COL 215 Hardware Assignment 3-Part 2

Submission by: -

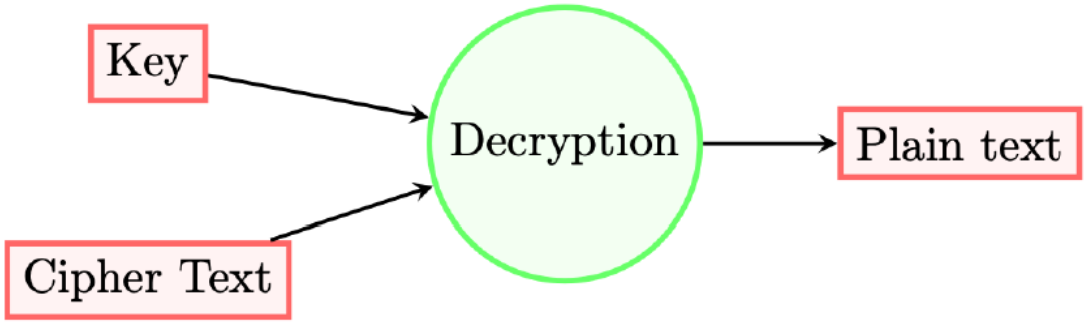
1. Aaditya Sharma(2023CS10420)
2. Namit Jain(2023CS10483)

Submitted On: - 10th October, 2024

Introduction

The main objectives are to develop a controller/finite state machine (FSM) to manage memory operations, execute decryption rounds, and display the decrypted plain text on a 7-segment display. The decryption process involves a series of transformations to revert the encrypted data (ciphertext) back to plaintext. The main steps in AES decryption include **Inverse ShiftRows**, **Inverse SubBytes**, **Inverse MixColumns**, and **AddRoundKey**. The decrypted output is displayed on a 7-segment display, cycling through 32-bit hexadecimal data.

The project is modular, with each AES decryption operation implemented as a separate module, controlled by a **Finite State Machine (FSM)** within the controller. This approach allows for sequential execution of each step, simplifying testing and debugging.



Approach

Controller (FSM)

The **controller** orchestrates each step in AES decryption. It progresses through states like ReadInput, InvShiftRows, InvSubBytes, XOR_RoundKey, InvMixColumns, seg_display, WriteOutput, and Completed. It uses counters to manage iterations within each state,

ensuring that all 16 bytes of the AES state matrix are processed correctly.

1. **ReadInput**: Reads ciphertext from memory into internal_state.
2. **InvShiftRows**: Applies the inverse row shifts to internal_state.
3. **InvSubBytes**: Substitutes bytes in internal_state using the inverse sbox.
4. **XOR_RoundKey**: XORs internal_state with the round key to reverse encryption.
5. **InvMixColumns**: Mixes columns in internal_state using Galois Field arithmetic.
6. **WriteOutput**: Writes the decrypted data back to memory.
7. **seg_display**: Sends data to the 7-segment display, cycling through 32-bit hexadecimal data.
8. **Memory**: -
    - BRAM has been used exclusively for this project.
    - We used blk_mem_gen_0 for round keys. It has read/write width of 8 bits and read/write depth of 160 bits.
    - We used blk_mem_gen_1 for reading input and writing output which stores input file. It has read/write width of 8 bits and read/write depth of 16 bits for 128 bit input and 32 bits for 256 bit inputs.
    - blk_mem_gen_2 is used in inv_sub_byte which stores inv_sbox. It has read/write width of 8 bits and read/write depth of 256 bits.

AES Decryption Overview :-

AES (Advanced Encryption Standard) is a symmetric encryption algorithm that performs encryption and decryption in 128-bit blocks using inverse transformations during decryption. The process is structured as follows:

1. Key Expansion :- The encryption key is expanded into round keys.
2. Initial round :-
    - AddRoundKey: Combines the state with the round key through a bitwise XOR operation.
    - InvShiftRows
    - InvSubBytes
3. Following 8 rounds :
    - XOR_operation
    - InvMixCols : Applies matrix multiplication with constants in Galois Field to reverse column mixing.
    - InvShiftRows: Shifts rows cyclically to the right, reversing the encryption's row shifts.
    - InvSubBytes : Replaces each byte with its inverse_sbox value, reversing the encryption's byte substitution.
4. Last round :
    - XOR operation.

