COL-215
Hardware Assignment - 3 Report

Shreeraj Jambhale 2023CS50048
Pallav Kamad 2023CS51067

Table Of Contents

1 Build Seven Segment Decoder

1.1 Aim

The aim of this assignment is to implement a Finite State Machine (FSM) for AES decryption on a Basys 3 FPGA board. The FSM controls the sequence of operations required for AES decryption, including key scheduling, inverse transformations, and data flow management. The design focuses on:

- Managing state transitions between different AES decryption stages
- Coordinating memory access for input data and round keys
- Orchestrating the inverse transformations (InvSubBytes, InvShiftRows, InvMixColumns)
- Controlling the display output of decrypted data

1.2 Approach

The FSM implementation follows a structured approach with distinct states and controlled transitions:

1.2.1 Round Key – XOR

- Input Setup: Take an 8-bit input from the AES state matrix (1 hexadecimal code).
- Round Key: Input the corresponding 8-bit round key for the current decryption round.
- XOR Operation: Perform a bitwise XOR operation between the 8-bit input and the 8-bit round key.
- Output: The result of the XOR operation forms part of the decrypted output for this step.

1.2.2 Inverse Shift Rows - InvShiftRows

The InvShiftRows transformation is the inverse of the Shift Rows step in AES encryption. In InvShiftRows, the bytes in each row of the state are shifted to the right by different offsets, in our implementation we provide one row at a time to the module and also provide the row number so to inform about the shift number

- Input: row_in (32 bits) representing the current row.
- Shift index: shift_idx (2 bits) to specify the shift amount.
- Output: row_out (32 bits) as the shifted row.

Used if-else blocks to execute the process

- If shift_idx is "00", no shift is applied.
- If shift_idx is "01", it shifts by 8 bits (1 byte).
- If shift_idx is "10", it shifts by 16 bits (2 bytes).
- If shift_idx is "11", it shifts by 24 bits (3 bytes).

1.2.3 Inverse Sub Bytes – Inv S-Box

This code implements an inverse S-Box lookup function in AES decryption using a single-port ROM. Here's a breakdown of the approach and how it works:

- Input Splitting: The 8-bit input is split into two 4-bit parts, dig1 and dig2. This is done to access the specific row and column of the S-Box.
- Address Formation: The dig2 (lower 4 bits) and dig1 (higher 4 bits) are concatenated to form an 8-bit intermediate_add, which serves as the address for the ROM.
- ROM Lookup: The rom_invSbox component is instantiated as a single-port ROM, using intermediate_add as the address input (addra). This ROM is generated using a Block Memory Generator, preloaded with the inverse S-Box values.
- Output: The douta from the ROM outputs the S-Box-transformed value for the given input.

1.2.4 Inv Mix Columns

1.2.4.1 GF8_2Mul

The GF8_2Mul function multiplies two 8-bit vector inputs within the Galois Field returning an 8-bit result. The function achieves this by applying bitwise shifts and XOR operations to perform the multiplication and reduction steps.

1.2.4.2 Inv Mix Col

The Inv Mix Col function leverages the GF8_2Mul function to compute the Galois Field sum of four separate products. It combines the outputs from four calls to GF8_2Mul to produce a final 8-bit vector output, which replaces a single hexadecimal value in the 4x4 matrix during the transformation process.

1.2.5 Display

This VHDL project converts a 32-bit hexadecimal input into ASCII characters and scrolls the result across four 7-segment displays on the Basys 3 board, showing four characters at a time with a 1-second interval. The purpose is to enable full data visibility by cycling through the ASCII-converted data continuously. Key steps include:

- Clock Divider: Generates a 1-second delay for smooth scrolling.
- Scrolling Index: Selects and updates the displayed 4-character segment every second.
- ASCII Conversion: Maps each 4-bit hexadecimal digit to its ASCII equivalent for display.
- Display Logic: Cycles through the ASCII characters, presenting them on the 7-segment displays.

This design enables scrolling for large ASCII data on limited hardware.

