1 Introduction

1.1 Referenced documents

1. A. Barrera, C. W. Cheng and S. Kumar, ”A Fast Implementation of the Rijndael Substitution Box for Cryptographic AES,” 2020 3rd International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA, 2020, pp. 20-25, doi: 10.1109/ICDIS50059.2020.00009.

2. National Institute of Standards and Technology (2001) Advanced Encryption Standard (AES). (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 197-upd1, updated May 9, 2023. https://doi.org/10.6028/NIST.FIPS.197-upd1.

3. G. Peng and S. Zhu, "FPGA Implementation of AES Encryption Optimization Algorithm," 2021 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), Xi'an, China, 2021, pp. 650-653, doi: 10.1109/ICITBS53129.2021.00165.

1.2 Design library name

uk65lscllmvbbr_100c25_tc_ccs

1.3 People involved in the block

1. Rahul (2023EEN2238)

2 Function

2.1 Brief overview

AES (Advanced Encryption Standard) encryption involves several steps that are performed in multiple rounds. The number of rounds depends on the key size: 10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256. Here are the steps in AES encryption for AES-128:

- **Key Expansion:**

The original 128-bit encryption key is expanded into a set of round keys, one for each round. This key expansion involves a series of transformations that create a key schedule.

- **Initial Round:**

The plaintext (input data) is XORed with the initial round key.

- **Rounds (9 Rounds for AES-128):**

Each round consists of several transformations:

SubBytes: Substitutes each byte of the block with a corresponding byte from an S-box (a predefined substitution table).
ShiftRows: Shifts the rows of the block by different offsets.
MixColumns: Mixes the columns of the block to introduce diffusion.
AddRoundKey: XORs the block with the round key for that specific round.

- **Final Round:**

The final round is slightly different from the previous rounds:

1. SubBytes
2. ShiftRows
3. AddRoundKey

- **Output:**

The processed block of data is now the ciphertext.

The number of rounds and the specific operations applied in each round are designed to provide security and resistance to various cryptographic attacks. AES-128 uses 10 rounds, each with these operations, to encrypt data. The process is then reversed during decryption, using the same key schedule but in reverse order. This ensures that the original plaintext can be accurately reconstructed from the ciphertext using the correct key.

2.2 Interfaces

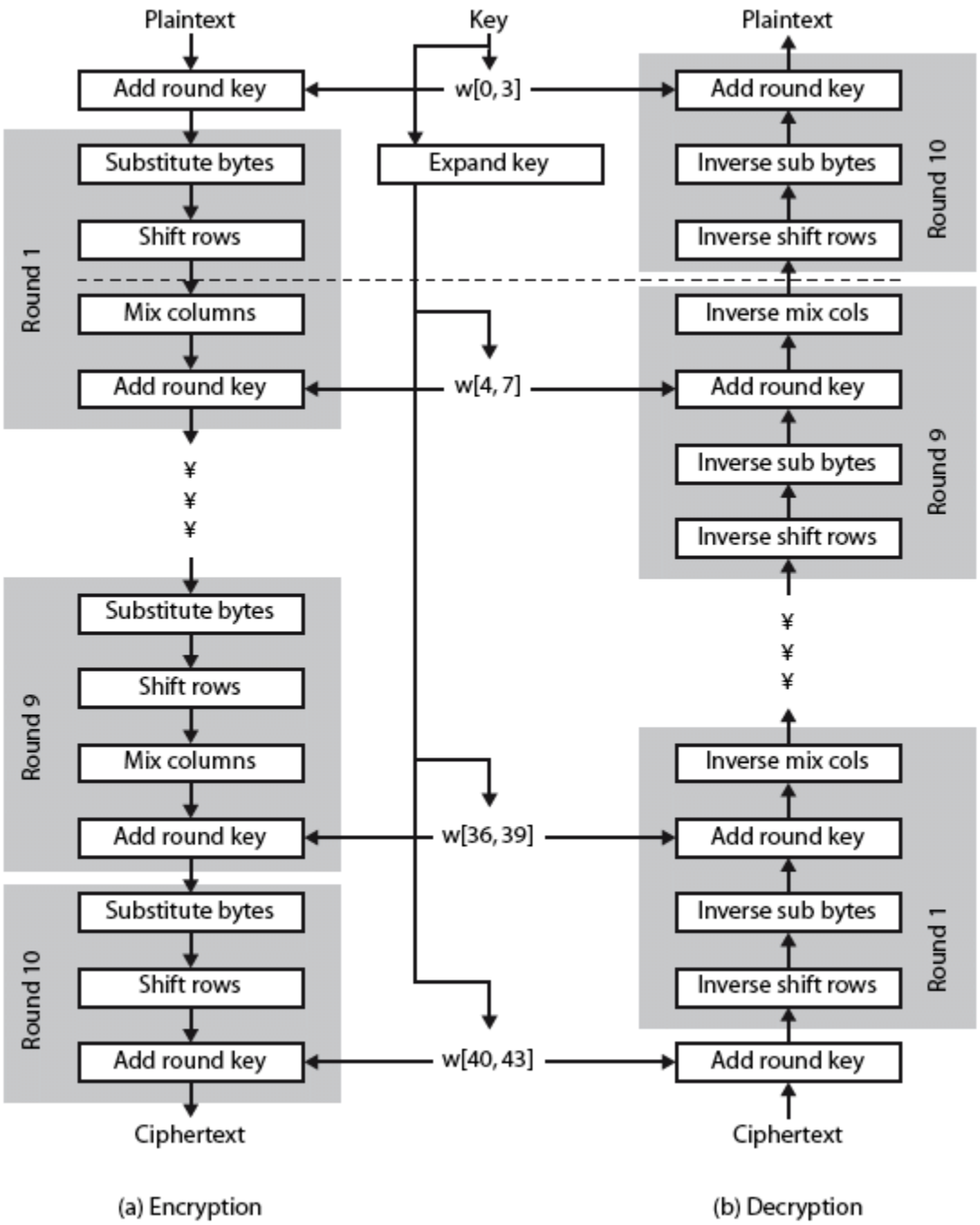
Table 1: Description of I/Os used in the AES-128 Encryption.

