VHDL project: AES encyption

1 Introdution

FPGAs are often used in cryptographic applications. Indeed, if designed carefully, FPGA-based implementations are very efficient and can outperform software-based implementations by several factors of speed.
Nowadays, one the most used specification for the encryption of electronic data is the Advanced Encryption Standard (AES) which was invented by Joan Daemen and Vincent Rijmen in 2000. The AES algorithm is a symmetric-key cryptography algorithm that can encrypt and decrypt data. As shown in Figure 1, encryption converts plain-text to an unintelligible form called cipher-text, while decryption converts this cipher-text back to the original plain-text.

The goal of this project if to implement the **encryption part** of the 128-bit AES algorithm on the Basys 3 FPGA Board.

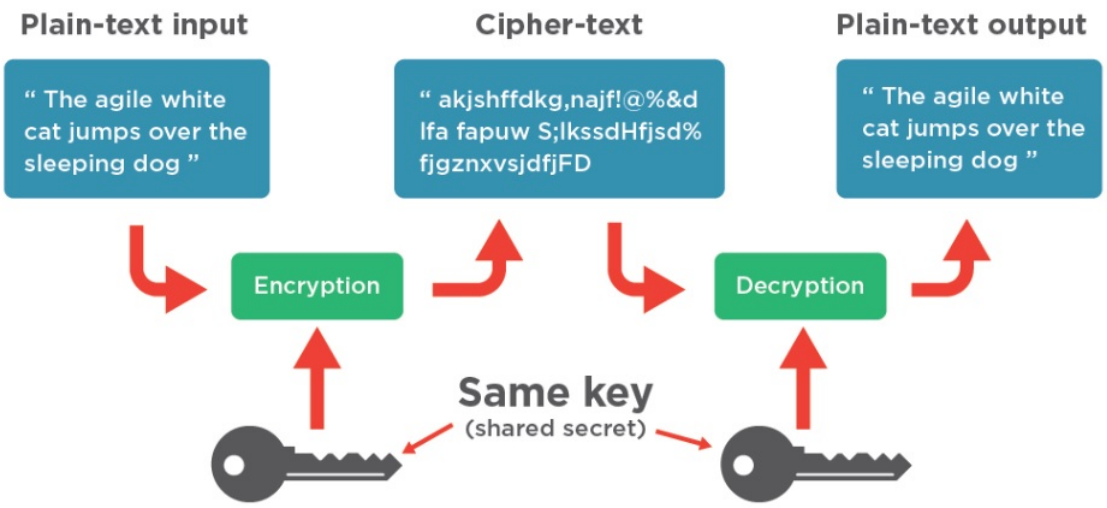

Figure 1: Symmetric-key cryptography

2 The 128-bit AES algorithm and implementation

The 128-bit AES algorithm encrypts data by blocks of 128 bits using a key of 128 bits. The AES architectural flow is given in Figure 2. To convert the Plain text into Cipher text, an initial round followed by 10 rounds of encryption are needed to progressively encrypt the plain text. These rounds are all implemented by using one of several steps among the following: `AddRoundKey`, `SubBytes`, `ShiftRows` and `MixColumns`. Theses 4 steps are explained in the following subsections.