Digit	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1
A/a	1	1	1	0	1	1	1
B/b	0	0	1	1	1	1	1
C/c	1	0	0	1	1	1	0
D/d	0	1	1	1	1	0	1
E/e	1	0	0	1	1	1	1
F/f	1	0	0	0	1	1	1
-	1	1	1	1	1	1	0

1.2.6 State Organization The FSM is organized into the following main states:

- IDLE: Initial state, waiting for start signal
- LOAD_INPUT: Loads input ciphertext from ROM
- LOAD_KEY: Retrieves round keys from key memory
- XOR_STATE: Performs AddRoundKey transformation
- SUBBYTES_STATE: Handles inverse S-box substitution
- SHIFTROWS_LOAD/PROCESS/STORE: Manages inverse shift rows operation
- MIXCOLS_STATE: Controls inverse mix columns transformation
- WRITE_RESULT: Updates state matrix with round results
- DISPLAY_STATE: Prepares output for seven-segment display
- DONE_STATE: Signals completion of decryption

1.2.7 Memory Management

- Input ROM: Stores the ciphertext to be decrypted
- Round Key ROM: Contains pre-computed round keys
- State Matrix: Internal storage for intermediate results
- Working Registers: 32-bit registers for temporary data storage

1.2.8 Control Logic

- Counter Management:
    - Round counter (0 to 9) for tracking decryption rounds
    - Word counter (0 to 3) for matrix column operations
    - Byte counter (0 to 3) for byte-level operations
    - Row counter (0 to 3) for row operations

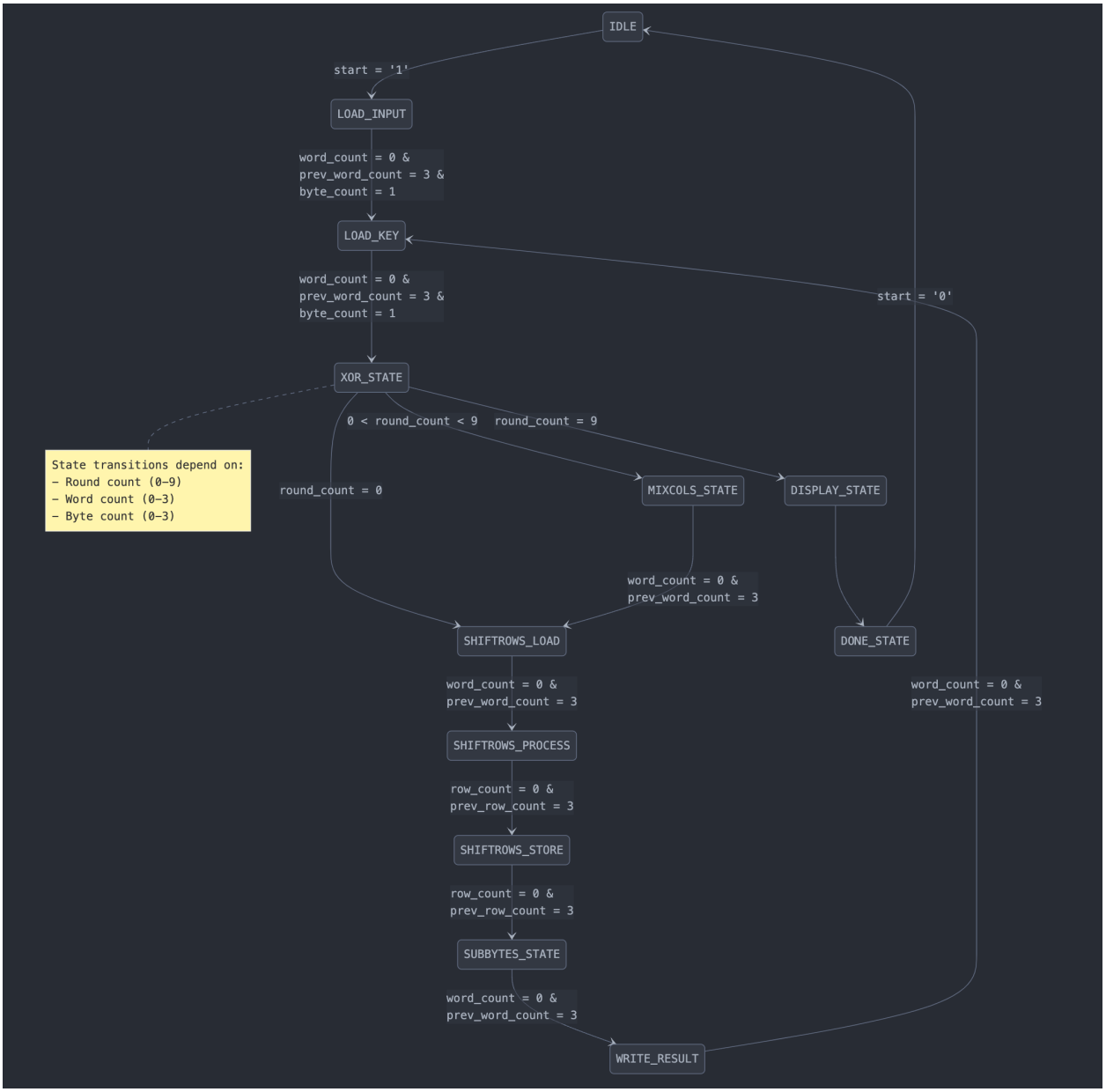
1.2.9 State Transitions

- Transition conditions based on multiple counters
- Synchronous state updates on clock edges
- Reset handling for initialization
- Completion detection and done signal assertion

