. Its takes in 1958.400 GE on ASIC and here are the primitives used in FPGA

Table 19: Linear Layer FPGA Area Analysis - Vivado

Components	Slice LuT (Logic)	OBUF	LUT3	IBUF
Number Used	169	320	320	320

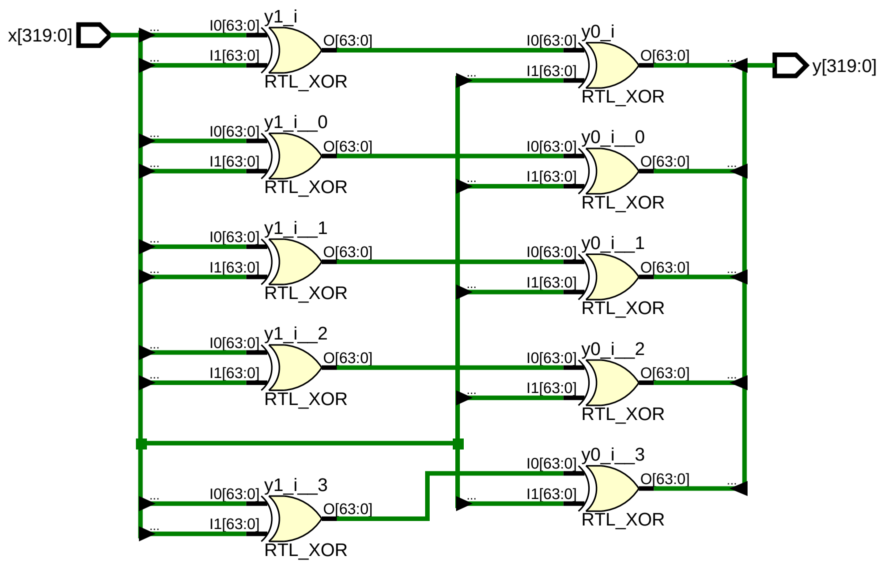


Figure 11: Linear Layer Schematic

3.3 Round Constant

Round Constant Module takes in a clock, start signal and a register containing number of round - could be 12 for end permutation (p^a) or 6, 8 for core permutation (p^b), It returns the 1 Byte of Round Constant Register after every clock cycle depending on the number of rounds as per discussed in the cipher design. After all the rounds are complete, it returns 00 byte for one clock cycle and then repeats round iteration. It takes in 321.840 GE in ASIC and here are the primitives used in FPGA

Table 20: Round Constant FPGA Analysis - Vivado

Components	Slice LuT (Logic)	Slice Register (Flip Flop)	FDRE	FDRE	OBUF	LUT1	LUT6	IBUF	LUT4	LUT2	LUT3	FDSE	LUT5	BUFG (Clock)
Number Used	29	14	12	10	9	7	7	7	6	5	3	2	1	1

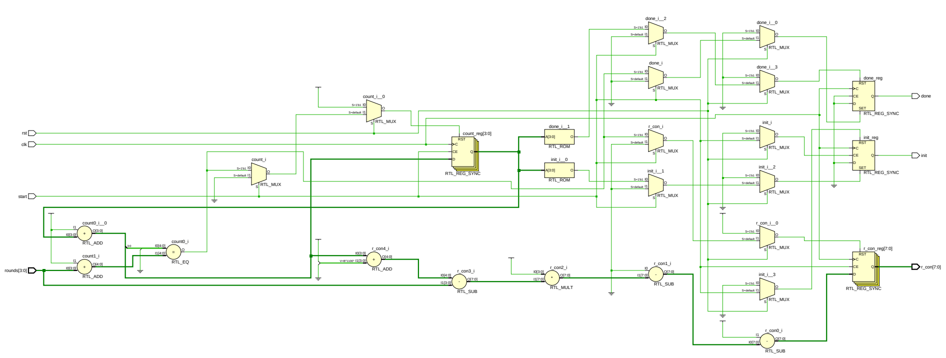


Figure 12: Round Constant Schematic

3.4 Permutation

The Permutation takes in a clock, start signal, number of rounds and 320 Bit Input. It returns 320 Bit Output after the rounds are completed for one clock cycle and then repeats the permutation from start again in a loop. To acheive an iterated design, D-Flip Flops are used to store intermediate round outputs and Multiplexers are used to select start and done signals. As we use 3 Different S-boxes, here are the Area Analysis for the same

Table 21: LuT Sbox - 9148.320 GE

Components	Slice LuT (Logic)	Slice Registers (Flip Flop)	LUT3	FDRE	IBUF	OBUF	LUT5	LUT4	LUT6	LUT	1LUT2	BUFG (Clock)
Number Used	740	334	862	334	326	320	196	110	99	9	5	1

Table 22: KMAP Sbox - 8848.800 GE

Components	Slice LuT (Logic)	Slice Registers (Flip Flop)	LUT3	FDRE	IBUF	OBUF	LUT5	LUT6	LUT4	LUT1	LUT2	BUFG (Clock)
Number Used	777	334	955	334	326	320	196	134	13	9	5	1

Table 23: ANF Sbox - 9056.160 GE

Components	Slice LuT (Logic)	Slice Registers (Flip Flop)	LUT3	FDRE	IBUF	OBUF	LUT6	LUT5	LUT4	LUT1	LUT2	BUFG (Clock)
Number Used	883	334	955	334	326	320	246	140	13	9	5	1

As expected K-Maps do the best in ASIC whereas LuT does better in FPGA as they are better programmed to run LuTs.

