Lab4 : Finite State Machine
– AES 128 Encryption

Advisor : Lih-Yih Chiou

Lecturer : David

Date : 2024/03/14

EE

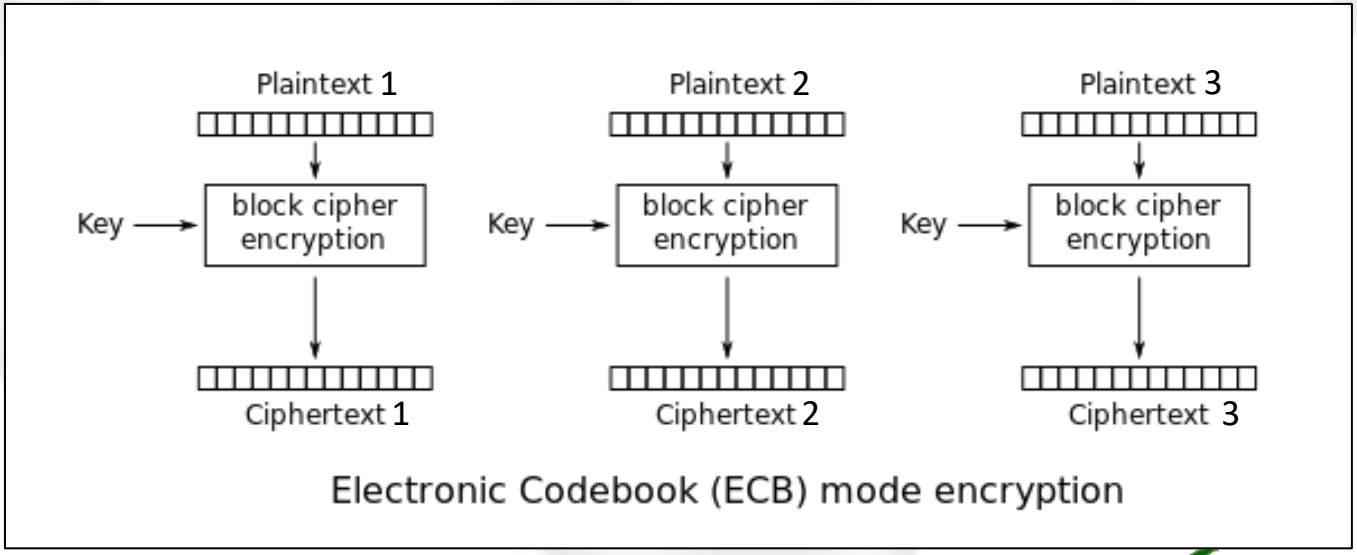
LPHPLAB
VLSI Design LAB

Outline

- Introduction
- Design Specifications
- System Description
- Criteria

Introduction (1/2)

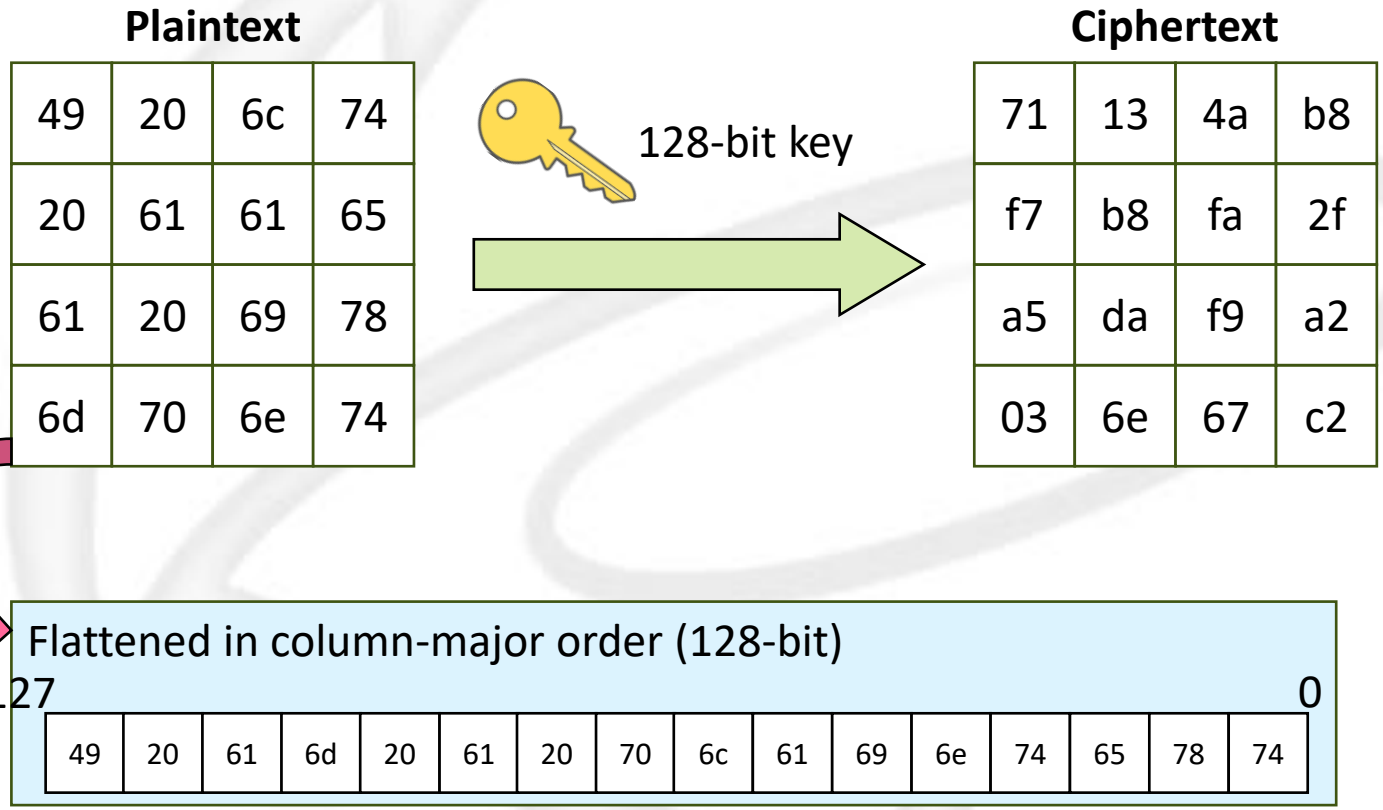
- The Advanced Encryption Standard (AES) is a symmetric-key and block cipher algorithm widely used for securing sensitive data.
- We are implementing is the simplest mode Electronic Code Book (ECB), which is one of several modes of operation for a block cipher. Each sub-block is encrypted independently in this process.