1.2.10 Data Flow Control

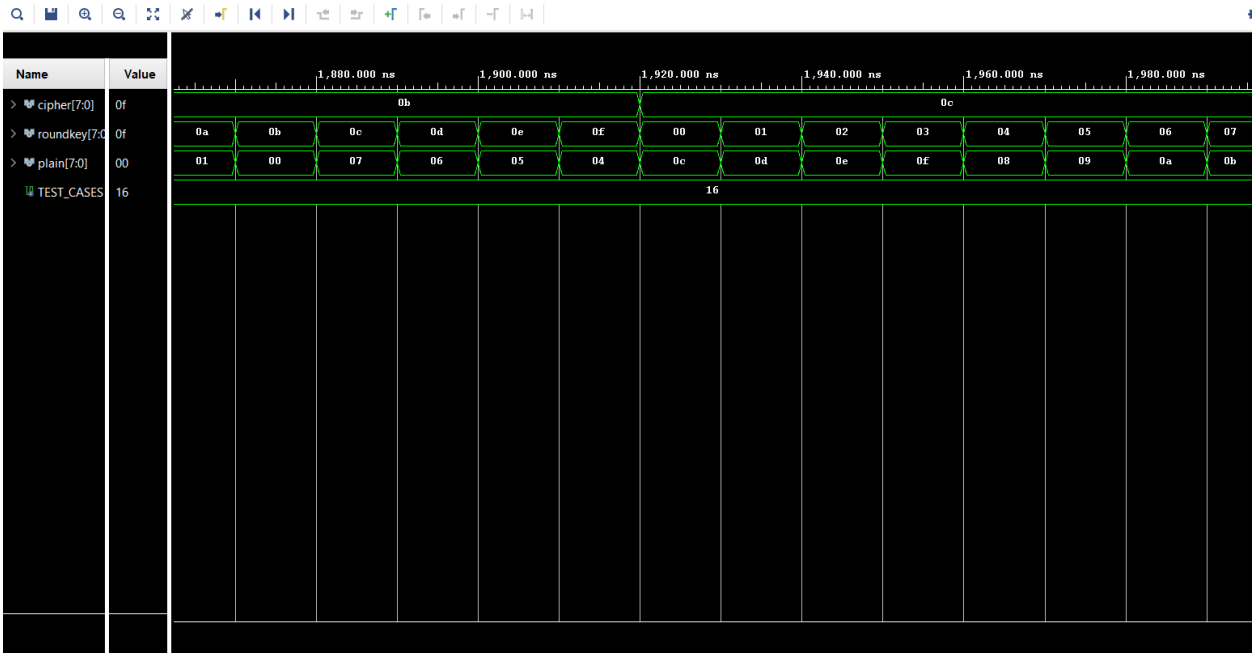
- Column-major matrix operations
- Byte reordering for different transformations
- Pipeline management between transformation stages
- Output formatting for display