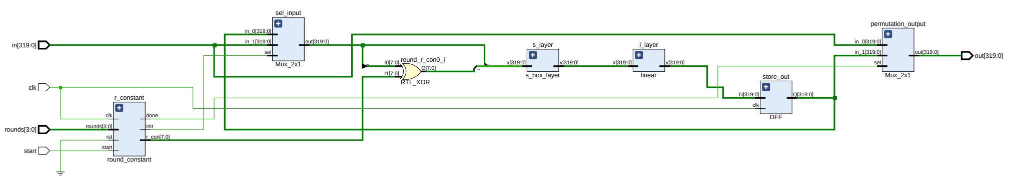


Figure 13: Permutation Schematic

Test Bench for the Permutation function is also implemented for 12, 8 and 6 rounds. For all the 3 round options we take the following input and we get the following outputs. The input is taken from ASCON Python Implementation referenced in the Official Website.

Table 24: Test Vectors

Word	Input	12 Rounds Output	8 Rounds Output	6 Rounds Output
x_0	80400c0600000000	b9117b5b23ee9b3c	7e23b301941a4a62	25c593c800b08021
x_1	c82cbe1c72be1a3a	3b070bfb69319e83	9b907c66bad8b7ed	3cb7732ae5f81096
x_2	85621d92797f8475	e014516471aaf66c	332e8d3ba929ce17	e8edf67b79fb4c3e
x_3	23fd6519897d9e12	cfb97303e71fc531	efc6c3a1c24a7b4e	2d1d39a69214abf8
x_4	5c0609b2f5ca3aaa	a039021db23a484c	45a7f92724354d6a	9e9ce6a15408bc5d

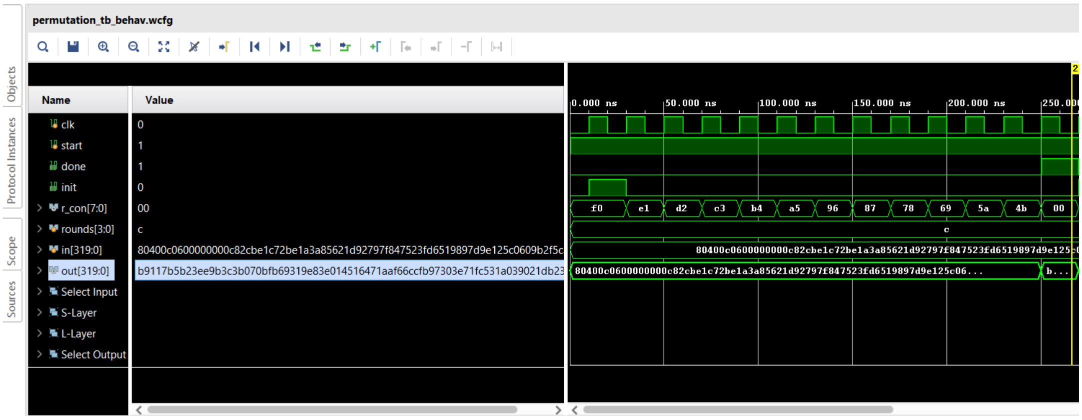


Figure 14: 12 Round Test Bench Wave Configuration

4 Automated Analysis

4.1 Basic MILP Model

A Basic MILP Model is designed which takes in basic properties of S-Box like Branch Number and Linear Layer Model to get the minimum number of active S-boxes for an r round differential. To create that model, we will need to define the following variables.

- $x_{r,w,b} \in \{0, 1\}$ - S-box input Bit b $(0 \leq b \leq 63)$ of word X_w $(0 \leq w \leq 4)$ of round r active or not.
- $y_{r,w,b} \in \{0, 1\}$ - S-box output Bit b $(0 \leq b \leq 63)$ of word X_w $(0 \leq w \leq 4)$ of round r active or not.
- $d_{r,b} \in \{0, 1\}$ - b^{th} $(0 \leq b \leq 63)$ S-box of round r is active
- $u_{r,w,b} \in \{0, 1\}$ - Linear layer model in word x_w $(0 \leq w \leq 4)$ of round r.