Signal Name	Input/Output (I/O)	Description
plainText <127:0>	Input	128 bit plain text we want to encrypt
privateKey <127:0>	Input	128 bit key to encrypt or decrypt
cipherText <127:0>	Output	128 bit encrypted cipher text
V_{DD}	Input/Output	DC supply voltage
V_{SS}	Input/Output	Ground

Table 2: Description of I/Os used in the AES-128 decryption.

Signal Name	Input/Output (I/O)	Description
cipherText <127:0>	Input	128 bit encrypted cipher text we want to decrypt
privateKey <127:0>	Input	128 bit key to encrypt or decrypt
plaintText <127:0>	Output	128 bit plain text after decryption
V_{DD}	Input/Output	DC supply voltage
V_{SS}	Input/Output	Ground

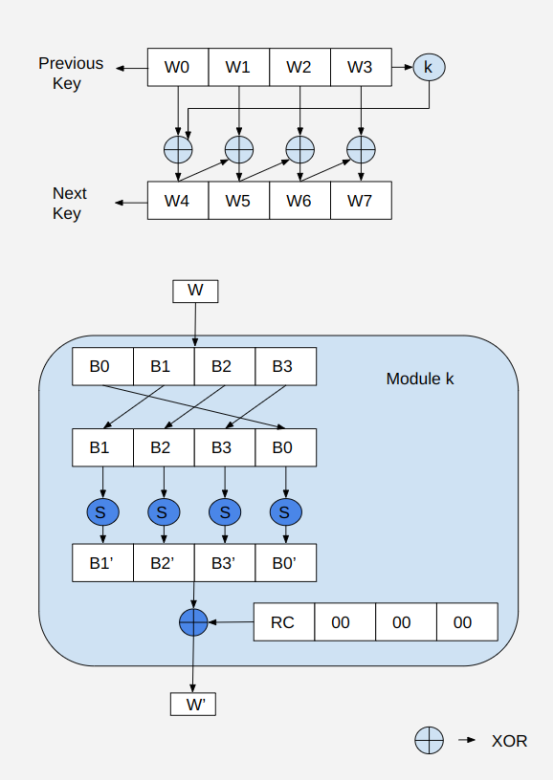
2.3 Architecture



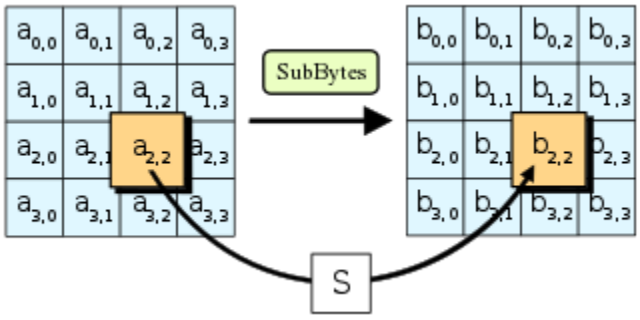
2.4 Detailed functional description

- **Key Expansion:**

The original 128-bit encryption key is expanded into a set of round keys, one for each round. This key expansion involves a series of transformations that create a key schedule as shown below to generate all 10 round keys.