1.3 FLOW Diagram



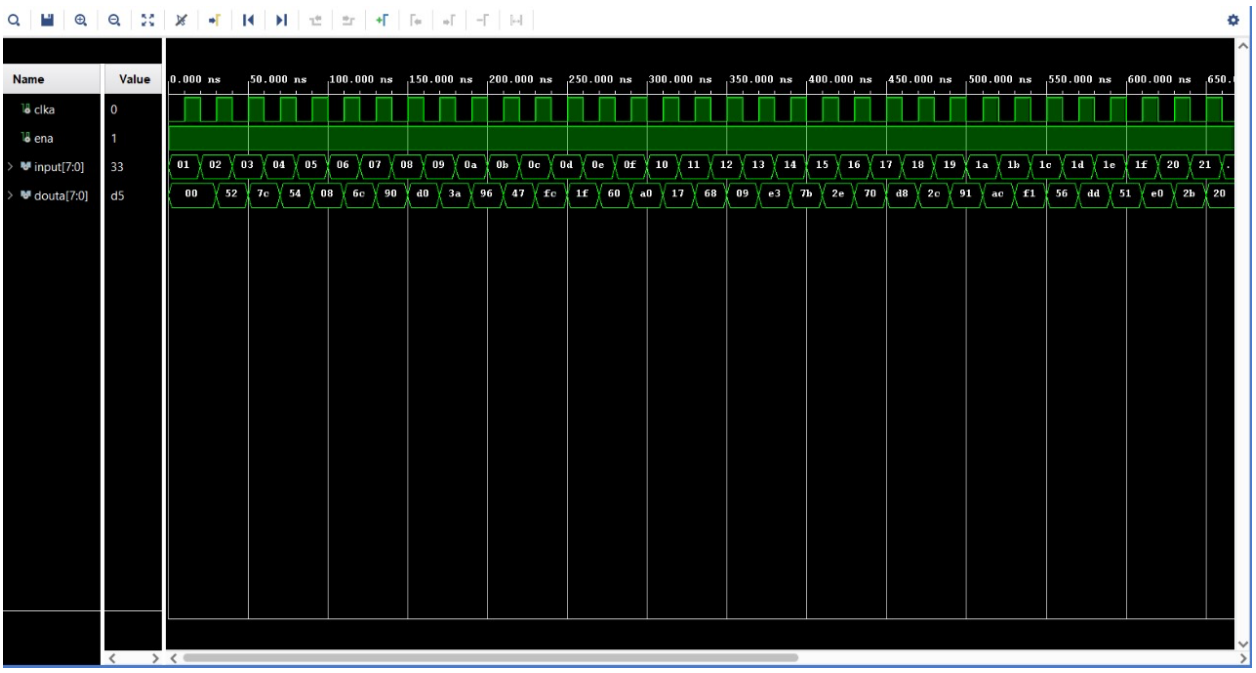
1.4 Simulation

1.4.1 Round Key (XOR)

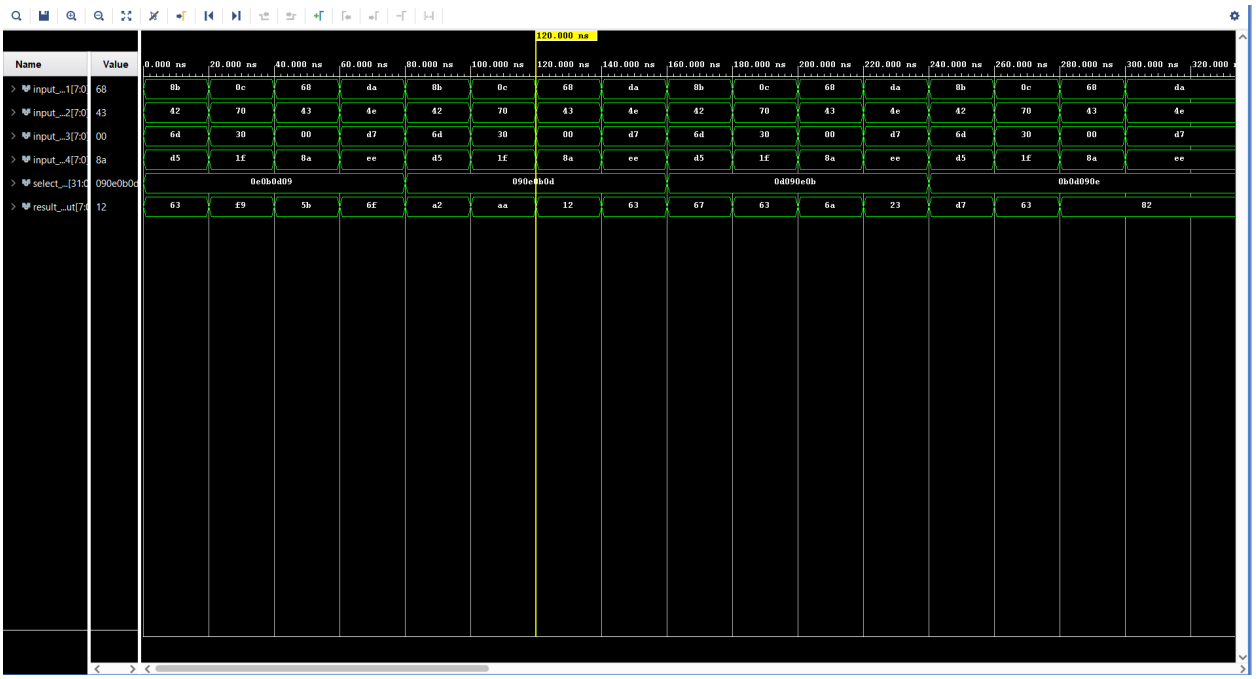