Electronic Codebook (ECB) mode encryption

Introduction (2/2)

- AES, a variant of Rijndael, features a consistent block size of 128 bits and supports key sizes of 128, 192, 256 bits. Most AES computations are done in a specific finite field.

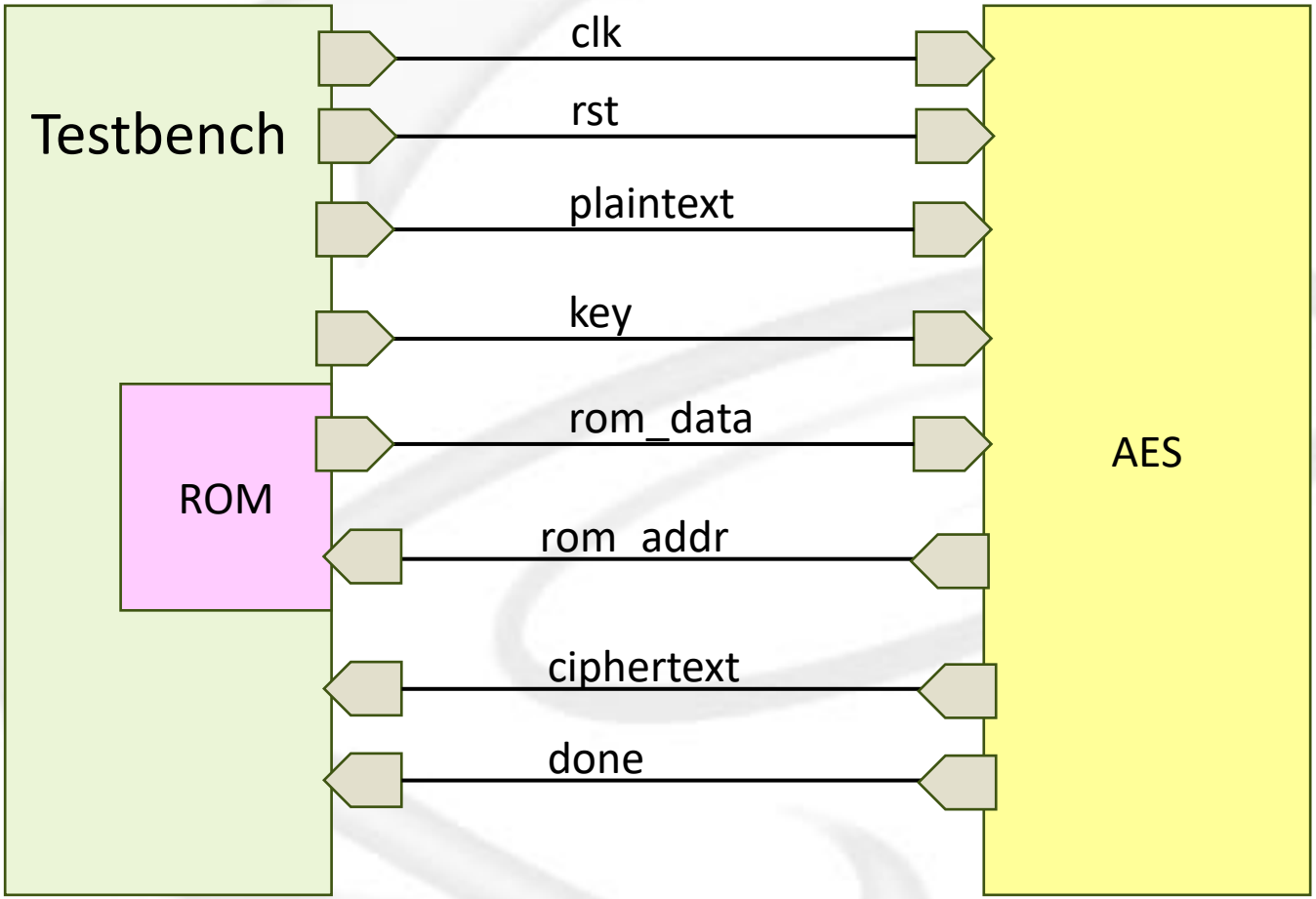


Outline

- Introduction
- Design Specifications
- System Description
- Criteria

Design Specifications (1/2)

Block Diagram



Design Specifications (2/2)

I/O information

Signal	I/O	width	Description
clk	I	1	Clock signal (positive edge trigger)
rst	I	1	Synchronous reset signal (active high)
plaintext	I	128	The original, unencrypted message
key	I	128	Secret key, 128-bit means have 10 rounds encryption
rom_data	I	8	The Rijndael S-box data
rom_address	O	8	The Rijndael S-box address
ciphertext	O	128	The encrypted version of a plaintext message
done	O	1	Complete Signal, testbench will receive the ciphertext

Outline

- Introduction
- Design Specifications
- System Description
- Criteria

System Description (1/14)

- In pattern 1, there is a 16-character English sentence represented in HEX ASCII code, along with a 128-bit encryption key. The plaintext is encrypted using the AES algorithm, resulting in the final ciphertext.