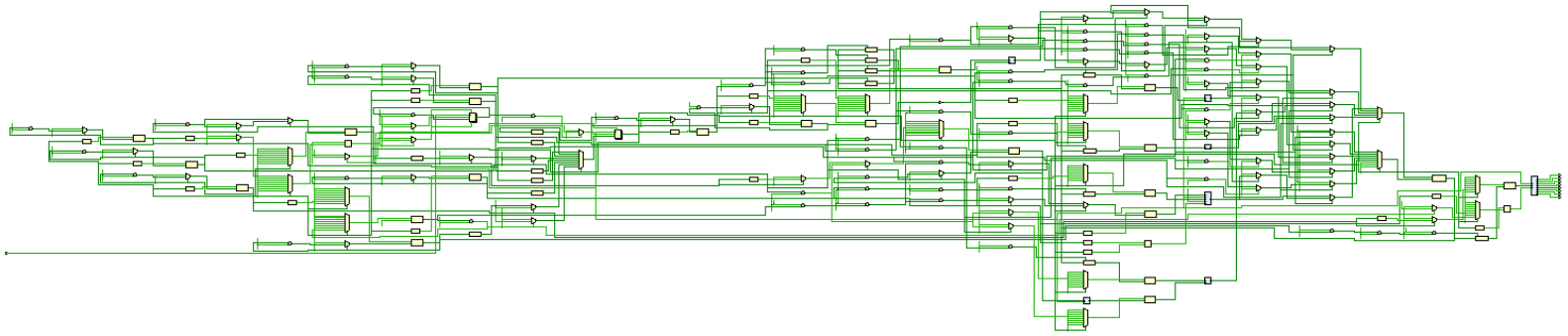
Functionality of Each Module

1. **Round Key Reader**:
    - Fetches round keys from memory in reverse order for decryption.
    - Addresses are calculated to retrieve the appropriate key bytes for each round.
2. **InvSubByte**:

- Performs byte substitution using the inverse S-Box, a required transformation in AES decryption.
- Uses a memory lookup for each byte, reversing the effect of the SubBytes step in encryption.

3. **InvShiftRows**:
   - Reverses the row shifts applied during encryption.
   - The mux4x1 multiplexer shifts each row by a different number of bytes based on the row index.
4. **InvMixColumns**:
   - Applies matrix multiplication in Galois Field (GF(2^8)) to reverse column mixing.
   - Each column of internal_state is transformed individually.
5. **XOR Operation (AddRoundKey)**:
   - XORs each byte of internal_state with the corresponding byte from the round key.
   - This step reverses the AddRoundKey operation in AES encryption.
6. **SegDisplay**:
   - Controls a 7-segment display to show the decrypted output in hexadecimal format.
   - Uses a clock divider and multiplexing to display four digits at a time, cycling through the full 32-bit output.

Schematics :-



Schematic of Controller

Resource Utilisation: -

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs	2250	0	0	20800	10.82
LUT as Logic	2250	0	0	20800	10.82
LUT as Memory	0	0	0	9600	0.00
Slice Registers	650	0	0	41600	1.56
Register as Flip Flop	87	0	0	41600	0.21
Register as Latch	563	0	0	41600	1.35
F7 Muxes	80	0	0	16300	0.49
F8 Muxes	40	0	0	8150	0.49

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
4	Yes	-	Set
573	Yes	-	Reset
1	Yes	Set	-
72	Yes	Reset	-

2. Slice Logic Distribution

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice	767	0	0	8150	9.41
SLICEL	549	0			
SLICEM	218	0			
LUT as Logic	2250	0	0	20800	10.82
using O5 output only	9				
using O6 output only	1757				
using O5 and O6	484				
LUT as Memory	0	0	0	9600	0.00
LUT as Distributed RAM	0	0			
LUT as Shift Register	0	0			
Slice Registers	650	0	0	41600	1.56
Register driven from within the Slice	411				
Register driven from outside the Slice	239				
LUT in front of the register is unused	100				
LUT in front of the register is used	139				
Unique Control Sets	43		0	8150	0.53

* * Note: Available Control Sets calculated as Slice * 1, Review the Control Sets Report for more information regarding control sets.

3. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	1.5	0	0	50	3.00
RAMB36/FIFO*	0	0	0	50	0.00
RAMB18	3	0	0	100	3.00
RAMB18E1 only	3				

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

4. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	90	0.00

5. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	12	12	0	106	11.32
IOB Master Pads	6				
IOB Slave Pads	6				
Bonded IPADs	0	0	0	10	0.00
Bonded OPADs	0	0	0	4	0.00
PHY_CONTROL	0	0	0	5	0.00
PHASER_REF	0	0	0	5	0.00
OUT_FIFO	0	0	0	20	0.00
IN_FIFO	0	0	0	20	0.00
IDELAYCTRL	0	0	0	5	0.00
IBUFDS	0	0	0	104	0.00
GTPE2_CHANNEL	0	0	0	2	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	20	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	20	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	250	0.00
IBUFDS_GTE2	0	0	0	2	0.00
ILOGIC	0	0	0	106	0.00
OLOGIC	0	0	0	106	0.00

6. Clocking

Site Type	Used	Fixed	Prohibited	Available	Util%
BUFGCTRL	12	0	0	32	37.50
BUFIO	0	0	0	20	0.00
MMCME2_ADV	0	0	0	5	0.00
PLLE2_ADV	0	0	0	5	0.00
BUFMRCE	0	0	0	10	0.00
BUFHCE	0	0	0	72	0.00
BUFR	0	0	0	20	0.00