1.4.2 InvShiftRows

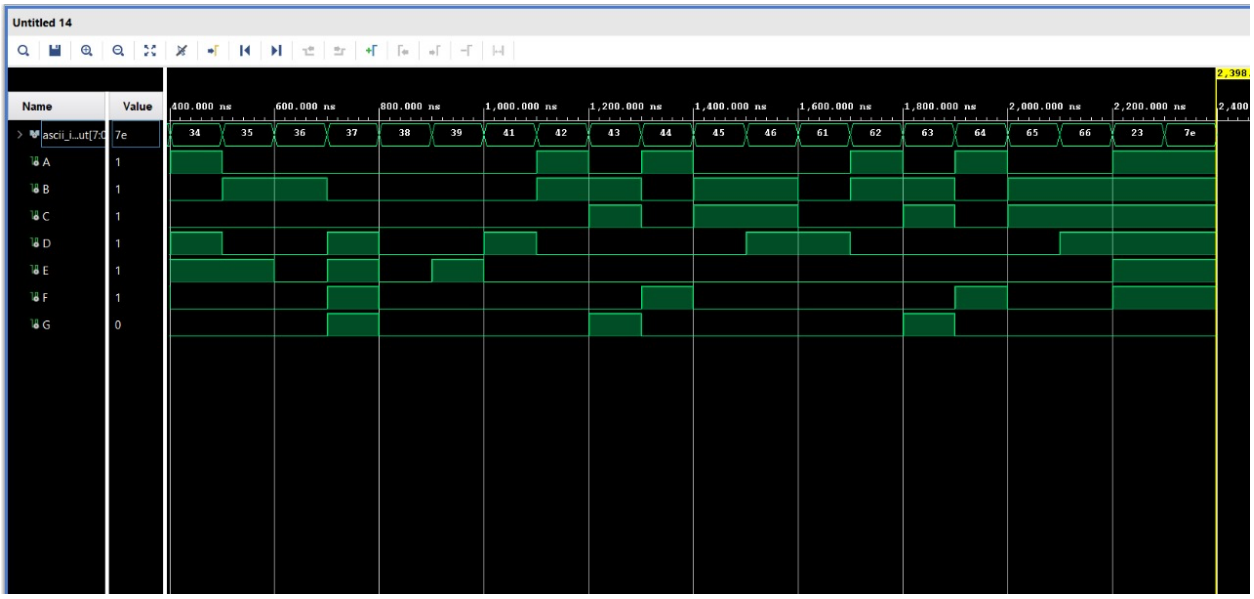
1.4.3 Inv S-Box (outputs are shifted by some Clk cycle because of ROM access delay)



