ECOLE POLYTECHNIQUE DE BRUXELLES

ELEC-H409

Université Libre de Bruxelles

École Polytechnique de Bruxelles

VHDL projet: AES encryption

Authors of Group C:
JANKE Nico (540076)
WOJTACH Kacper (513025)

Professors:
MILOJEVIC Dragomir
STEFANIDIS Lucas
LADNER Navid

January 14, 2025

Contents

1 Diagram

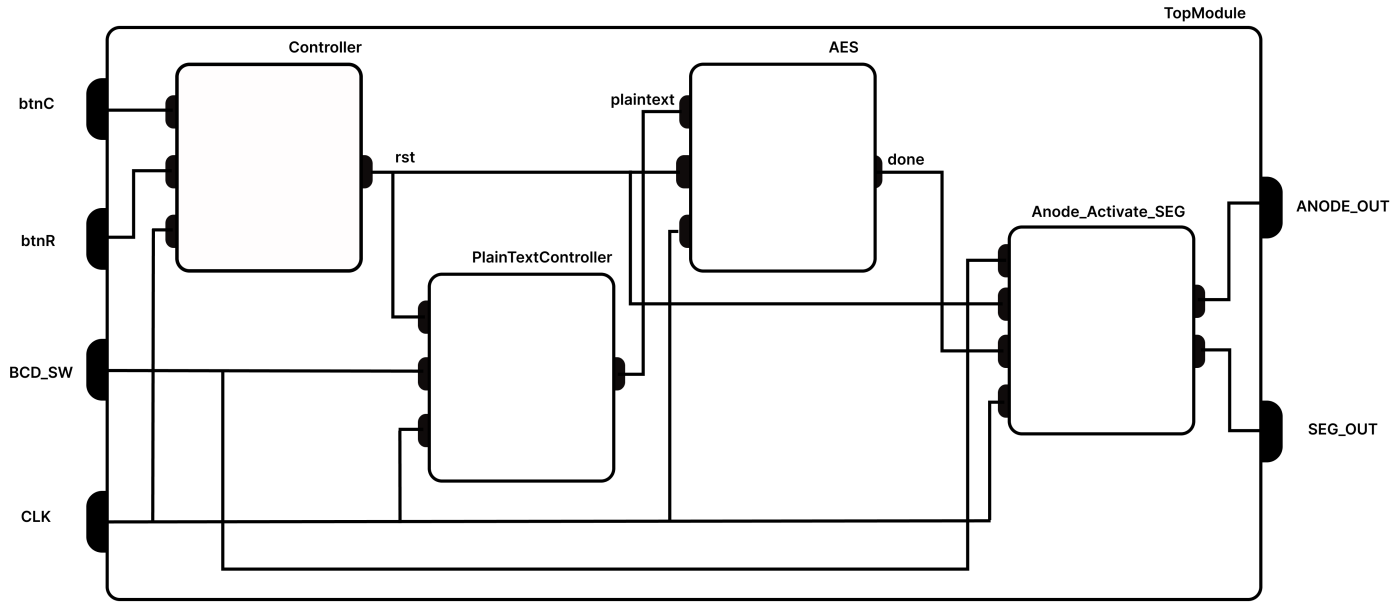

Figure 1: Top Module Diagram

Our TopModule consists of 4 inputs and 2 outputs.

- **btnC & btnR**: Buttons of the Basys3 used to start end reset AES as explained in subsection 2.1.
- **BCD_SW**: Physical switches of the Basys3 used to select the plain text to encrypt as explained in subsection 2.2.
- **CLK:** Clock of the Basys3 used for synchronizing all the sub modules together and important for counting the rounds of AES.
- **ANODE_OUT & SEG_OUT**: Output pins of the Basys3 used to control the seven segment display.

The different sub modules are explained below alongside with the waveforms of their test-benches.

2 Controllers

2.1 Controller

The Controller module has three inputs: Clk, btnC, and btnR, and a single output, rst. Initially, the module sets rst to 1. When btnC is pressed and released, rst changes to 0 on the next clock cycle. In contrast, pressing and releasing btnR sets rst to 1 in the following clock cycle.
The waveforms obtained by the test-bench of this module can be seen in figure 2.

