EE671: Course Porject 2
Group Number: 3

Devavrat Patni : 22B3969
Raghav Sapre : 22B1241
Samar Perwez : 22B3913
Tanay Bhat: 22B3303

November 2024

Contents

Chapter 1

Verilog Design

1.1 Overview of AES-128

The AES-128 encryption algorithm operates on a 128-bit plaintext block and utilizes a 128-bit key to produce a 128-bit ciphertext. It is built upon the Galois Field $GF(2^8)$, which is represented by the polynomial $x^8 + x^4 + x^3 + x + 1$. According to this, addition is defined as XOR and multiplication is defined as multiplication in the field $GF(2^8)$ followed by modulo $x^8 + x^4 + x^3 + x + 1$. The encryption process involves two primary components:

Key Expansion

- The 128-bit key provided as input is expanded into 11 separate 128-bit round keys using a key schedule algorithm.
- The first round key is the original key, while the subsequent 10 round keys are derived through a series of transformations including substitutions, rotations, and XOR operations with a round constant.

Data Transformation

The 128-bit plaintext block undergoes a series of 40 well-defined cryptographic operations distributed across 10 main rounds, plus an initial and a final step:

- **Initial AddRoundKey:** The plaintext block is XORed with the initial round key.
- **Main Rounds (1 to 9):** Each round consists of four operations:
  - **SubBytes:** A non-linear substitution step where each byte is replaced by a value from a fixed S-box.
  - **ShiftRows:** A permutation step where rows of the block are cyclically shifted.

– **MixColumns:** A mixing operation that transforms the columns of the block.
– **AddRoundKey:** The block is XORed with the round key for the current round.

- **Final Round:** The final round is similar to the main rounds but excludes the *MixColumns* operation.

1.2 Architectural Design

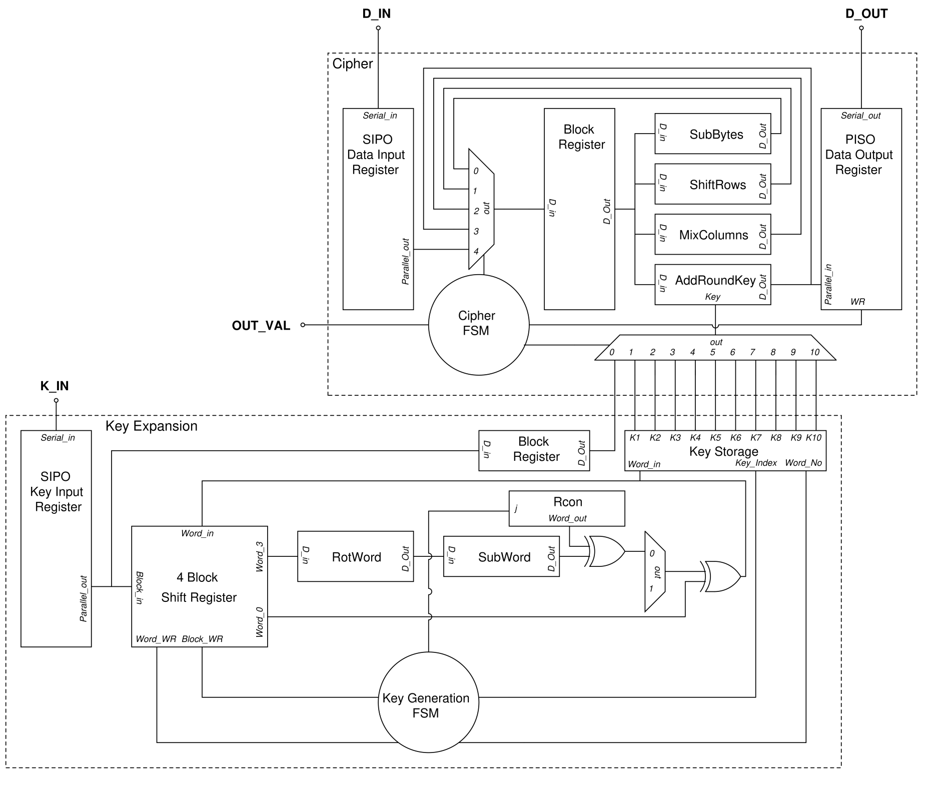

Figure 1.1: Architectural View

The datapath for implementing AES-128 encryption is designed to efficiently handle the encryption process by breaking it into two major modules:

Cipher Module

- This module is responsible for executing the core AES encryption operations on the data block.
- It performs the following steps:
  - **SubBytes:** A non-linear byte substitution using an S-box.
  - **ShiftRows:** A row-wise cyclic shift to introduce diffusion.
  - **MixColumns:** A column-wise linear transformation for mixing data (skipped in the final round).
  - **AddRoundKey:** Combines the current state with the round key using XOR.
- The cipher operates iteratively over 10 rounds, with the final round excluding the *MixColumns* step.

Key Expansion Module

- This module generates the 11 round keys required for the encryption process from the initial 128-bit input key.
- The key schedule algorithm involves byte substitutions, rotations, and XOR operations with a round constant to ensure each round key is unique and secure.

Serial Input and Output

- **Input (Serial):**
  - The plaintext block and the initial key are provided serially.
  - A *Serial-In Parallel-Out (SIPO)* register is used to convert the serial input into a 128-bit parallel format for processing.
- **Output (Serial):**
  - The encrypted ciphertext is produced serially through the output signal **DOUT**, with the Least Significant Bit (LSB) transmitted first.
  - An output signal **OUT_VAL** is asserted high to indicate when the output data is valid.
  - A *Parallel-In Serial-Out (PISO)* register is used to convert the 128-bit parallel output into the serial format for transmission or storage.

Control Mechanism

- The entire datapath is controlled by 2 Finite State Machines for *Cipher* and *KeyExpansion*:
  - The FSM orchestrates the sequence of operations for both the Cipher and Key Expansion modules.
  - It ensures that:
    - Round keys are generated and applied at the correct steps.
    - The required operations (*SubBytes*, *ShiftRows*, etc.) are executed in the proper sequence.
    - Data movement between modules (via SIPO and PISO registers) is synchronized.

Key Features of the Design

- **Efficiency:** The serial input/output design minimizes I/O pin usage, making it suitable for hardware-constrained environments.
- **Modularity:** Separation into Cipher and Key Expansion modules simplifies design and debugging.

Chapter 2

Table of Performance & Final Magic Screenshot

The performance parameters extracted from the GRT_STA file are as follows,

Parameter	**Value**
Area	392,264 μm^2
Total Power	105nW
Clock Frequency	62.5 MHz

Table 2.1: Table of Performance

The final screenshot of the magic layout is as follows:

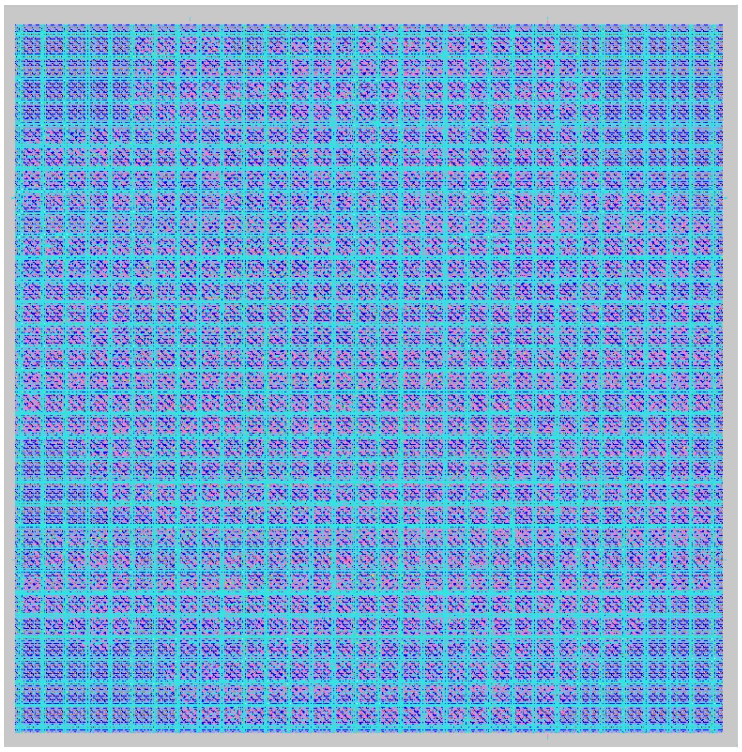

Figure 2.1: Final Layout

Chapter 3

Testbench Waveforms

We used the following key and plaintext for testing the AES-128 encryption system:

- **Key**: `0x2b7e151628aed2a6abf7158809cf4f3c`
- **Plaintext**: `0x3243f6a8885a308d313198a2e0370734`
- **Ciphertext**: `0x3925841d02dc09fbdc118597196a0b32`

The following waveforms were obtained for the above key and plaintext post synthesis:

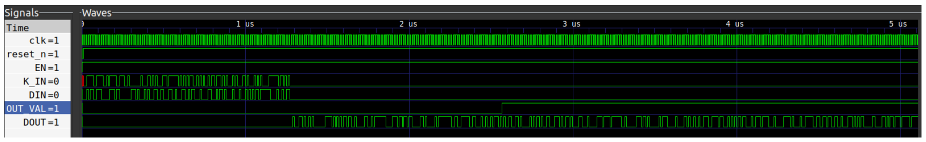

Figure 3.1: Post Synthesis Waveform

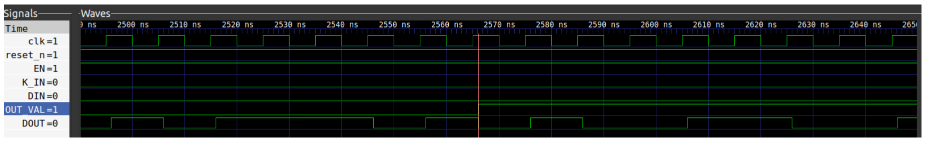

Figure 3.2: Post Synthesis Waveform (zoomed)

The output can be observed when OUT_VAL is asserted and is serially produced (LSB first).

The post-routing waveforms are as follows,

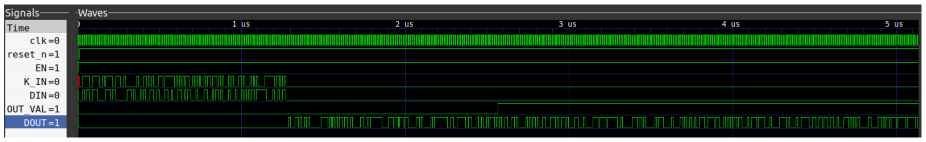

Figure 3.3: Post Synthesis Waveform

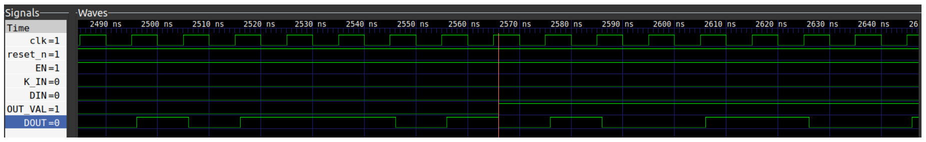

Figure 3.4: Post Synthesis Waveform (zoomed)

Chapter 4

Work Distribution

The work was distributed among the team members as follows:

- Devavrat Patni : Helped in all the steps including and after synthesis and in debugging errors caught post synthesis
- Raghav Sapre : Worked on KeyGeneration module including datapath and FSM.
- Samar Perwez : Worked on Cipher module including datapath and FSM.
- Tanay Bhat : Worked on KeyGeneration module including datapath and FSM. Also explored possibility of decryption by inverting all the blocks.