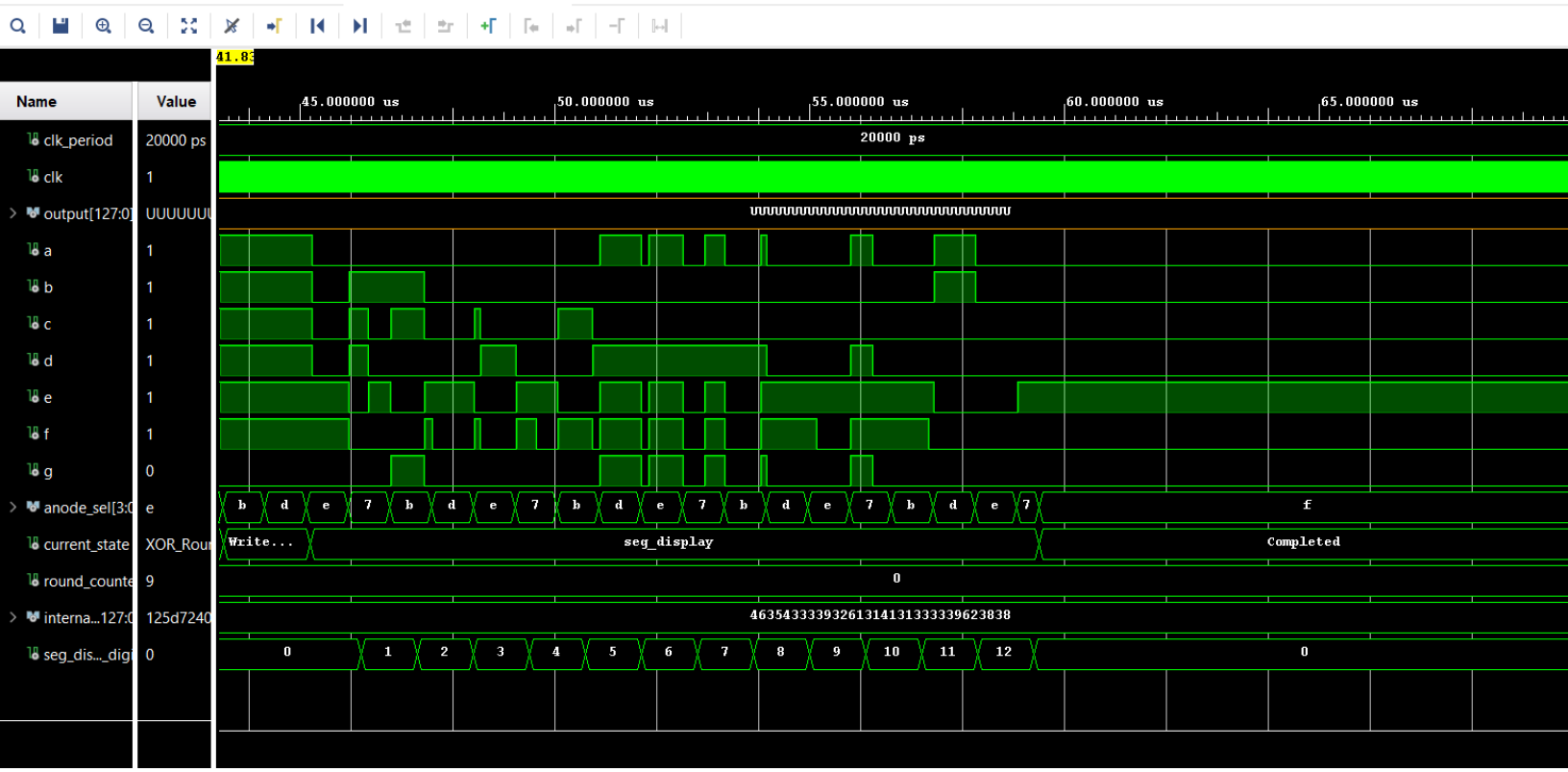
7. Specific Feature

Site Type	Used	Fixed	Prohibited	Available	Util%
BSCANE2	0	0	0	4	0.00
CAPTUREE2	0	0	0	1	0.00
DNA_PORT	0	0	0	1	0.00
EFUSE_USR	0	0	0	1	0.00
FRAME_ECCE2	0	0	0	1	0.00
ICAPE2	0	0	0	2	0.00
PCIE_2_1	0	0	0	1	0.00
STARTUPE2	0	0	0	1	0.00
XADC	0	0	0	1	0.00

8. Primitives

Ref Name	Used	Functional Category
LUT6	1113	LUT
LUT2	565	LUT
LDCE	563	Flop & Latch
LUT5	432	LUT
LUT4	249	LUT
LUT1	198	LUT
CARRY4	186	CarryLogic
LUT3	177	LUT
MUXF7	80	MuxFx
FDRE	72	Flop & Latch
MUXF8	40	MuxFx
BUFG	12	Clock
OBUF	11	IO
FDCE	10	Flop & Latch
FDPE	4	Flop & Latch
RAMB18E1	3	Block Memory
IBUF	1	IO
FDSE	1	Flop & Latch

Simulations: -



The Decrypted hex result is 463543333932613141313333396238 38. Hence the output is F5C392a1A1339b88 which matches the 128 bit CM expected output.

a, b, c, d, e, f, g are the cathodes and anode_sel gives the anode.

The Decrypted hex result is 463362326638303346423942346533 66 and 33446145336465413731414132436330. Hence the output is F3b2f803FB9B4e3f3DaE3deA71AA2Cc0 which matches the 256 bit CM expected output.

a, b, c, d, e, f, g are the cathodes and anode_sel gives the anode.

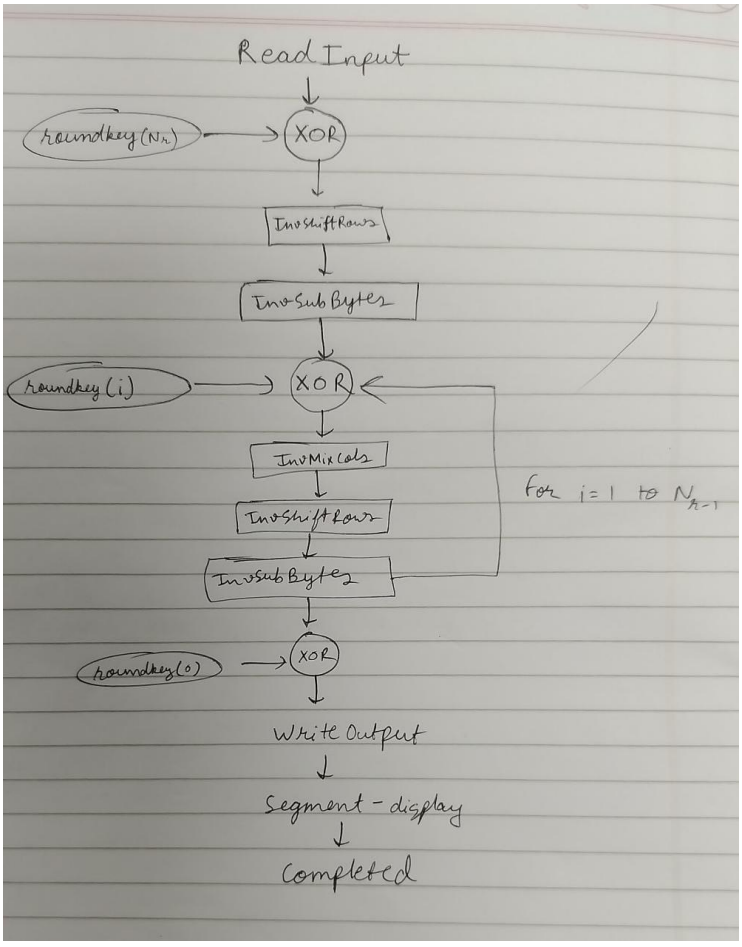
We reduced the value of “segment_display_ratio to 10” and max_count(in segment display) to 40 so that their working is visible.

Value of segment_display_ratio is 50000000 and max count is 50000.