Plaintext – I am a plaintext

I		a	m		a		p	l	a	i	n	t	e	x	t
49	20	61	6d	20	61	20	70	6c	61	69	6e	74	65	78	74

Encryption Key – ThisIsASecretkey

T	h	i	s	I	s	A	S	e	c	r	e	t	k	e	y
54	68	69	73	49	73	41	53	65	63	72	65	74	4b	65	79

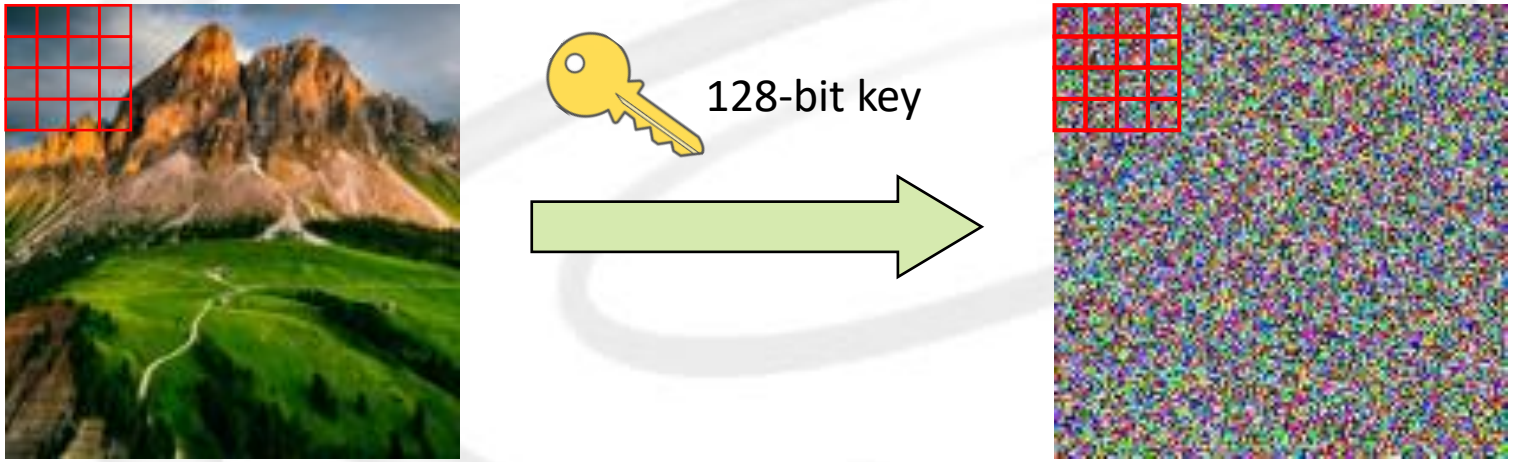
AES encryption

Ciphertext – not a sentence

q	÷	¥			¸	Ú	n	J	ú	ù	g	.	/	ç	Â
71	f7	a5	03	13	b8	da	6e	4a	fa	f9	67	b8	2f	a2	c2

System Description (2/14)

- In pattern 2, 3, the testbench will partition each image into 4*4 block matrices and sequentially send them into your design. At the end of simulation, an encrypted image will be generated.

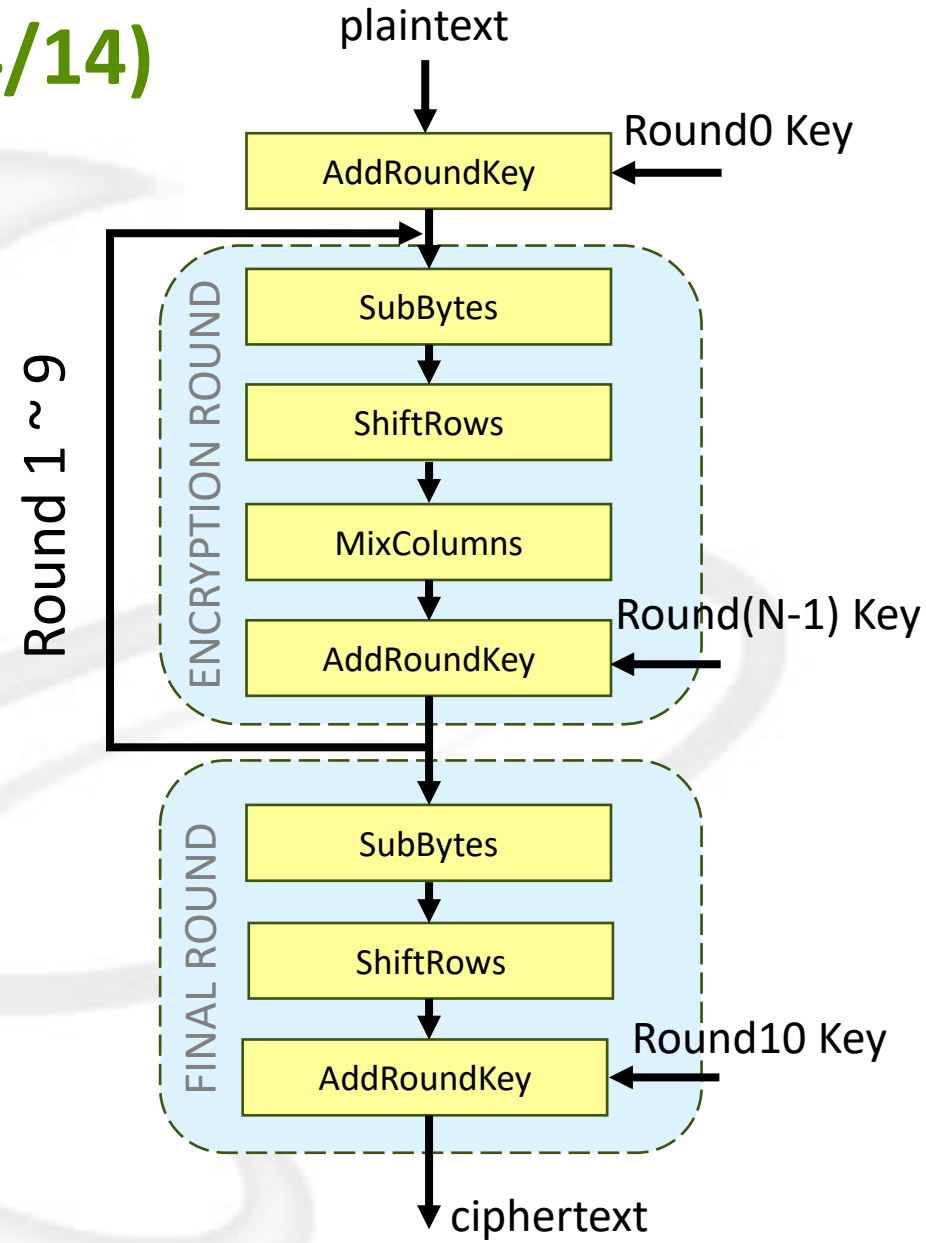


System Description (3/14)

- AES processing step
  - AddRoundKey : the state is combined with round key using bitwise XOR.
  - SubBytes : the non-linear substitution step where each byte is replaced with another by lookup table.
  - ShiftRows : the last three rows of the state are shifted cyclically.
  - MixColumns : a linear transformation which operates on the columns of the state.
  - KeyExpansion: round keys are derived from the cipher key using the AES key schedule.