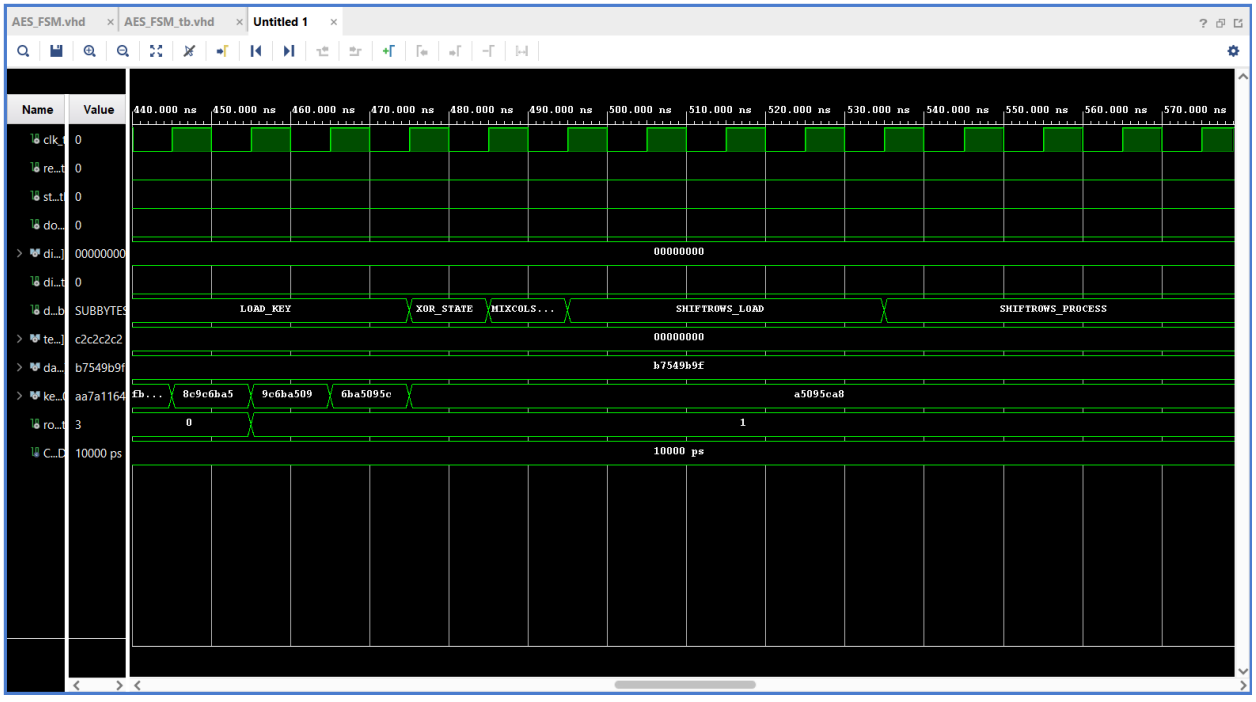
1.4.4 Inv Mix Columns



1.4.5 Display

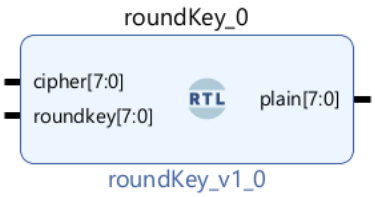


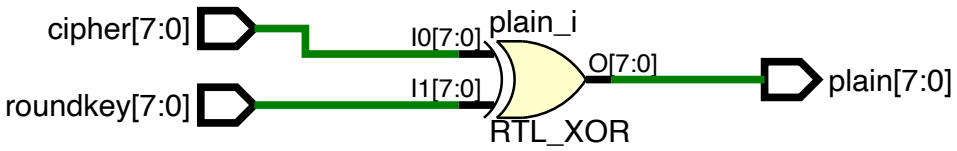
FINAL AES DECRYPT FSM →



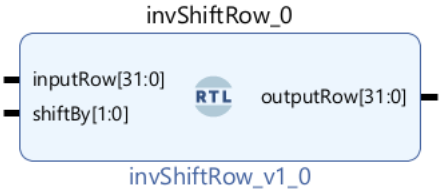
1.5 Block Diagrams

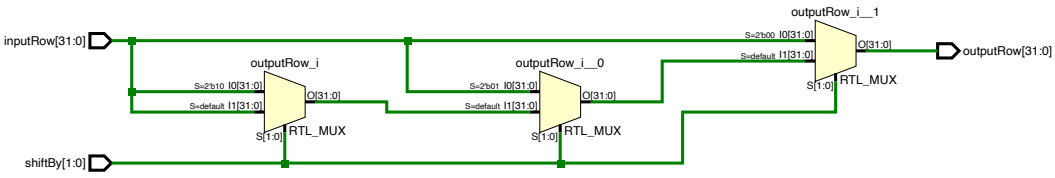
1.5.1 Round Key (XOR)