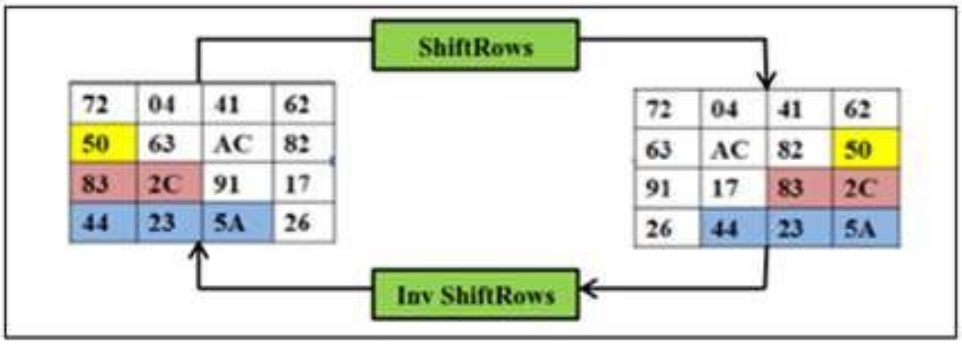


- **Substitute Bytes**
  Substitutes each byte of the block with a corresponding byte from an S-box (a predefined substitution table).



- **Shift rows / Inverse Shift Rows**

Shifts the rows of the block by different offsets depending upon encryption or decryption.



- **Mix Columns / Inverse Mix Columns**

Mixes the columns of the block to introduce diffusion.

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \times \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix}$$

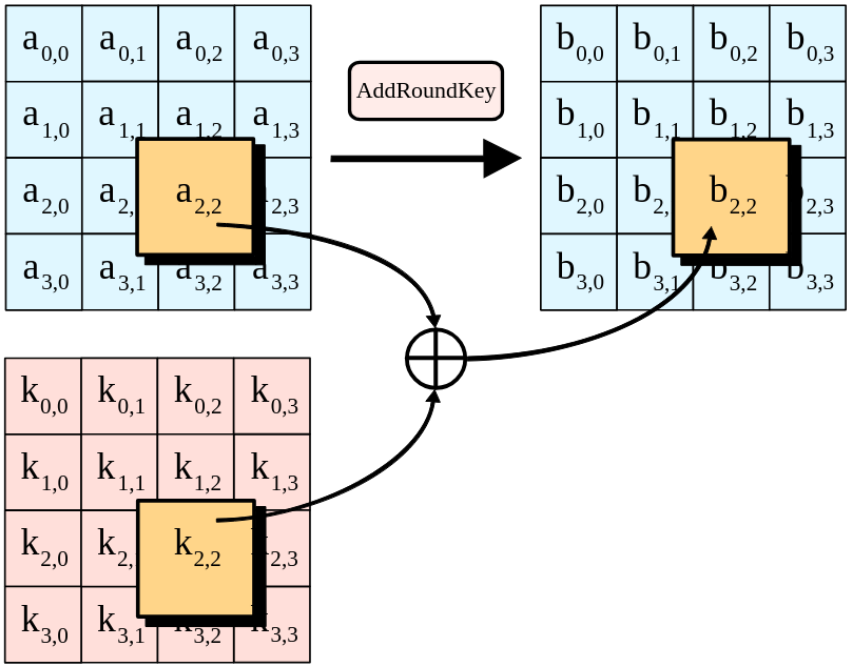
(a) Mix Columns for Encryption

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

(b) Inverse Mix Columns for Decryption

- **Add Round Key**

The state matrix is XOR'ed with the round key for that specific round.



3 Design parameters

3.1 Performance Requirements

- Operating frequency required: 150MHz

3.2 Clock Distribution

Frequency: 150MHz
Duty cycle: 50%
Rise time: 0.13ns
Fall time: 0.13ns
Clock uncertainty: 0.01ns
Active Clock Edge: Positive edge

3.3 Reset

Polarity: Active Low

At reset, the registers are assigned the following values:

```
out <= 0;
plainText <= 0;
rk0_reg <= 0;
rk1_reg <= 0;
rk2_reg <= 0;
rk3_reg <= 0;
rk4_reg <= 0;
rk5_reg <= 0;
rk6_reg <= 0;
rk7_reg <= 0;
rk8_reg <= 0;
rk9_reg <= 0;
rk10_reg <= 0;
r0_state_matrix_reg <= 0;
r1_state_matrix_reg <= 0;
r2_state_matrix_reg <= 0;
r3_state_matrix_reg <= 0;
r4_state_matrix_reg <= 0;
r5_state_matrix_reg <= 0;
r6_state_matrix_reg <= 0;
r7_state_matrix_reg <= 0;
r8_state_matrix_reg <= 0;
r9_state_matrix_reg <= 0;
round_count <= 0;
```

3.4 Timing Description

- Latency: 140ns for Encryption (21 Clock Cycles)
  273.33ns for Decryption (41 Clock Cycles)

- Update conditions: At every positive edge of clock.