System Description (4/14)

- AES flow diagram:
  - A total of 10 rounds are performed, with no MixColumns transformation required in the final round.
  - The Round 0 key is a secret key sent through the I/O port "key", while the other round keys are computed through the key expansion operation.
  - Design your own finite stat machine.



System Description (5/14)

- AddRoundKey: bitwise XOR operation

49	20	6c	74
20	61	61	65
61	20	69	78
6d	70	6e	74

Plaintext

54	49	65	74
68	73	63	4b
69	41	72	65
73	53	65	79

Round0 Key

1d	69	09	00
48	12	02	2e
08	61	1b	1d
1e	23	0b	0d

State

System Description (6/14)

- Rijndael 8-bit substitution box: Lookup table (ROM)

1d	69	09	00
48	12	02	2e
08	61	1b	1d
1e	23	0b	0d

Old state

a4	f9	01	63
52	c9	77	31
30	ef	af	a4
72	26	2b	d7

New State

Example: 0x48
Least significant nibble: 0x8 (find column)
Most significant nibble: 0x4 (find row)
Substitution value is 0x52

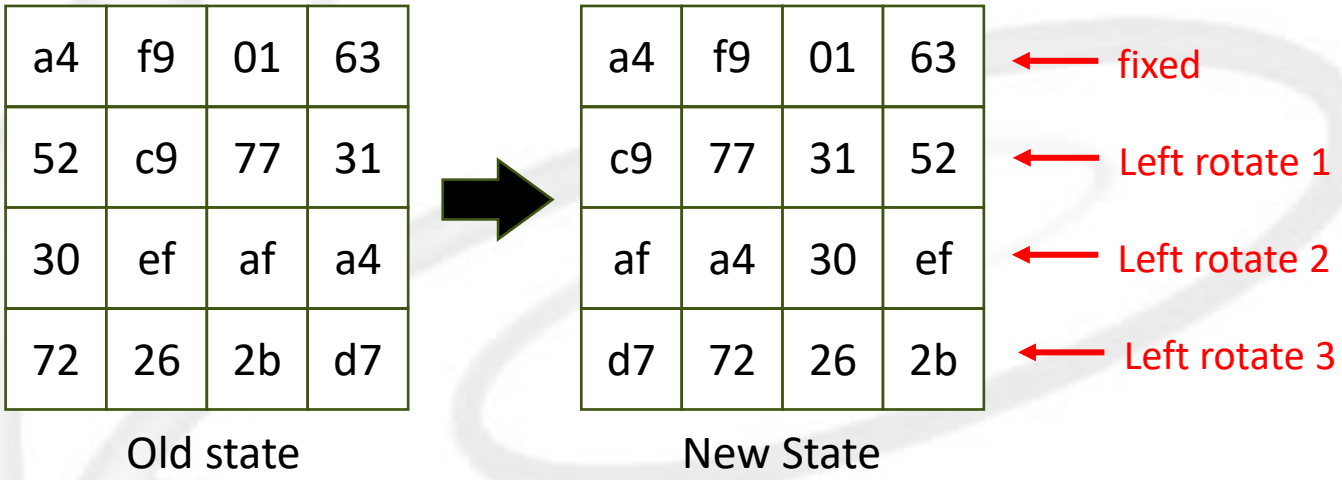
AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

- The S-box map is stored in ROM using row-major layout.

System Description (7/14)

- ShiftRows: shift array circularly



System Description (8/14)