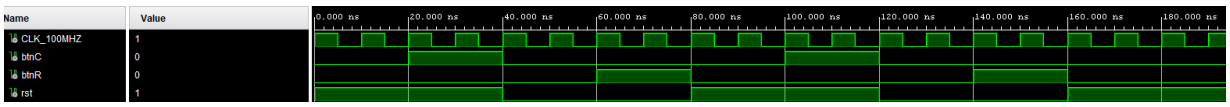

Figure 2: Controller Test Bench

2.2 Switches

Instead of pressing the central button to initiate AES encryption with the same plain text every time, we introduced a more flexible option that allows the user to select one of four different plain texts.
When the rst signal is set to 1, the user can choose the desired plain text by using the two right-most switches, and the corresponding number, ranging from 0 to 3, will be displayed on the screen. Additionally, we used the seven segment display to show the nummber of the selected plain text (0 to 3).

2.3 Plain-text Controller

The plain-text Controller takes as input the two switches (*BCD_SW*), *CLK* and the *rst* output of the **Controller** and gives as output a plain text based on the number formed by the switches. The plain text is selected using a LUT: **plain texts**.
The waveforms obtained by the test-bench of this module can be seen in figure 3.

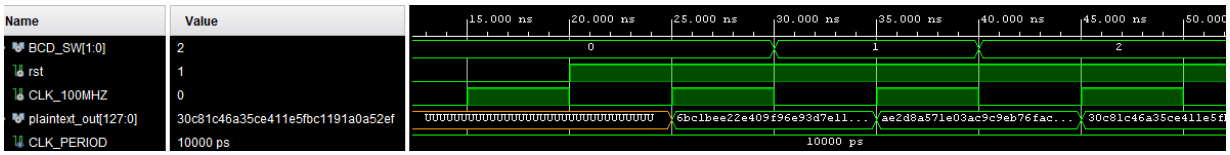

Figure 3: plain textController Test Bench

3 AES

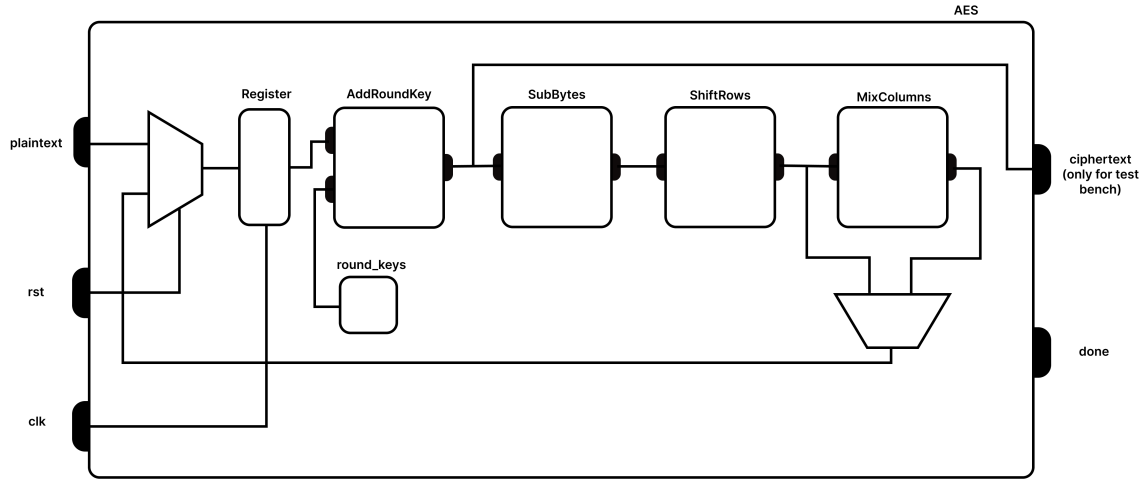

Figure 4: AES Diagram

Our AES module takes as input the plain text, a reset and a clock signal. *rst* is the selector for the entry multiplexer. If rst is active the plaintext is stored in the register, otherwise the output of **ShiftRows** or **MixColums** is selected based on the round count.
The rounds are counted by the module using the clock once the *rst* signal is off. In one clock cycle one round of AES takes place and the result is stored in the register at the start of the second round. Before the last round the module changes the selector in the second multiplexer to select the output of **ShiftRows** because in the last round **MixColums** needs to be skipped. The final cipher-text is the output of **AddRoundKey** after the 10 rounds and at that moment the done signal is set to 1.

3.1 AddRoundKey

The AddRoundKey module takes a 128-bit sequence and a key as input and outputs the result of the XOR operation between the two. The test bench was conducted using the first plain text and the key from round 0 (figure 5).