4 Verification Strategy

4.1 Objectives

The encryption will take 128 bit key and 128 bit message as input and the encrypt the message with AES ECB mode. The decryption module will take 128 bit key and 128 bit cipher as input and decrypt the cipher to give plain text with AES ECB mode.

4.2 Tools and Version

- Syntheisis - Genus
- Place and Route - Innovus
- Simulation - ModelSim
- Verification support tools - python3, pycryptodome

4.3 Checking mechanisms

A python script has been used to verify different test cases using a library called pycryptodome. Initially, simulation is done to get the encrypted and decrypted outputs and then they are compared to the python scripts output manually.

```
Key in plain text:  b'Thats my Kung Fu'
Key in Hex:  b'5468617473206d79204b756e67204675'

Message in plain Text:  b'Two One Nine Two'
Message in Hex:  b'54776f204f6e65204e696e652054776f'

------------------------------------------------

Encrypted Message:  b'29c3505f571420f6402299b31a02d73a'
Decrypted Messgae:  b'54776f204f6e65204e696e652054776f'
```

(a) Python script output

```
# Encrypted messgae: 29c3505f571420f6402299b31a02d73a

VSIM 15>
```

(b) ModelSim Output for Encryption

```
# Decrypted messgae: 54776f204f6e65204e696e652054776f

VSIM 19>
```

(c) ModelSim Output for Decryption

5 Functional Checklist

The following things needs to be checked:

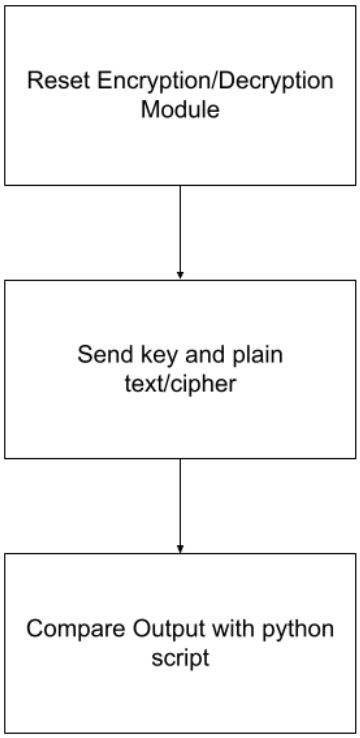
1. Operating Frequency: 150MHz
2. Encrypt message in ECB mode.
3. Decrypt cipher in ECB mode.

5 Testbench

5.1 Overview

The testbench initially resets the encryption and decryption module to initialise the registers to 0. Then the key and plain text (for encryption) or key and cipher (for decryption) is given as input to get the final output. To verify the functionality, the output is compared to the output of a python script which using pycryptodome library for encryption and decryption.

5.2 Architecture



(Testbench Architecture)

6 Tests Specification

For Decryption:

Simulation time: 520ps

Clock Frequency: 150Mhz

Key: 5468617473206d79204b756e67204675

Cipher 1: 29c3505f571420f6402299b31a02d73a
Cipher 2: 8b9412a0aabd718f7ac5bf7fe59dbdaf
Cipher 3: 26ab74baeef2b149edbda2a281e0e7b6

For Encryption:

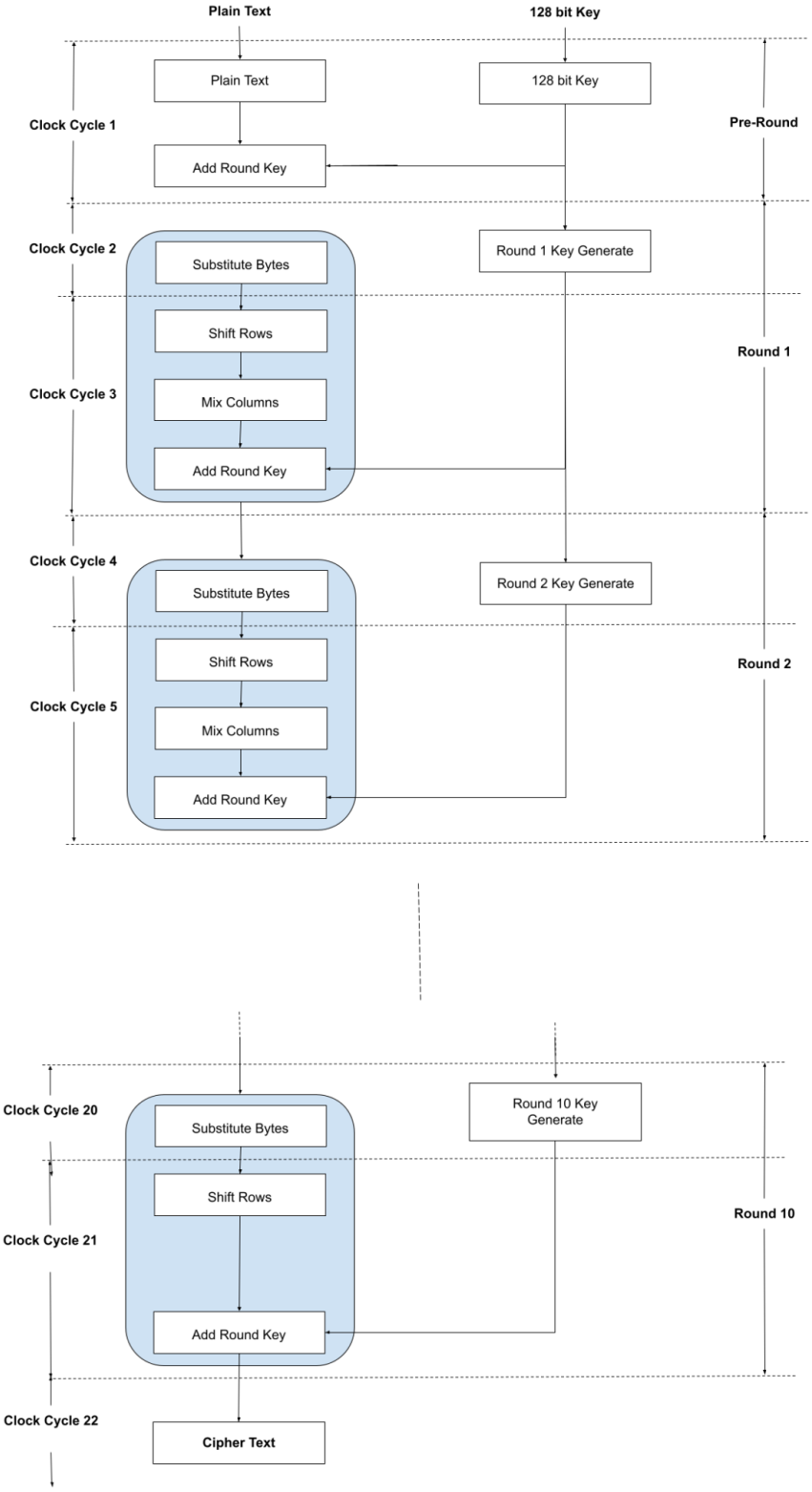
Simulation time: 310ps

Clock Frequency: 150Mhz

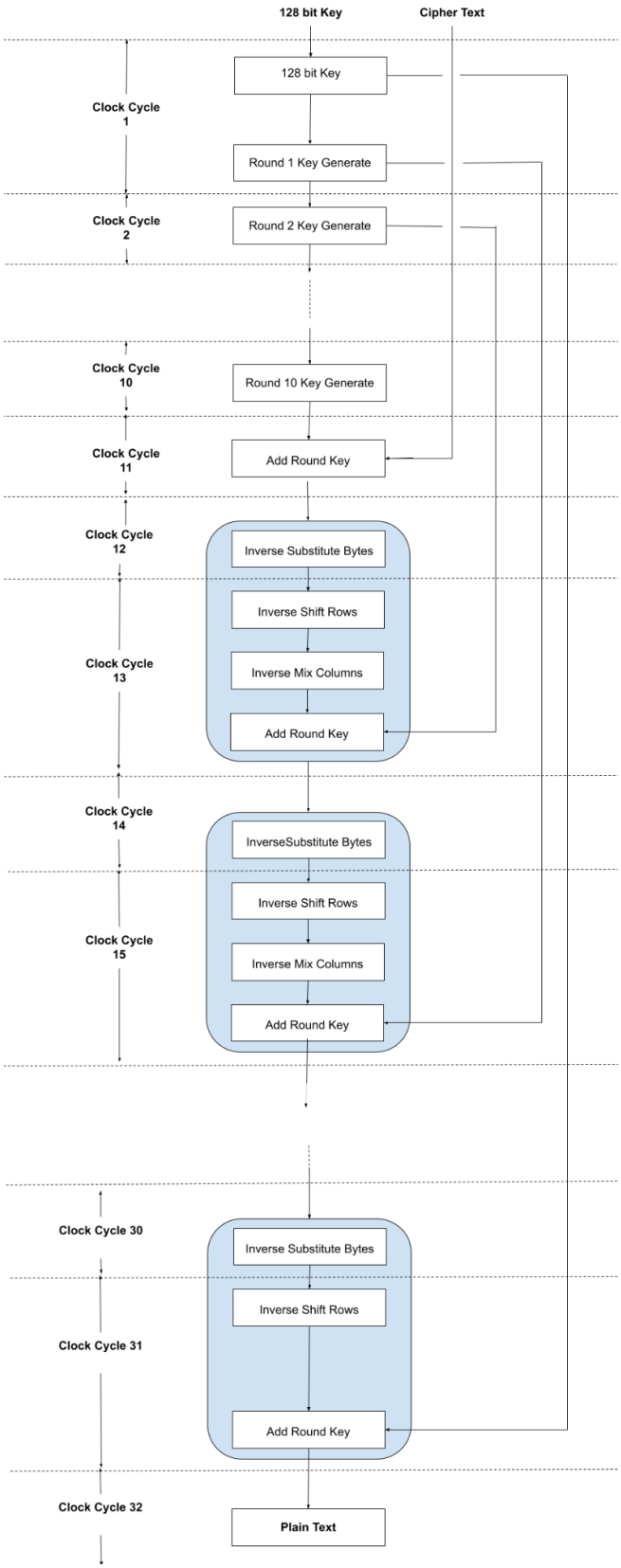
Key: 5468617473206d79204b756e67204675

Plain Text 1: 54776f204f6e65204e696e652054776f
Plain Text 2: 54776f204f6e65204e696e652054666f
Plain Text 3: 54776f204f6e56204e696e652054666f

7 Design Microarchitecture



(a) Encryption Microarchitecture



(b) Decryption Microarchitecture

7.1 Top Level Interface

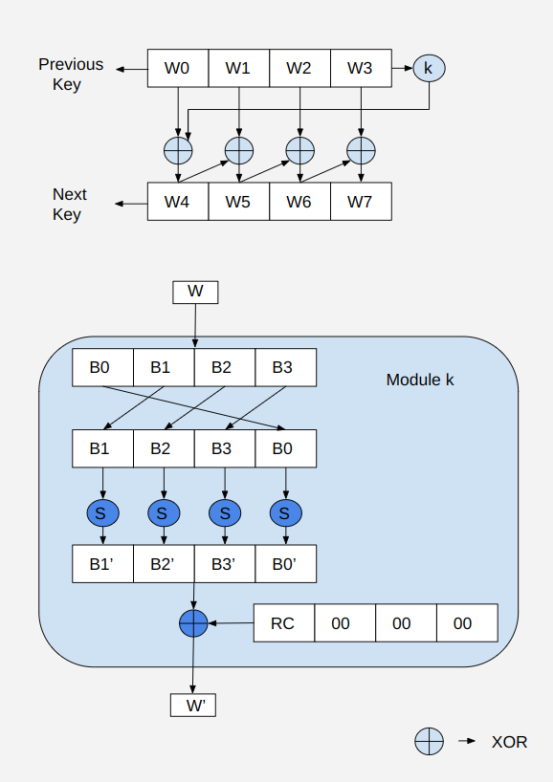
NA

7.2 Sub-Block Description

- Sub-blocks used in the overall design:
  1. Key Expansion
  2. Add Round Key
  3. Substitute/Inverse substitute Bytes
  4. Shift/Inverse Shift Rows
  5. Mix Coulumns/Inverse Mix Columns
  6. Lookup Table

1) Key Expansion

The original 128-bit encryption key is expanded into a set of round keys, one for each round. This key expansion involves a series of transformations that create a key schedule as shown below to generate all 10 round keys.



2) Add Round Key

This block performs addition of round key to the state matrix in each round using bit by bit XOR. Each round key is generated by key expansion block.

3) Substitute/Inverse substitute Bytes