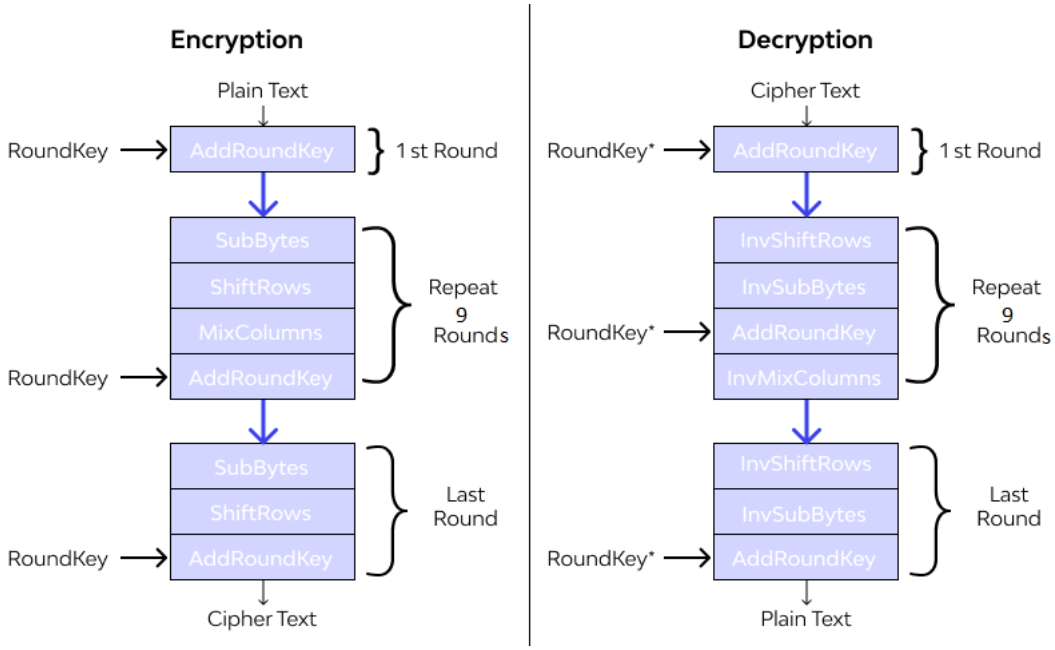

Figure 2: AES architectural flow

2.1 AddRoundKey

In AES, the plain text and the round key are organised as 4x4 bytes matrices such as shown in Figure 3.

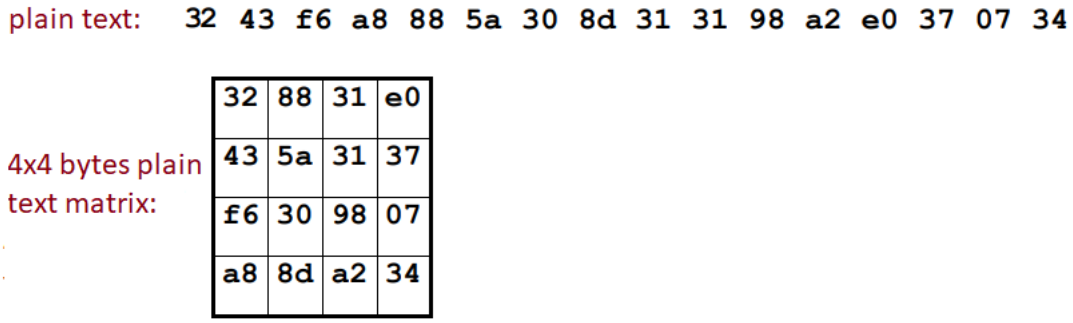

Figure 3: Example of plain text organised as a 4x4 bytes matrix

In the `AddRoundKey` step, the 128-bits block a — obtained at the previous step — is bit-wise Xored with the `RoundKey` k, as shown in Figure 4. Note that the `RoundKey` k is different in each round. To obtain the different round keys, a key schedule procedure is used. This key expansion procedure is explained here. You don't need to implement this key schedule procedure yourself, the `RoundKeys` for each round are provided in the *RoundKeys.txt* file.

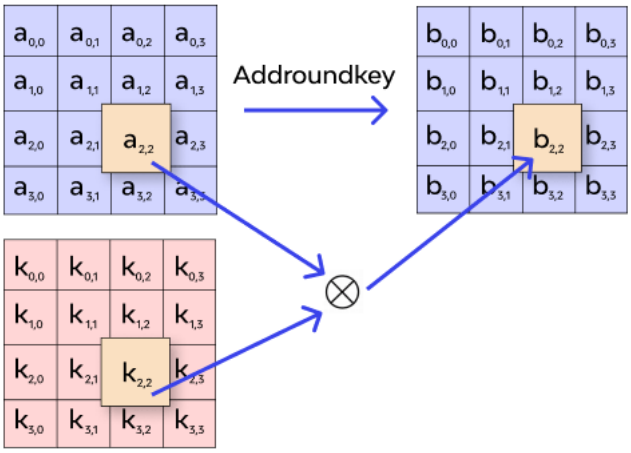

Figure 4: AddRoundKey step

2.2 SubBytes

In the `SubBytes` step, each byte is substituted using a substitution table called the S-box. As shown in Figure 5. The S-box is provided to you as a lookup table in the file *S_box.vhd*.

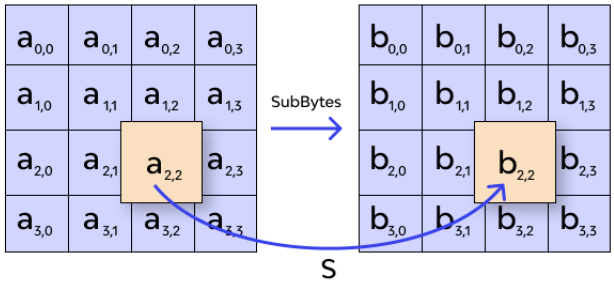

Figure 5: SubBytes step

2.3 ShiftRowns

In this step, the bytes in the last three rows are cyclically shifted to the left over a number of bytes equal to the row number, as seen in Figure 6. The first row is not shifted.