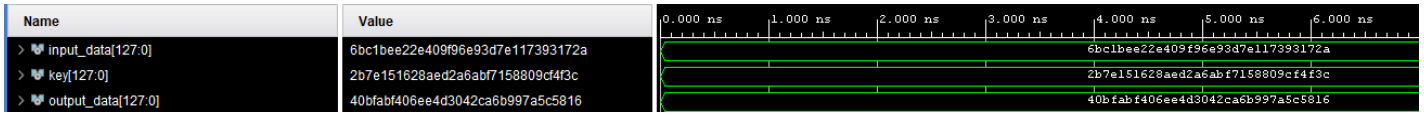

Figure 5: AddRoundKey Test Bench

3.2 SubBytes

The SubByte module takes a 128-bit sequence as input and outputs another 128-bit sequence. The test bench was conducted using the output of the AddRoundKey operation from round 0. After one clock cycle we obtain the wanted output (figure 6).

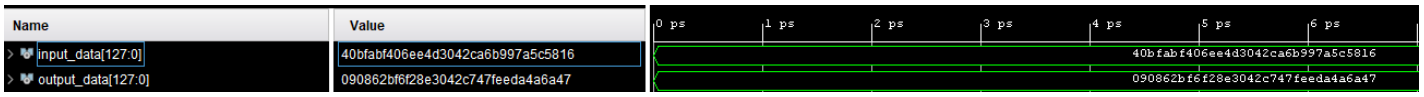

Figure 6: SubBytes Test Bench

3.3 ShiftRows

The ShiftRows module takes a 128-bit sequence as input and outputs another 128-bit sequence. The test bench was conducted using the output of the SubBytes operation from round 1. After one clock cycle we obtain the wanted output (figure 7).

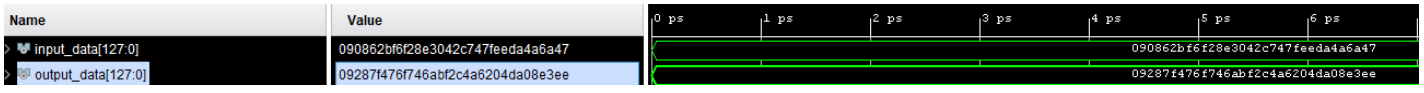

Figure 7: ShiftRows Test Bench

3.4 MixColumns

The MixColumns module takes a 128-bit sequence as input and outputs another 128-bit sequence. The test bench was conducted using the output of the ShiftRows operation from round 1. After one clock cycle we obtain the wanted output (figure 8).

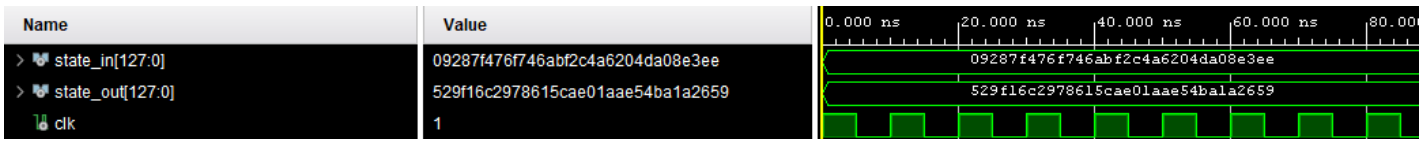

Figure 8: MixColumns Test Bench

3.5 Multiplexer

The Multiplexer module takes two 128-bit sequences as input and outputs one 128-bit sequence based on the select signal. The test bench was conducted using the feedback and plain text inputs, with the SEL signal controlling the selection between them. When SEL is high, the feedback input is passed to the output (Q), and when SEL is low, the plain text input is selected. The expected result was achieved, with the output reflecting the correct input based on the state of SEL at each moment. The waveforms obtained by the test-bench of this module can be seen in figure 9.

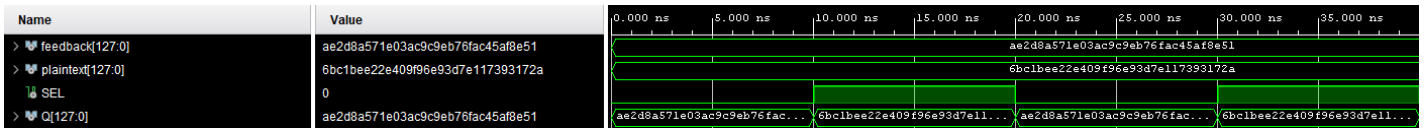

Figure 9: Multiplexer Test Bench

3.6 Register

The Register module takes a 128-bit sequence as input and outputs the same 128-bit sequence. The test bench was conducted by applying a clock signal and feeding a 128-bit value to the data input (D). At each rising edge of the clock (Clk), the value of D is captured and passed to the output (Q). The expected result was achieved, with the register correctly storing and outputting the value at each clock cycle. When D changes, the corresponding value is reflected in Q at the next clock edge, confirming that the register is correctly capturing and storing the input data. The waveforms obtained by the test-bench of this module can be seen in figure 10.