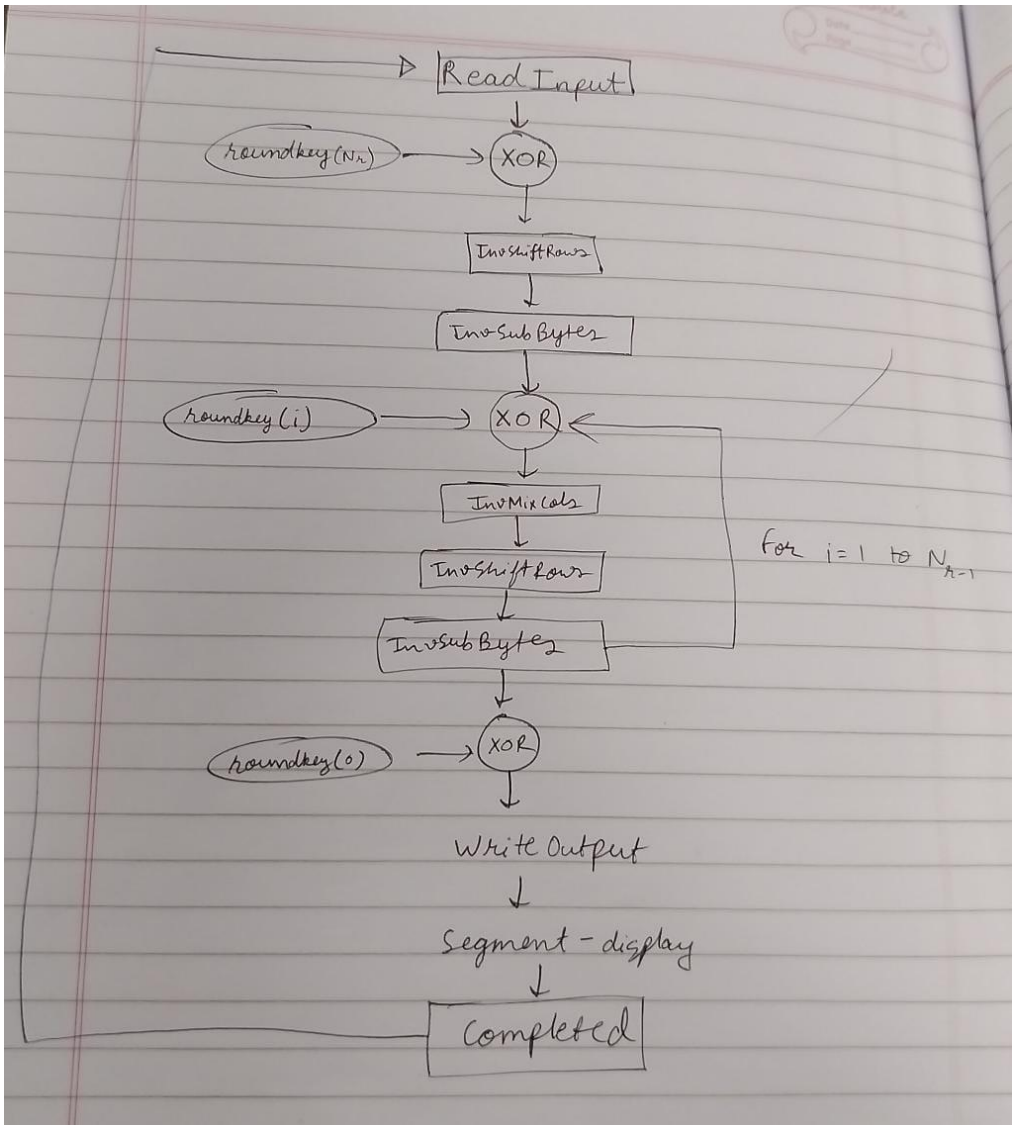
We chose these values because 10ns (duration of each clock cycle) * 4(no. of clock cycles given to each state before going to next state) * 50000 = 2ms.

And 50000000 because it is 2 seconds for displaying a set of 4 digits.

FSM Diagram



For 128 bit input



For 128*n bits

The Completed state will go back to the ReadInput state while changing an **internal counter** that is a measure of the number of 128 bit parts in the entire input and hence the first 16 digits of the output will be displayed, and then the calculation for the next set of output will be done (and since it takes a couple of micro seconds, we won't see any lag between consecutive outputs). The scrolling-display is continuous for every set of 16 digits. Every time we transition from one set of outputs

to another, the scrolling skips 3 positions and begins for the next set of output.

Note: - The code is in accordance with column major notation only.