Substitution is implemented using lookup tables. The byte that needs to be substituted is given as input and in output the byte that the input needs to be replaced with is received.

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	a	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	b	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	c	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	d	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	e	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	f	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

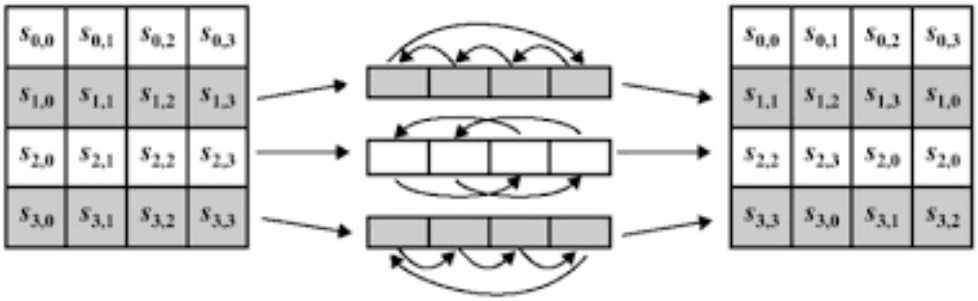
(a) Substitute bytes lookup table

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	83	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	a	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	b	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	c	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	d	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	e	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	f	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(b) Inverse substitute bytes lookup table

4) Shift/Inverse Shift Rows

This block just shifts the rows in the state matrix array. This block does not involve any combinational circuit, it just connects the input to the output directly through wires .