- MixColumns: matrix multiplication and XOR addition

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Constant Matrix

a4	f9	01	63
c9	77	31	52
af	a4	30	ef
d7	72	26	2b

Old state

6b	a6	47	f4
10	92	15	c6
4a	4b	3a	89
24	27	4e	4e

New state

Understanding AES Mix-Columns Transformation Calculation

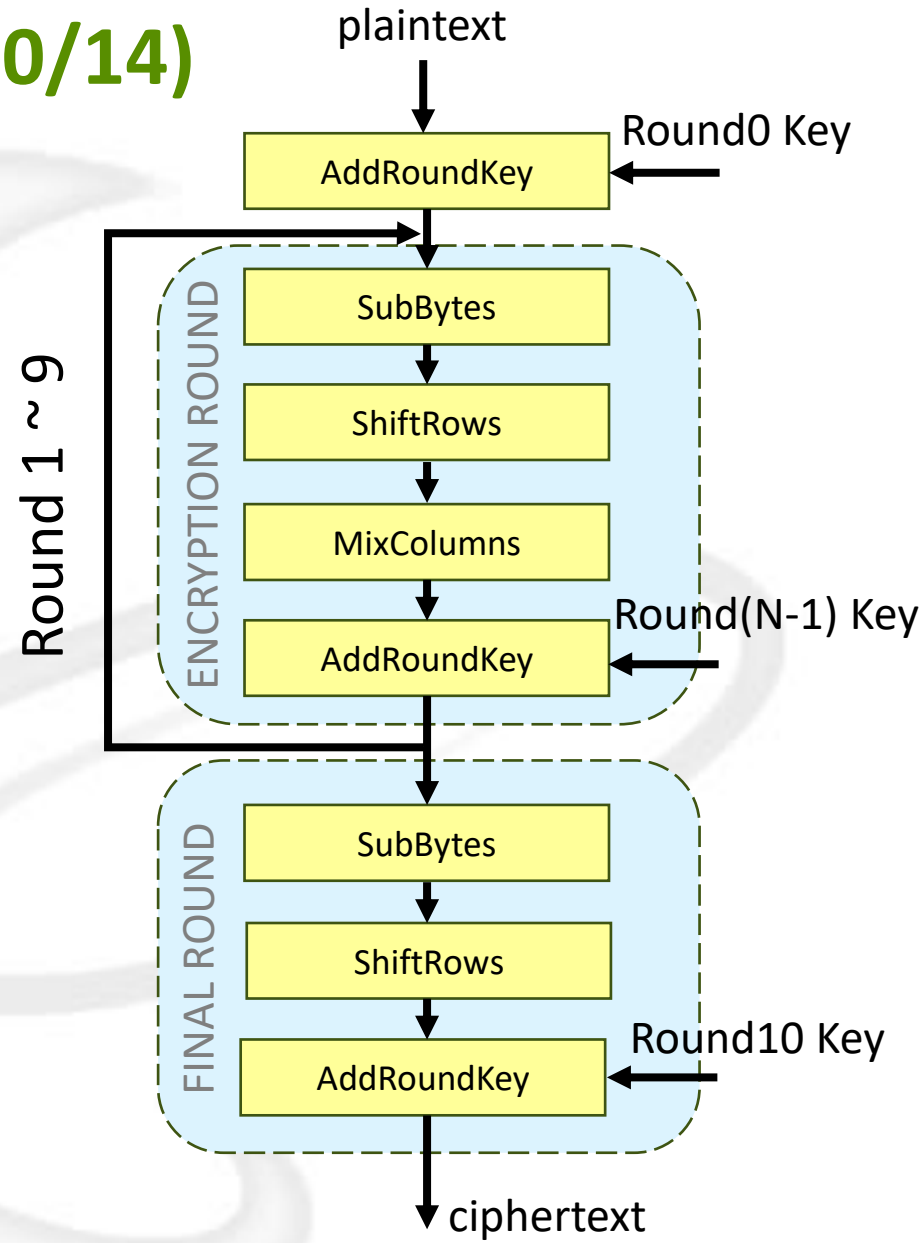
System Description (9/14) - KeyExpansion

- The AES key expansion algorithm takes the initial 128-bit cipher key as input to generate 10 round keys, each used for an AddRoundKey operation in the encryption process. This algorithm can be divided into four step:
  - RotWord: Perform a one-byte circular shift on the last column of the block key.
  - SubWord: Perform a byte substitution on each byte using S-box.
  - Rcon: Take the first byte and XOR it with the round constant, with each round having a different constant.

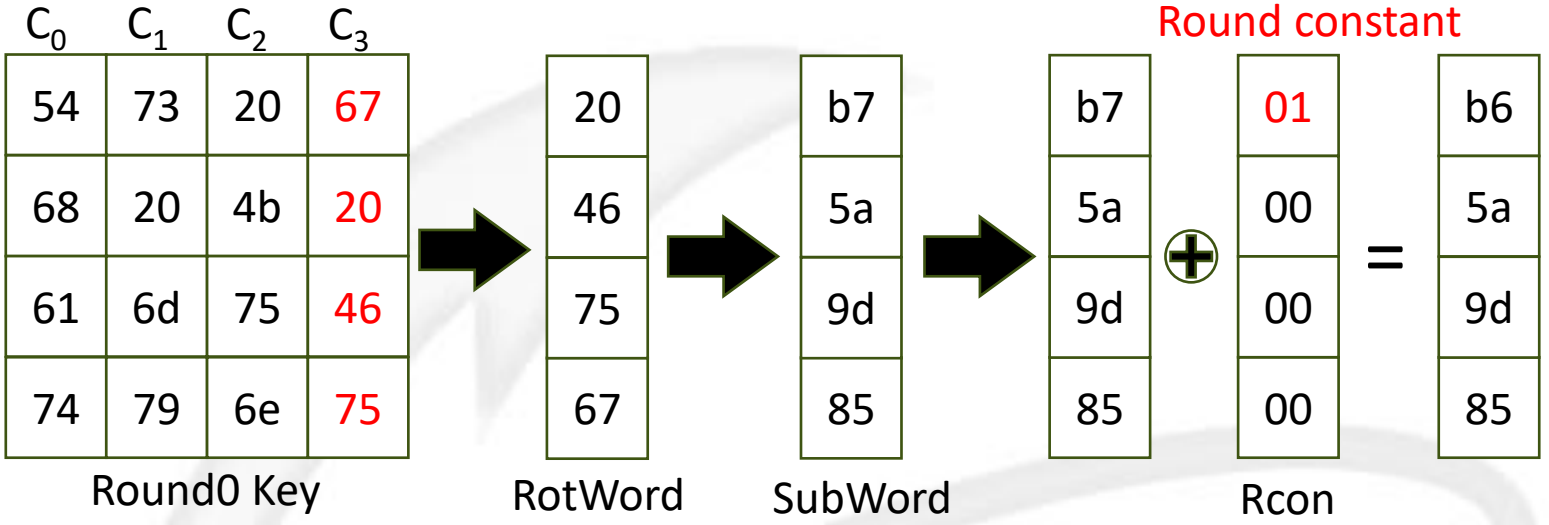
    [1, 2, 4, 8, 10, 20, 40, 80, 1b, 36] in hexadecimal

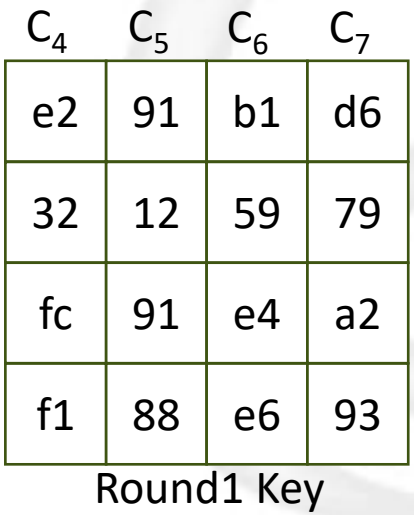
System Description (10/14)