Our Objective Function is to Minimize the Number of Active S-Boxes across the Differential Trail

$$\min \sum_{r=1}^{R} \sum_{b=0}^{63} d_{r,b}$$

For Constraints, for starters we should add **non-triviality condition** - that atleast one input bit should be active at start.

$$\sum_{w=0}^{4} \sum_{b=0}^{63} x_{0,w,b} \geq 1$$

Then we can model in the Basic Properties of S-box like Branch Number

$$\sum_{w=0}^{4} (x_{r,w,b} + y_{r,w,b}) \geq 3d_{r,b}$$

If the S-box is Active, at least one input and output bits and at max 5 input and output bits should be active

$$d_{r,b} \leq \sum_{w=0}^{4} x_{r,w,b} \leq 5d_{r,b}$$

$$d_{r,b} \leq \sum_{w=0}^{4} y_{r,w,b} \leq 5d_{r,b}$$

We can also add in that, if there is an active input or output bit, S-Box should be active

$$d_{r,b} \geq x_{r,w,b} \quad 0 \leq w \leq 4$$
$$d_{r,b} \geq y_{r,w,b} \quad 0 \leq w \leq 4$$

The Linear Layer can be modelled in using Mouha et al.'s XOR Framework

$$y_{r,0,b} + y_{r,0,b-19} + y_{r,0,b-28} + x_{r+1,0,b} = 2u_{r,0,b}$$
$$y_{r,1,b} + y_{r,1,b-61} + y_{r,1,b-39} + x_{r+1,1,b} = 2u_{r,1,b}$$
$$y_{r,2,b} + y_{r,2,b-1} + y_{r,2,b-6} + x_{r+1,2,b} = 2u_{r,2,b}$$
$$y_{r,3,b} + y_{r,3,b-10} + y_{r,3,b-17} + x_{r+1,3,b} = 2u_{r,3,b}$$
$$y_{r,4,b} + y_{r,4,b-7} + y_{r,4,b-41} + x_{r+1,4,b} = 2u_{r,4,b}$$

$$\begin{array}{c|c|c|c}
2u_{r,0,b} \geq y_{r,0,b} & 2u_{r,0,b} \geq y_{r,0,b-19} & 2u_{r,0,b} \geq y_{r,0,b-28} & 2u_{r,0,b} \geq x_{r+1,0,b} \\
2u_{r,1,b} \geq y_{r,1,b} & 2u_{r,1,b} \geq y_{r,1,b-61} & 2u_{r,1,b} \geq y_{r,1,b-39} & 2u_{r,1,b} \geq x_{r+1,1,b} \\
2u_{r,2,b} \geq y_{r,2,b} & 2u_{r,2,b} \geq y_{r,2,b-1} & 2u_{r,2,b} \geq y_{r,2,b-6} & 2u_{r,2,b} \geq x_{r+1,2,b} \\
2u_{r,3,b} \geq y_{r,3,b} & 2u_{r,3,b} \geq y_{r,3,b-10} & 2u_{r,3,b} \geq y_{r,3,b-17} & 2u_{r,3,b} \geq x_{r+1,3,b} \\
2u_{r,4,b} \geq y_{r,4,b} & 2u_{r,4,b} \geq y_{r,4,b-7} & 2u_{r,4,b} \geq y_{r,4,b-41} & 2u_{r,4,b} \geq x_{r+1,4,b}
\end{array}$$

The results after modelling it in the automation tool gurobi and running for 3 rounds are the following

Table 25: Basic MILP Model

Rounds	Active S-Boxes	Variables (Real)	Variables (Binary)	Inequalities	Time Taken
1	1	960	384	2561	0.03 s
2	4	1920	448	5121	0.51 s
3	12	2880	512	7681	2 min 52.47 s

4.2 Logical Conditonal Modelling

In the above basic MILP Model we have not modeled the intricacies of the S-box. To add in few elements of ASCON S-box, we can use the Undisturbed Bits to create Logical Conditional Modelling the following way.

$$x_{r,0,b}, x_{r,1,b}, x_{r,2,b}, x_{r,3,b}, x_{r,4,b}) = (\delta_0, \delta_1, \delta_2, \delta_3, \delta_4) \Rightarrow y_{r,w,b} = \delta$$

$$\sum_{w'=0}^{4} (-1)^{\delta_i} x_{r,w',b} + (-1)^{\delta+1} y_{r,w,b} - \delta + \sum_{w'=0}^{4} \delta_i \geq 0$$

There are 13 such Undisturbed bits such that Δ_o^r has one bit invariant.