5) Mix Coulumns/Inverse Mix Columns

For mix columns, simple abstract mathematics is used. Since we only need to multiply with 2 and 3, we can just use shift and xor operation to mix the columns.

- To multiply with 2, the input is left shifted once.
- To multiply with 3, the input is left shifted once and 1 is added to LSB.

For inverse mix columns, following operation are done:

- Multiply by {0e} is done by :
  1) (multiplying by {02} 3 times which is equivalent to multiplication by {08}) xor
  2) (multiplying by {02} 2 times which is equivalent to multiplication by {04}) xor
  3) (multiplying by {02}) so that 8+4+2= e. where xor is the addition of elements in finite fields

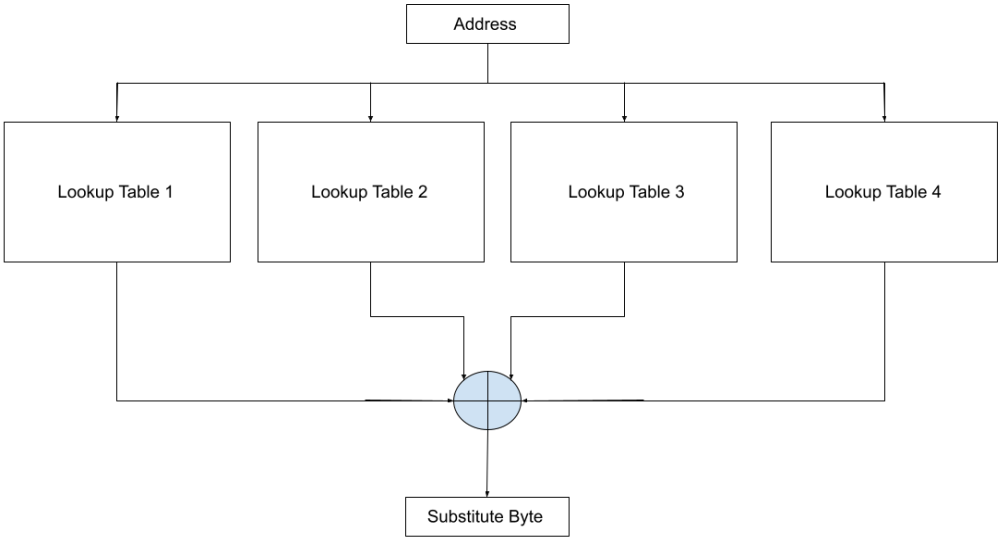
- Multiply by {0d} is done by :
  1) (multiplying by {02} 3 times which is equivalent to multiplication by {08}) xor
  2) (multiplying by {02} 2 times which is equivalent to multiplication by {04}) xor
  3) (the original x) so that 8+4+1= d. where xor is the addition of elements in finite fields.

- Multiply by {0b} is done by :
    1) (multiplying by {02} 3 times which is equivalent to multiplication by {08}) xor
    2) (multiplying by {02}) xor (the original x) so that 8+2+1= b. where xor is the addition of elements in finite fields

- Multiply by {09} is done by :
    1) (multiplying by {02} 3 times which is equivalent to multiplication by {08}) xor
    2) (the original x) so that 8+1= 9. where xor is the addition of elements in finite fields