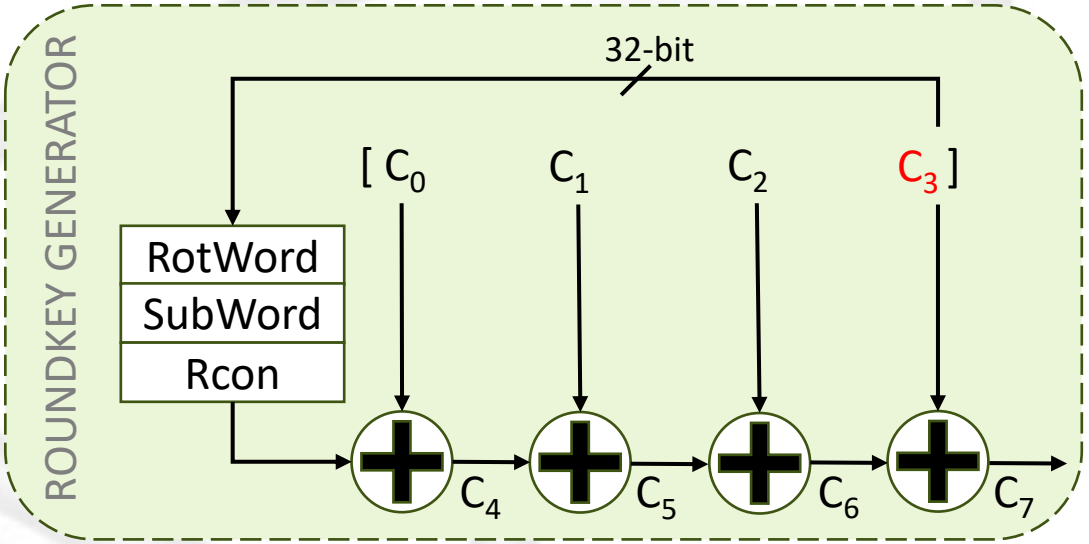
- AES flow diagram:
  - A total of 10 rounds are performed, with no MixColumns transformation required in the final round.
  - The Round 0 key is a secret key sent through the I/O port "key", while the other round keys are computed through the key expansion operation.
  - Design your own finite stat machine.