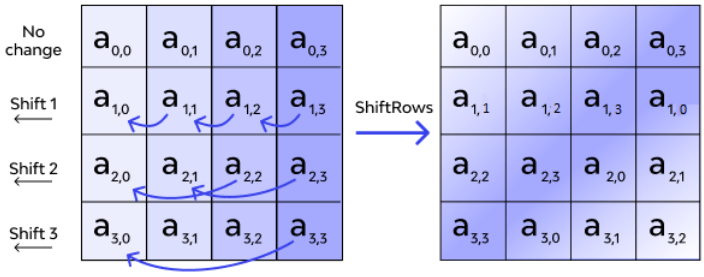

Figure 6: ShiftRows step

2.4 MixColumns

This transformation operates column-by-column. During this step, each column is multiplied by a constant matrix $c(x)$, as presented in Figure 7. The different multiplications done in this step are Galois field multiplications (see this page for more information).

$$c(x) = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

How to implement the MixColumn step and the Galois field multiplication?
As an example, let's see how to determine $b_{0,0}$, the MSB byte of the text to encrypt after the `MixColumn` step.

1. The `MixColumn` step operates on the first column as follows:

$$\begin{bmatrix} b_{0,0} \\ b_{1,0} \\ b_{2,0} \\ b_{3,0} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \end{bmatrix} \tag{1}$$

2. The first element is then given by the following expression:

$$b_{0,0} = (2.a_{0,0}) \oplus (3.a_{1,0}) \oplus (1.a_{2,0}) \oplus (1.a_{3,0}) \tag{2}$$

   where $\oplus$ is the bit-wise XOR logical operator.

3. $(2.a_{0,0})$ and $(3.a_{1,0})$ can be calculated using the LUTs respectively given in the *LUT_mul2.vhd* and *LUT_mul3.vhd* files.

All the other bytes are calculated in a similar way.

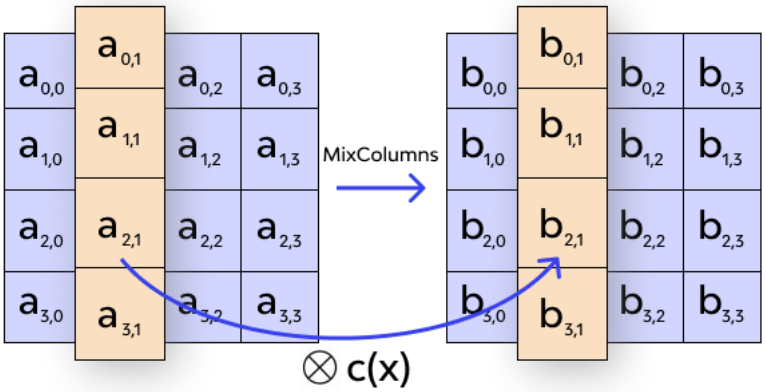

Figure 7: MixColumns step

3 Test-vectors

To make sure that your implementation is correct, you should use the test vectors given in the following file, which is provided by the National Institute of Standards and Technology. This file gives the results you should obtain for each step of the flow of the 128-bit AES, for 4 different 128-bits plain texts and a 128 cipher key.

4 Project Specifications

During this project, you should create a VHDL project implementing the AES encryption algorithm according to the following specifications:

- Write each of the 4 AES encryption steps in a separate VHDL module. For each module, test your implementation by writing a test-bench using the provided test-vectors.
- Using all the 4 steps modules, implement the AES encryption **such that one AES step is executed in one clock cycle.** Write a test-bench to test your implementation using the test-vectors provided to you.
- Write a module such that the encryption of one of the given test-vectors starts when pressing the central button. Once the encryption is finished, "AES" should be displayed on the 7-segment display. To be able to start the encryption again, the right-button should be pressed. This RST should be synchronous. Write a test bench to test your implementation.
- You are free to add any additional functionalities.

You must also write a project report which contains the following elements:

- A block diagram of how your different modules are interconnected with each other.
- A screen capture of the waveforms obtained for each test-bench. Each of them must be briefly commented.
- Explain any additional functionalities that you have implemented.

The project deadline is the **22/12/2023 at 23:59**. The project submission is done by email to oscar.van.slijpe@ulb.be, muhammad.ali@ulb.be. The object of your email should be "ELECH409 AES Project". You must attach two different files to your email:

1. Your project report in pdf format named as "lastname1_lastname2_AES.pdf"
2. Your Vivado AES project as a zip file. We ask you to archive your project directly from Vivado. Open your project then select File → Project → Archive. Name your achive as "lastname1_lastname2_AES" and only select "Include configuration settings".