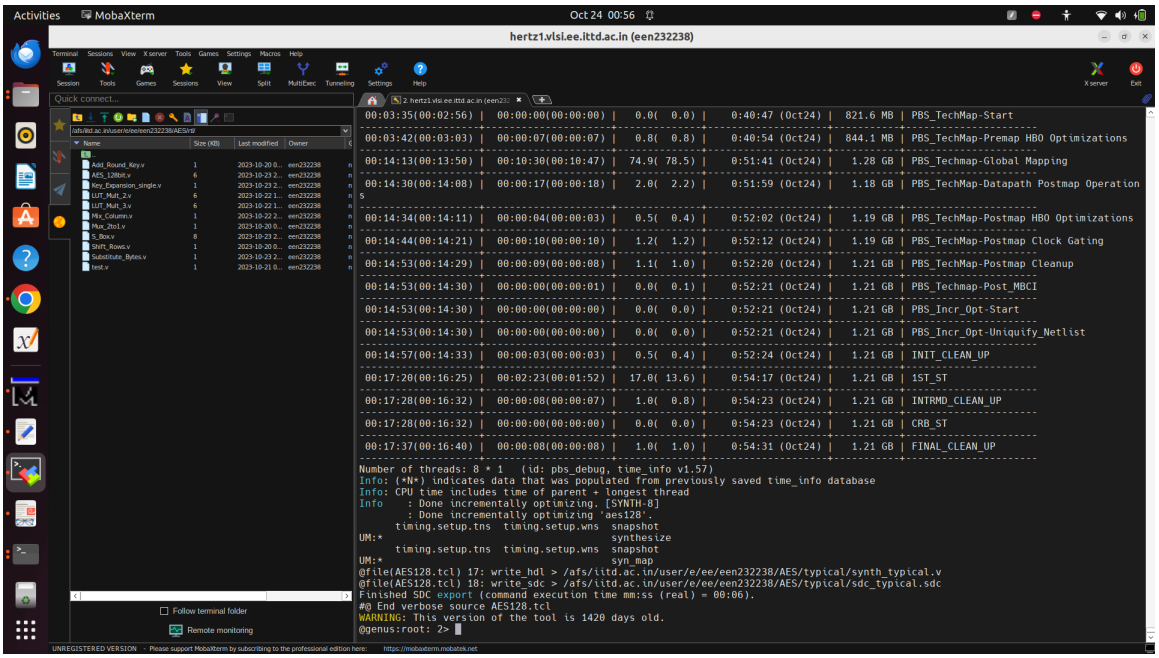
6) Lookup Table

Lookup tables has been implemented using case statements in verilog. Also to reduce delay, a lookup table has been divided into four lookup tables which are given the input parallely. The output of each lookup table is then XORed bit by bit to get the final output.



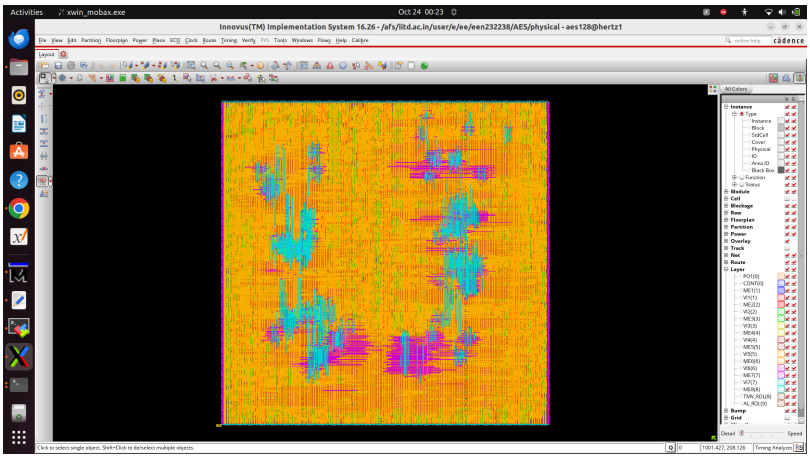
7.3 Structural Mapping Process

Genus was used to synthesise the RTL code, and encounter to do the place and route and make layout.



8 Physical hierarchy

8.1 Floorplanning



Area: 395589 um^2
Aspect Ratio: 1:1
Shape: Square

8.2 Clocktree insertion

Following clock constraints were provided to the synthesis tool:

1. create_clock -name "clk" -add -period 6.66 -waveform {0 3.33} [get_ports "clk"]
2. set_clock_transition -rise 0.13 [get_clocks "clk"]
3. set_clock_transition -fall 0.13 [get_clocks "clk"]
4. set_clock_uncertainty 0.01 [get_clocks "clk"]

8.3 Layout Strategy

- Core to Die Boundary from all sides: 4.5um
- Power planning:
    1. Ring
       Width: 0.7
       Spacing: 0.985
    2. Stripe:
       Width: 0.3
       Spacing: 0.4

- Routing: Timing driven and SI driven

9 Results

9.1 Area

Block Name	Standard Cells Count	Dimensions (W*L)	Layout Area(um2)
Encryption	53039	628.96*628.96 um	395589
Decryption	52839	613.8*613.8	376745

10.2 Timing

All timing constraints have been met and following worst negative slack has been observed:

WNS	Setup	Hold
Pre-CTS	+2.165	+0.146
Post-CTS	+2.178	+0.141

(a) Encryption

WNS	Setup	Hold
Pre-CTS	+2.132	+0.113
Post-CTS	+2.118	-0.042

(b) Decryption

10.3 Testability analysis

Test signals

1) Encryption
   - clk
   - rst
   - plainText
   - cipherText
   - privateKey

2) Decryption
   - clk
   - rst
   - plainText
   - cipherText
   - privateKey

10.4 DRC rule violations

None.

11 Bugs known at submission date

None.