System Description (11/14) - KeyExpansion



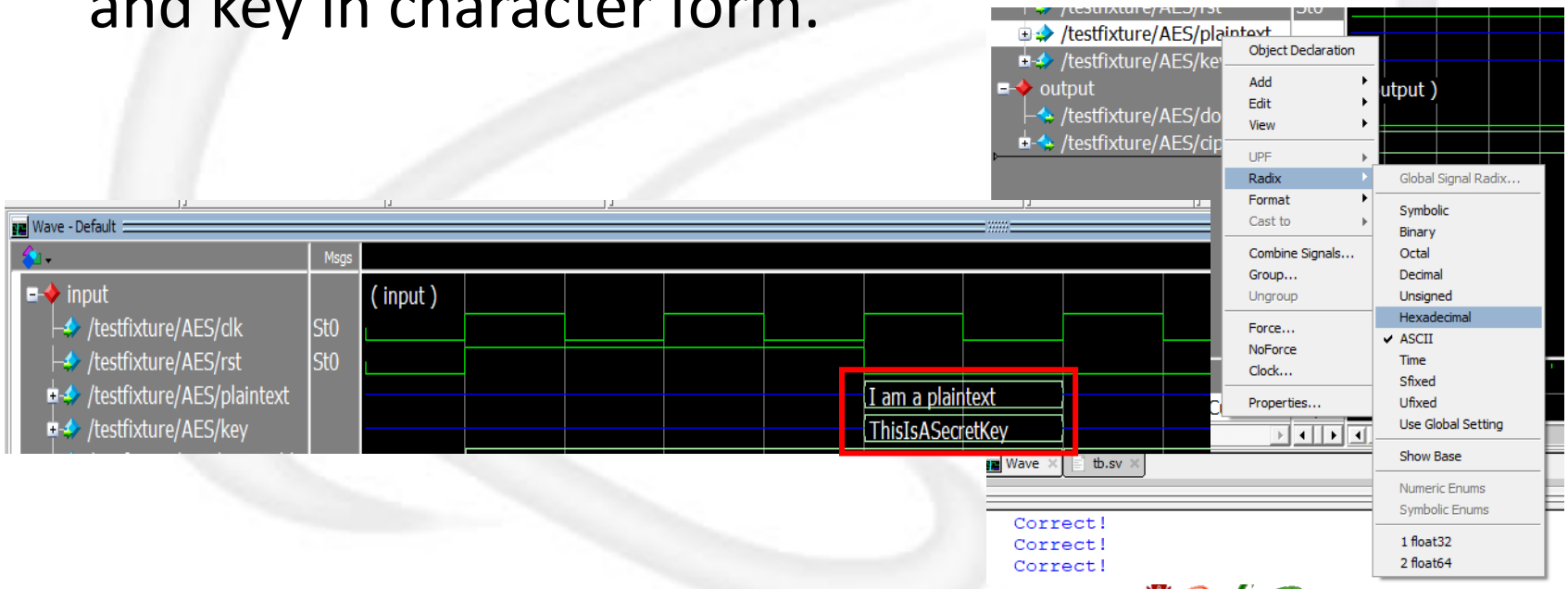




AES – Key Schedule/Key Expansion Explained

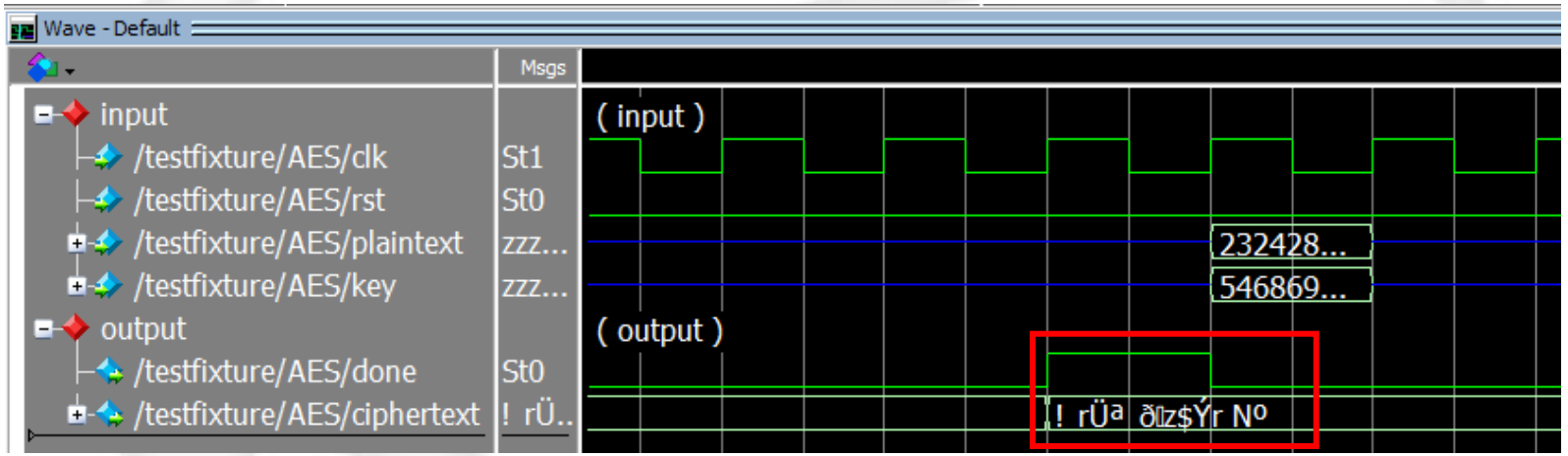
System Description (12/14)

- The first pattern will be sent promptly upon the assertion of the ‘rst’ signal to a low state. And each pattern will be transmitted within a single cycle.
- In pattern 1, you could change the waveform radix to ASCII. This allows you to visualize the plaintext and key in character form.



System Description (13/14)

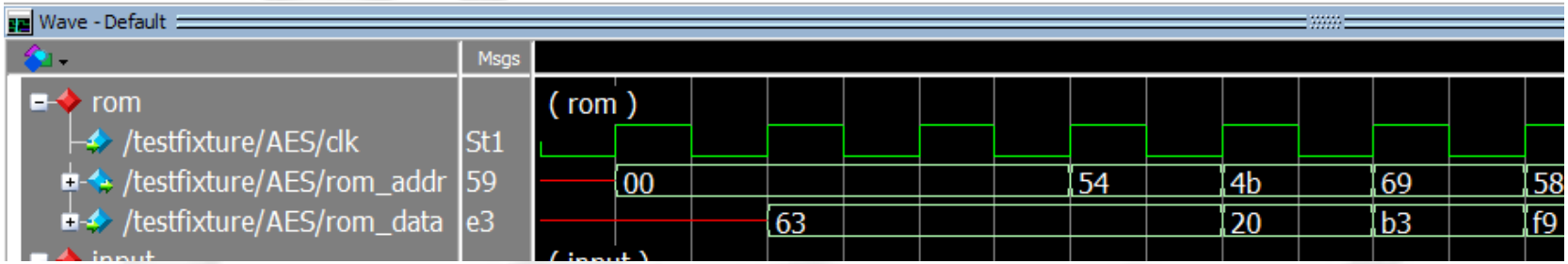
- Once the testbench detects that the output signal 'done' be asserted, the testbench will promptly verify the value of 'ciphertext'. Then the testbench will proceed to send the next pattern.



Verify ciphertext

System Description (14/14)

- All data in the ROM has been preloaded. When the ROM address is assigned, the ROM data will be transmitted back with a one clock cycle delay.



Address (hex)	Data (hex)
00	63
4b	b3
54	20
69	f9

Table. A few instances where addresses are mapped to data in the ROM

Outline

- Introduction
- Design Specifications
- System Description
- Criteria

Criteria (1/4)

- Three different patterns will be sent into your design. Please use the following command to verify that all patterns can pass without any errors.

- Define pattern:

Pattern	VSIM Command
Pattern1 (ASCII sentence)	vlog tb.sv +define+P1
Pattern2 (mount picture)	vlog tb.sv +define+P2
Pattern3 (tux picture)	vlog tb.sv +define+P3
All pattern	Without any define

➔ After defining each pattern command, please remember to type the command “restart” to ensure proper execution.

Criteria (2/4) – Simulation result

- Pattern1: Simulation pass

```
#    1/   1: Process...  Correct!
#
#
#
#
#          ****************************
#          **                        **       |__||
#          **  Congratulations !!     **      / O.O  |
#          **                        **    /_____   |
#          **  Simulation PASS!!      **   /^ ^ ^ \  |
#          **                        **  |^ ^ ^ ^ |w|
#          ****************************   \m___m__|_|
#
#
# Pattern name: Patter1 (I am a plaintext)
# ** Note: $stop    : tb.sv(155)
#    Time: 2435 ns  Iteration: 2  Instance: /testfixture
# Break in Module testfixture at tb.sv line 155
```

- Pattern2: Simulation pass

```
# 3072/3072: Process...  Correct!
#
#
#
#          ****************************
#          **                        **       |__||
#          **  Congratulations !!    **      / O.O  |
#          **                        **    /_____   |
#          **  Simulation PASS!!     **   /^ ^ ^ \  |
#          **                        **  |^ ^ ^ ^ |w|
#          ****************************   \m___m__|_|
# An encrypted image will be generated at the specified path.
# Pattern name: Patter2 (mount.bmp)
# Plot cipher image ...
# The image had been generate in the path: ./image/Cipher_mount.bmp
# ** Note: $stop    : tb.sv(155)
#    Time: 7434255 ns  Iteration: 2  Instance: /testfixture
# Break in Module testfixture at tb.sv line 155
```

- Pattern3: Simulation pass

```
# 3072/3072: Process...  Correct!
#
#
#
#          ****************************
#          **                        **       |__||
#          **  Congratulations !!    **      / O.O  |
#          **                        **    /_____   |
#          **  Simulation PASS!!     **   /^ ^ ^ \  |
#          **                        **  |^ ^ ^ ^ |w|
#          ****************************   \m___m__|_|
# An encrypted image will be generated at the specified path.
# Pattern name: Patter3 (tux.bmp)
# Plot cipher image ...
# The image had been generated in the path: ./image/Cipher_tux.bmp
# ** Note: $stop    : tb.sv(155)
#    Time: 7434255 ns  Iteration: 2  Instance: /testfixture
# Break in Module testfixture at tb.sv line 155
```

- Simulation pass

```
#    1/3072: Process...  Wrong!
#
# Golden:
# 0e  ee  68  4c
# c1  c8  3f  ee
# d7  ff  b0  c8
# b9  1a  fe  e2
#
# Your Cipher text
# xx  xx  xx  xx
# xx  xx  xx  xx
# xx  xx  xx  xx
# xx  xx  xx  xx
#
#
#
#
#          ****************************
#          **                        **       |__||
#          **  OOPS!!                **      / X,X  |
#          **                        **    /_____   |
#          **  Simulation Failed!!    **   /^ ^ ^ \  |
#          **                        **  |^ ^ ^ ^ |w|
#          ****************************   \m___m__|_|
# Pattern name: Patter3 (tux.bmp)
```

Display the first detected error for each operation and print the matrix for debugging.