Table 26: Undisturbed Bits

Δ_i	Δ_o^*
00001	*1***
00011	***0*
00101	1****
11111	*0***

Δ_i	Δ_o^*
10101	****1
10110	1****
11110	*1***
01110	*0***
01011	***1*

Δ_i	Δ_o^*
10111	****0
11000	**1**
11100	**0**
00110	****1

After Modelling these inequalities for every S-box in every round and implementing it on the automation tool for 3 rounds, we get the following results

Table 27: Logical Conditional Modelling

Rounds	Active S-Boxes	Variables (Real)	Variables (Binary)	Inequalities	Time Taken
1	1	965	384	3393	0.02 s
2	4	1925	448	6785	1.32 s
3	12	2885	512	10177	3 min 28.54 s

4.3 Convex Hull

Convex Hull can exhaustively represent every element of ASCON S-box, however it would also lead to huge overhead as per S-box 2415 inequalities will be added in whereas in the above just 13 per S-box. Hence while we found the inequalities as shown below, we have not integrated it into the automation tool.

```
ascon_hull = list(Polyhedron(vertices=generate_vertices(Ascon)).inequality_generator())

ascon_hull
```

```
[An inequality (-1, 0, 0, 0, 0, 0, 0, 0, 0, 0) x + 1 >= 0,
 An inequality (0, -1, 0, 0, 0, 0, 0, 0, 0, 0) x + 1 >= 0,
 An inequality (0, 0, -1, 0, 0, 0, 0, 0, 0, 0) x + 1 >= 0,
 An inequality (0, 0, 0, -1, 0, 0, 0, 0, 0, 0) x + 1 >= 0,
 An inequality (-1, 0, 0, -1, -1, 0, 0, -1, -1, 0) x + 4 >= 0,
 An inequality (0, -1, 0, -1, -1, 0, 1, 1, 1, 0) x + 2 >= 0,
 An inequality (0, 0, 0, 0, -1, 0, 0, 0, 0, 0) x + 1 >= 0,
 An inequality (1, -1, -1, -1, -1, 0, 1, 0, 0, 0) x + 3 >= 0,
 An inequality (0, 0, 0, 0, 0, 0, 1, 0, 0, 0) x + 0 >= 0,
 An inequality (-1, 0, 0, -1, -1, 0, 0, 1, 1, 0) x + 2 >= 0,
 An inequality (0, 0, -1, -1, -1, 0, 1, 1, 1, 0) x + 2 >= 0,
 An inequality (-3, -2, -3, -4, -6, -1, -3, -3, -3, -1) x + 23 >= 0,
 An inequality (-1, -1, -1, -1, -1, 0, -1, 0, 0, 0) x + 5 >= 0,
 An inequality (-1, -1, -1, -2, -2, 0, -1, -1, -1, 0) x + 8 >= 0,
 An inequality (-1, -1, -1, -1, 1, 0, 1, 0, 0, 0) x + 3 >= 0,
 An inequality (0, -1, -1, -1, -1, 0, 0, -1, -1, 0) x + 5 >= 0,
 An inequality (-2, -2, -2, -1, 1, 0, 2, -1, -1, 0) x + 7 >= 0,
 An inequality (-1, -1, -1, -2, -1, 0, 1, -2, -2, 0) x + 8 >= 0,
 An inequality (-1, -1, -1, -2, -1, 0, 1, 2, 2, 0) x + 4 >= 0,
 An inequality (-1, -1, -1, -3, -2, 0, 2, 3, 3, 0) x + 5 >= 0,
 An inequality (-1, -1, -1, -1, 0, 0, 1, 1, 1, 0) x + 3 >= 0,
 An inequality (1, -1, -1, -2, -2, 0, 2, 1, 1, 0) x + 4 >= 0,
 An inequality (0, 0, 0, -1, -1, -1, 1, 1, 1, -1) x + 3 >= 0,
 An inequality (0, 0, 0, 0, 0, 0, 0, 0, 0, -1) x + 1 >= 0,
 An inequality (0, 0, -1, -1, -1, -1, 0, -1, -1, 0) x + 5 >= 0,
```

Figure 15: Convex-Hull - 2415 Inequalities

5 Conclusion

In conclusion we have covered all four aspects of the cipher - Construction, Cryptanalysis, Hardware Implementation, Automated Analysis extensively. We have covered the design of the Authenticated Encryption as well as the permutation components. We have analyses the cipher through various cryptanalysis techniques like Differential, Linear, Zero-Sum. We have then implemented the permutation in verilog and have done the Area Analysis in both ASIC and FPGA. In the end, we have implemented Basic as well as Logical Condtional Modelling in MILP to find out the minimum number of active S-boxes in a Differential.

6 References

- https://ascon.iaik.tugraz.at
- [**DEMS15**] *Cryptanalysis of Ascon* - Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. **IACR** - 2015/030 (pp. 28, 31–33, 35).
- [**Tez16**] *Truncated, Impossible, and Improbable Differential Analysis of Ascon* - Cihangir Tezcan. **IACR** - 2016/490 (pp. 28, 33).