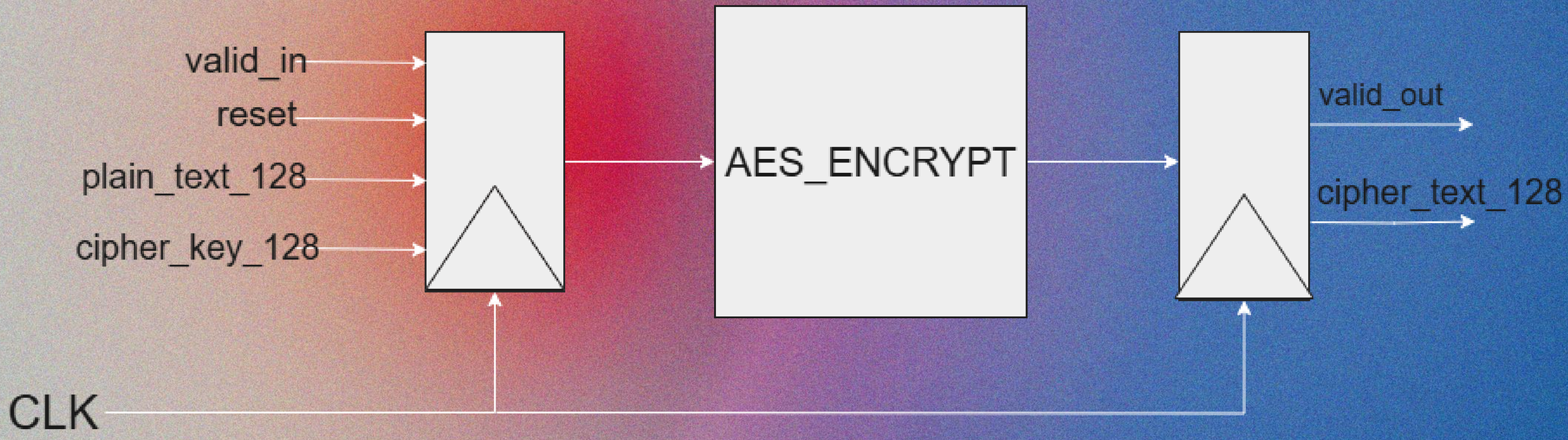
Full UVM environment of

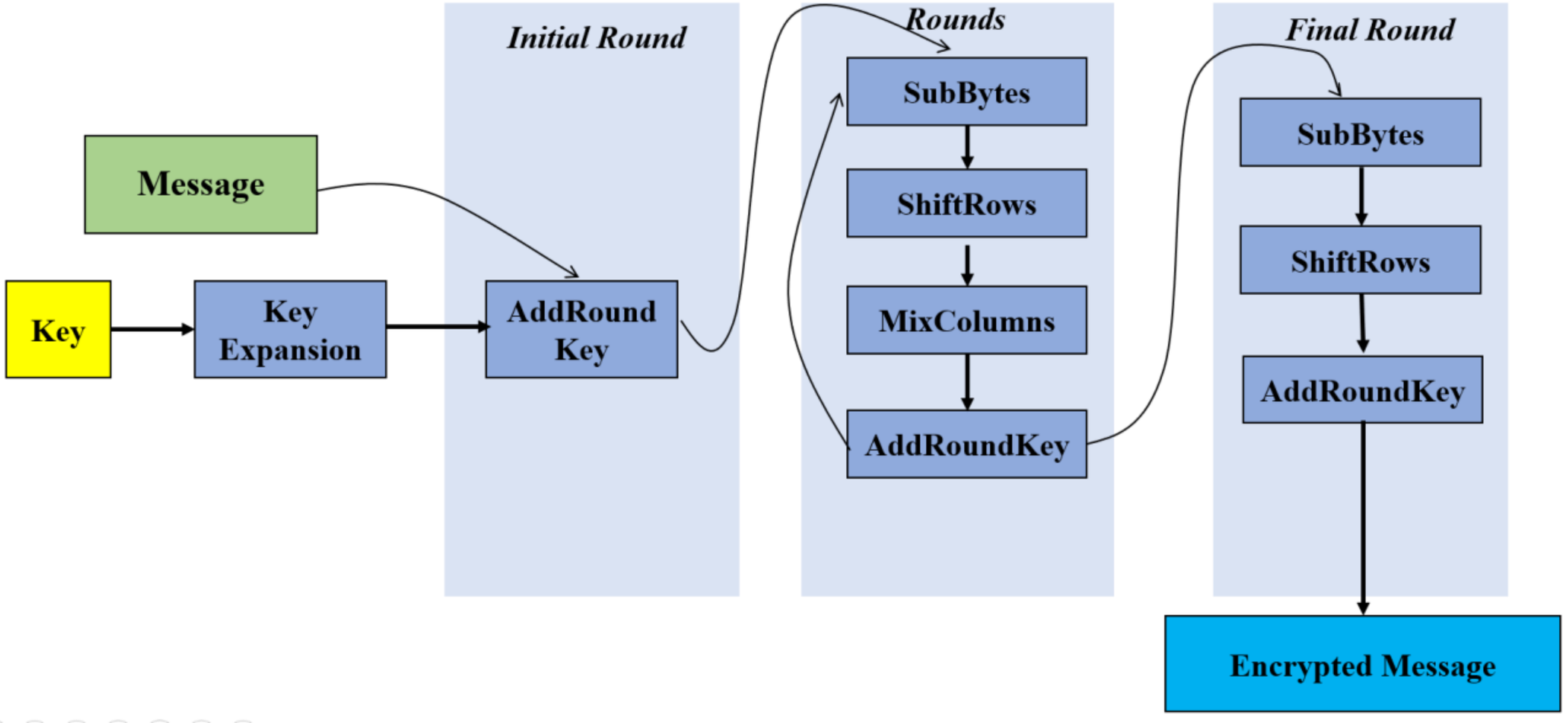
Advanced Encryption Standard (AES)