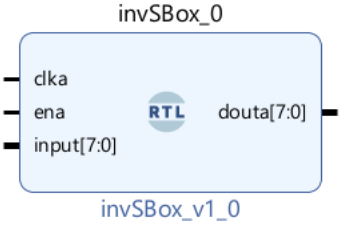


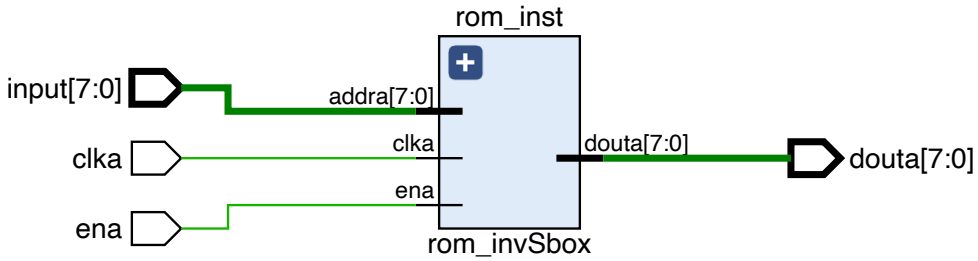
1.5.2 InvShiftRows



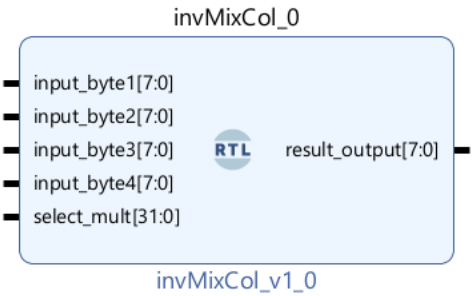


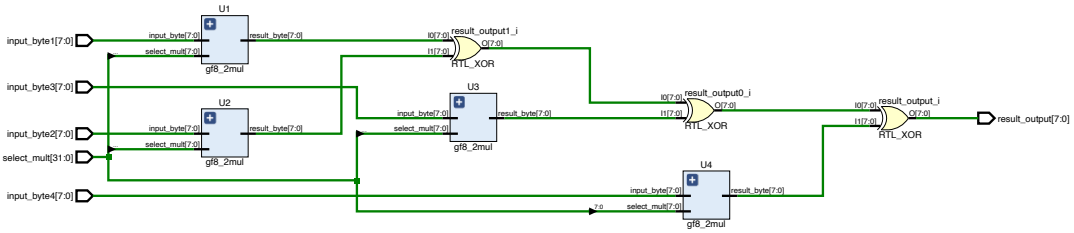
1.5.3 Inv S-Box



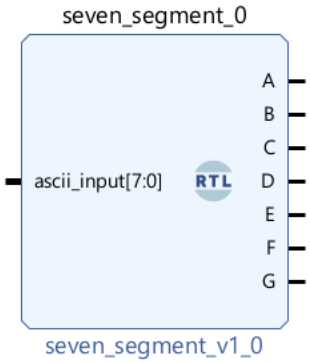


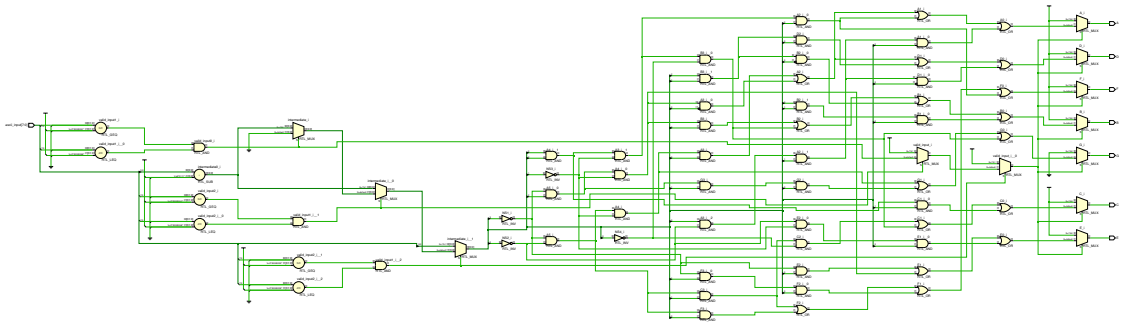
1.5.4 Inv Mix Columns



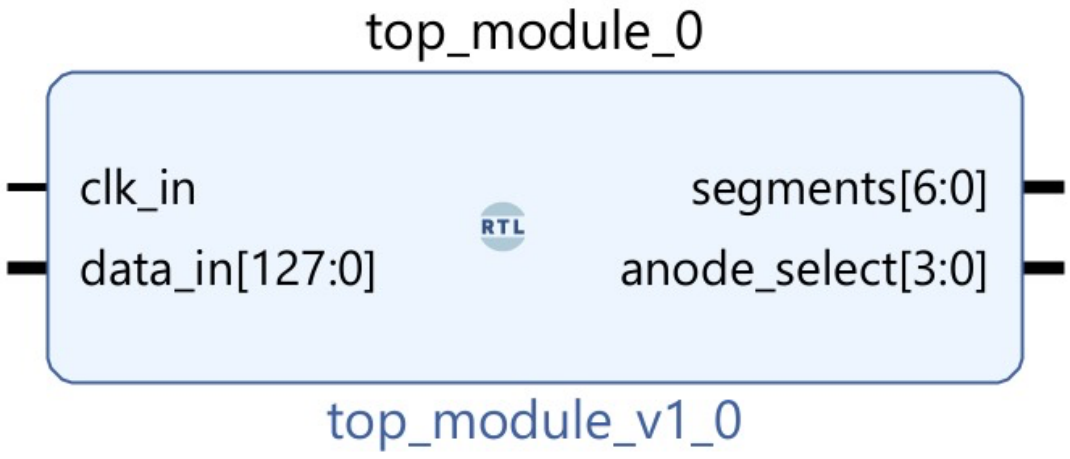


1.5.5 Display

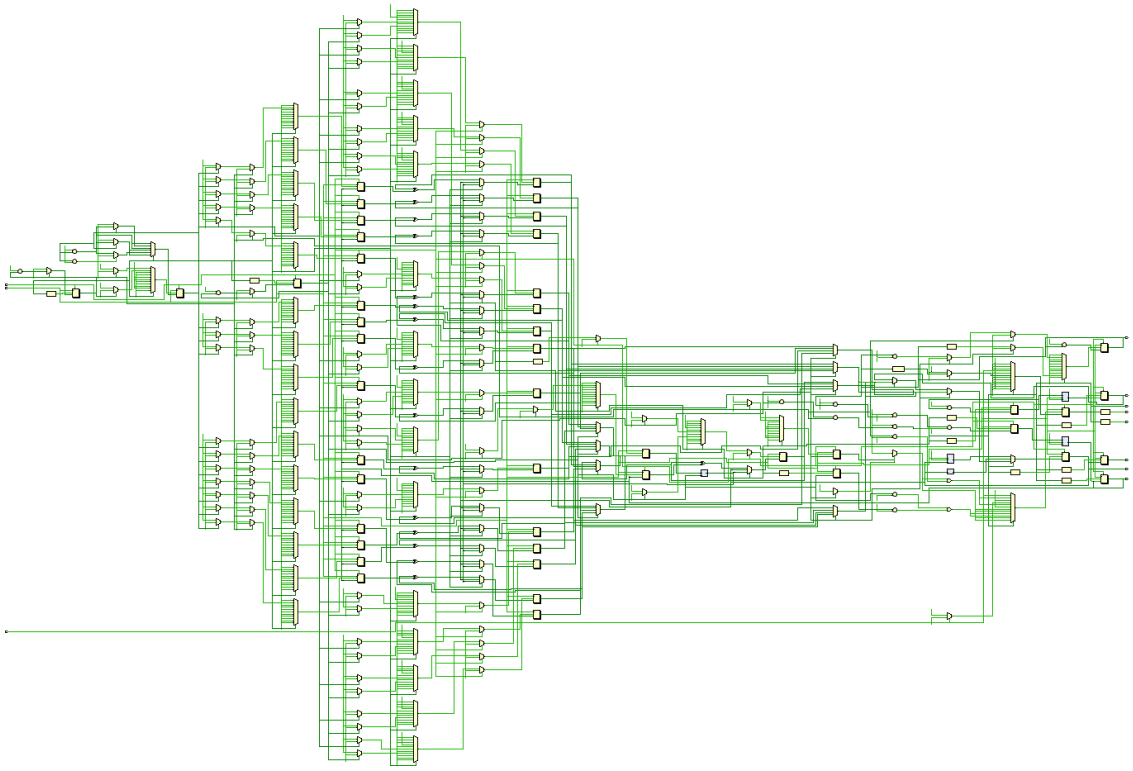




SCROLLING TOP MODULE BLOCK DIAGRAM



AES DECRYPT SCHEMATIC DIAGRAM



1.6 Resource Utilization

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	403	0	0	20800	1.94
LUT as Logic	403	0	0	20800	1.94
LUT as Memory	0	0	0	9600	0.00
Slice Registers	446	0	0	41600	1.07
Register as Flip Flop	446	0	0	41600	1.07
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typ

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
286	Yes	-	Reset
0	Yes	Set	-
160	Yes	Reset	-

```
2. Memory
---------

+-------------------+------+-------+------------+-----------+-------+
|     Site Type     | Used | Fixed | Prohibited | Available | Util% |
+-------------------+------+-------+------------+-----------+-------+
| Block RAM Tile    |    3 |     0 |          0 |        50 |  6.00 |
|   RAMB36/FIFO*    |    0 |     0 |          0 |        50 |  0.00 |
|   RAMB18          |    6 |     0 |          0 |       100 |  6.00 |
|     RAMB18E1 only |    6 |       |            |           |       |
+-------------------+------+-------+------------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate


3. DSP