Criteria (3/4)

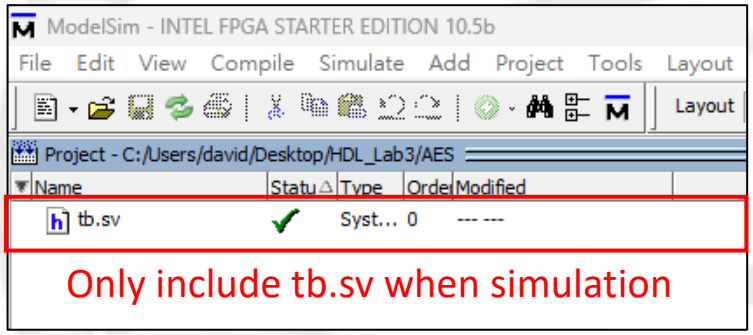
- Grading policy(100%)
  - Lab4
    - Simulation pass (90%)
      - Pattern 1 pass (50%)
      - Pattern 2 pass (20%)
      - Pattern 3 pass (20%)
    - Report (10%)

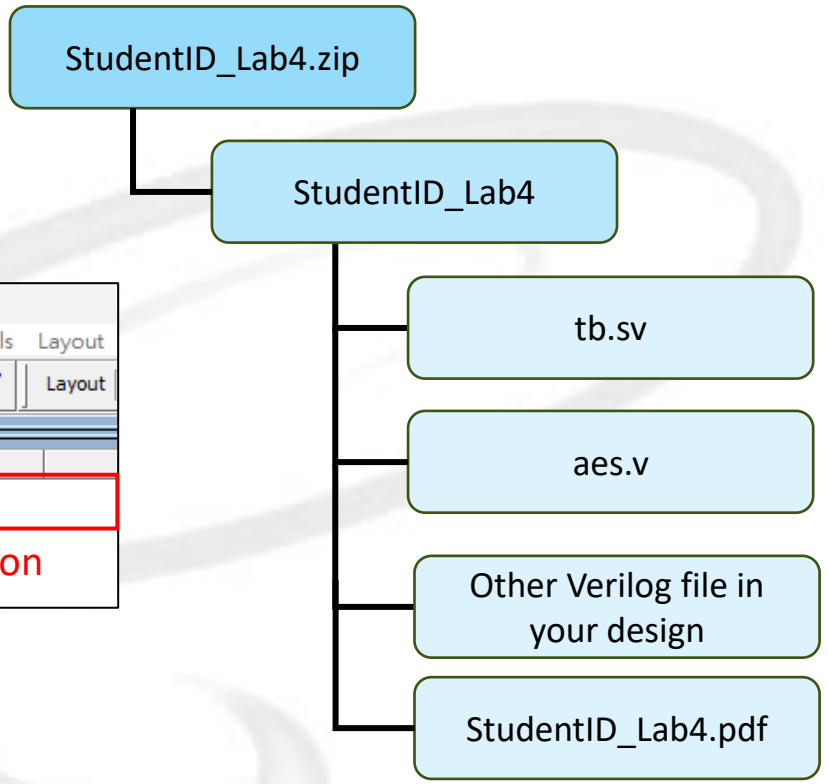
Criteria (4/4)

- Friendly reminder
  - Please complete the assignment by your own, discussion with peers is recommended, but do not cheat.
  - Warning! Any dishonesty found will result in zero grade.
  - Warning! Any late submission will also receive zero.
  - Warning! Please make sure that your code can be compiled in Modelsim, any dead body that we cannot compile will also receive zero.
  - Warning! Please submit your work according to the specified file format, making sure not to include any unnecessary files. Any unnecessary file found, will lead to 10% deduction from the overall score.
- Deadline: 2024/03/21 8:59 a.m.

Lab4 Requirement & file format

- File format
  - We will only include ‘tb.sv’ in ModelSim project to verify your design. Be cautious about the file path included in top module ‘aes.v’.



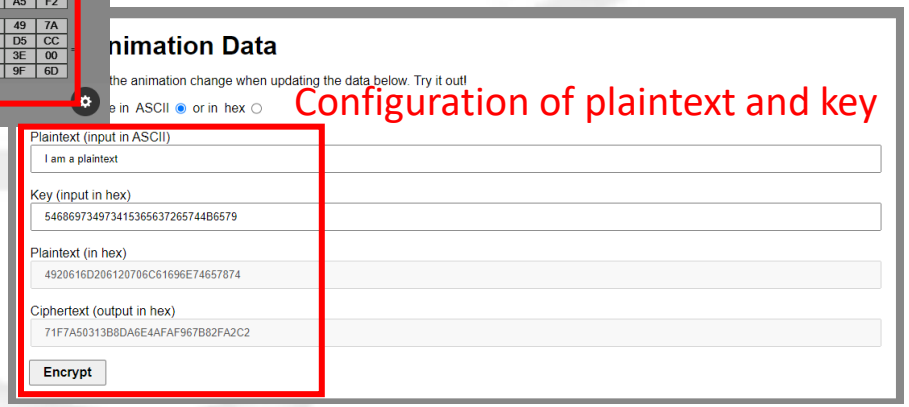


Reference

- CrypTool-Online: Animation Step by Step
- The provided configuration is equivalent to pattern1, you can verify your outcome using this website.

plaintext

Key Expansion



Configuration of plaintext and key

LPHPLAB VLSI Design LAB

Thanks for your attention !!