Done by: Mawaddah Mohammed

Table of contents

Steps of Encryption



IP design details of wrapper

Port	Direction	Function
Valid_in	**Input**	Signals that the incoming plain_text_128 and cipher_key_128 are valid and ready for encryption. Starts the encryption process when high.
Valid_out	**Output**	When output passes through the 2nd register, it gets high. Signals that cipher_text_128 is valid and ready to be read. Accounts for AES latency goes high when encryption result is available.
Reset	**Input(Active low sync)**	When low, clears all registers and output data.
Clk	**Input**	Clock signal that drives the sequential logic inside the module
Plain_text_128	**Input**	128-bit input data block to be encrypted
Cipher_key_128	**Input**	128-bit secret key used for AES encryption-→ **10 cycles**
Cipher_text_128	**output**	128-bit output block containing the AES-encrypted data

Verification strategy

Constraint random testing and Direct testing (corner cases)

wrapper.sv

```
19 always @(posedge clk)begin
21 valid_in_output_reg<=0;
22 cipher_key_128_reg<=0;
23 plain_text_128_reg<=0;
27 plain_text_128_reg<=plain_text_128;
28 cipher_key_128_reg<=cipher_key_128;
30 valid_in_output_reg<=valid_in;
33 always @(posedge clk)begin
35 cipher_text_128<=0;
39 cipher_text_128<=cipher_text_128_reg;
40 valid_out<=1;
43 valid_out<=0;
```

Branches - by instance (/top/wrapper)

wrapper.sv

```
20 if(!reset)begin
25 else begin
26 if(valid_in)begin
34 if(!reset)begin
37 else begin
38 if(valid_in_output_reg)begin
42 else begin
```

Toggles - by instance (/top/wrapper)

- sim:/top/wrapper
  - cipher_key_128
  - cipher_key_128_reg
  - cipher_text_128
  - cipher_text_128_reg
  - clk
  - plain_text_128
  - plain_text_128_reg
  - reset
  - valid_in
  - valid_in_output_reg
  - valid_out

```
# ** Report counts by severity
# UVM_INFO :   22
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
```

Exit Criteria

1. 100% functional Coverage
2. 100% Code Coverage
3. 0 bugs

/pack/subscriber		100.0%			
TYPE g1	subscriber	100.0%	100	100.0%	auto(1)
CVP g1::valid_in_cp	subscriber	100.0%	100	100.0%	
CVP g1::reset_cp	subscriber	100.0%	100	100.0%	
CVP g1::plain_text_128_cp	subscriber	100.0%	100	100.0%	
CVP g1::cipher_key_128_cp	subscriber	100.0%	100	100.0%	
CROSS g1::plain_text_key_cc	subscriber	100.0%	100	100.0%	
bin mix_alt_1		1	1	100.0%	
bin mix_alt_2		1	1	100.0%	

valid_in

- Set to 1 → Accepts plain_text_128 and cipher_key_128 for encryption.
- Set to 0 → No new encryption input is latched.

clk

- Normal operation: Provide active clock edges for sequential logic.
- Test with stopped or irregular clock to check stability (optional robustness test).

reset

- Set to 0 → Clears cipher_text_128, valid_out, internal registers, and queues.
- Set to 1 → Normal operation; retains and processes input data.

plain_text_128

- Provide 128-bit test vectors (e.g., NIST known plaintexts, all-zeros, all-ones, alternating bits).
- Check that output matches expected ciphertext from reference model.

cipher_key_128

- Provide 128-bit keys (random, fixed patterns, NIST keys).
- Verify encryption output changes appropriately with different keys.

valid_out (output)

- When 1 → cipher_text_128 is valid and ready to be read.
- When 0 → Output is not valid; ignore cipher_text_128.

cipher_text_128 (output)

- Verify against golden reference model output for each input and key.
- Test with multiple latency cycles to ensure data integrity.

Test Items

Test Sequence	Description	Purpose
reset_sequence	Drives reset low (0) to reset DUT outputs and clear internal states.	Verify DUT reset functionality.
valid_sequence	Drives reset high (1) and sets valid_in high to send valid input data.	Check DUT behavior when receiving valid data.
random_sequence	Generates and drives 30 random plaintext-key pairs to DUT.	Stress test DUT with random data patterns.
all_zeros_sequence	Sends plaintext and key as all zeros.	Verify encryption correctness with all-zero inputs.
all_ones_sequence	Sends plaintext and key as all ones (0xFF).	Verify encryption correctness with all-one inputs.
repeated_key_sequence	Uses same key with different plaintext values.	Check DUT with repeated key usage.
consecutive_key_sequence	Uses consecutive keys (e.g., 500, 501) with random plaintexts.	Test DUT's key schedule correctness for near keys.
alternating_sequence	Uses alternating bit patterns between plaintext and key (0x55/0xAA).	Test DUT's sensitivity to alternating patterns.

Traceability Matrix

Test Case	valid_in	reset	plain_text_128	cipher_key_128
reset_sequence	Randomized	0 (reset active)	Randomized	Randomized
valid_sequence	1	1	Randomized	Randomized
random_sequence	Randomized	Randomized	Randomized	Randomized
all_zeros_sequence	Randomized	1	All 0s	All 0s
all_ones_sequence	Randomized	1	All 1s	All 1s
repeated_key_sequence	Randomized	1	Fixed value (changes once)	Same key both times
consecutive_key_sequence	Randomized	1	Randomized	500 then 501
alternating_sequence	Randomized	1	Alternating 0x55 / 0xAA	Alternating 0xAA / 0x55

Thank you