------

+-----------+------+-------+------------+-----------+-------+
| Site Type | Used | Fixed | Prohibited | Available | Util% |
+-----------+------+-------+------------+-----------+-------+
| DSPs      |    0 |     0 |          0 |        90 |  0.00 |
+-----------+------+-------+------------+-----------+-------+
```

```
4. IO and GT Specific
---------------------

+-----------------------------+------+-------+------------+-----------+--------+
|          Site Type          | Used | Fixed | Prohibited | Available |  Util% |
+-----------------------------+------+-------+------------+-----------+--------+
| Bonded IOB                  |  141 |     0 |          0 |       106 | 133.02 |
| 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 |
+-----------------------------+------+-------+------------+-----------+--------+


5. Clocking
-----------

+------------+------+-------+------------+-----------+-------+
|  Site Type | Used | Fixed | Prohibited | Available | Util% |
+------------+------+-------+------------+-----------+-------+
| BUFGCTRL   |    1 |     0 |          0 |        32 |  3.13 |
| 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 |
+------------+------+-------+------------+-----------+-------+
```

6. 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

7. Primitives

Ref Name	Used	Functional Category
LUT6	309	LUT
FDCE	286	Flop & Latch
FDRE	160	Flop & Latch
OBUF	138	IO
LUT5	64	LUT
LUT2	25	LUT
LUT4	18	LUT
LUT3	14	LUT
RAMB18E1	6	Block Memory
LUT1	6	LUT
IBUF	3	IO
BUFG	1	Clock