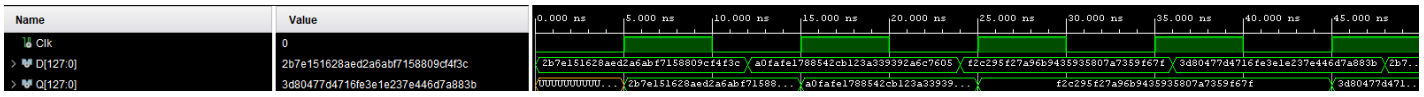

Figure 10: Register Test Bench

3.7 AES

The AES module links all its component as detailed in figure 4. It is responsible for counting the rounds, choosing the right key and putting the right value in the selectors of both multiplexers based on the reset signal and the round count.
The key given to **AddRoundKey** is retrieved using a LUT: **RoundKeys**.
Once the encryption is done, the cipher-text is set as the output of **AddRoundkey** and the output *done* is set to 1.
For testing purposes we did a test-bench with more outputs to show the signals of every component of AES. For the final product we only have one *done* signal output.

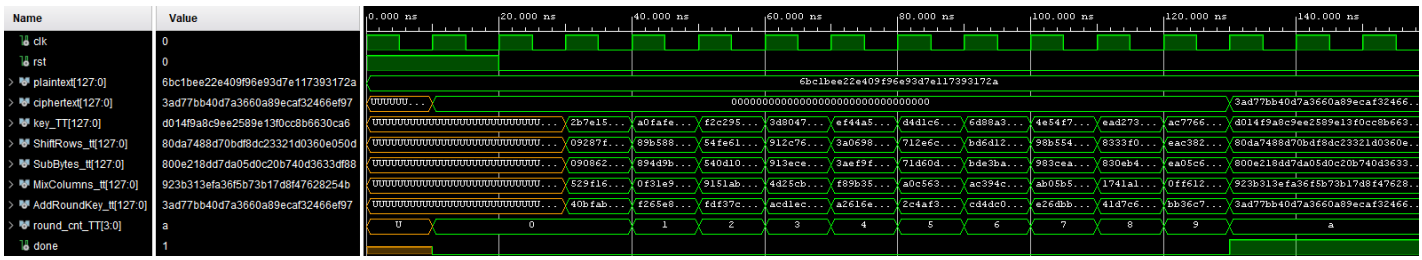

Figure 11: AES Test Bench

4 Anode_Activate_SEG

The Anode_Activate_SEG module is used to control the seven-segment display on the Basys3 board. It ensures that the correct numbers or symbols are shown on the display based on input signals. This module is important because it provides visual feedback to the user about the system's status, such as the selected plain text or whether the encryption process is finished.

The module works by using a 100 MHz clock signal to create a refresh counter. The counter cycles through four values, which are used to control which of the four seven-segment displays is active at any given time. This process makes it seem like all four displays are working at the same time, even though only one is active at a time.

The output signals of the module are:

- anode_out, which determines which of the four segments is active.
- seg_out, which controls the pattern of lights on the active segment to show a specific number or symbol.

The module's behavior depends on the input signals:

- If the rst signal (reset) is active, the display shows a number based on the value of the BCD_SW input. For example:
  - If BCD_SW = "00", the display shows "0"
  - If BCD_SW = "01", the display shows "1".
  - If BCD_SW = "10", the display shows "2".
  - If BCD__SW = "11", the display shows "3".
- If the done_in signal is active, AES is displayed on the segments to show that the encryption process is finished.
- If neither rst nor done_in is active, the display turns off (seg_out = "1111111").

5 TopModule

The TopModule is responsible for linking all the modules together and its inputs and outputs are linked to the Basys3 device. It can therefore not really be tested with more outputs using a test-bench.
For testing purposes we impemented a TopModule with outputs for all the signals of AES and more to visualize what happens when me press the buttons. This can be seen in figure 12.

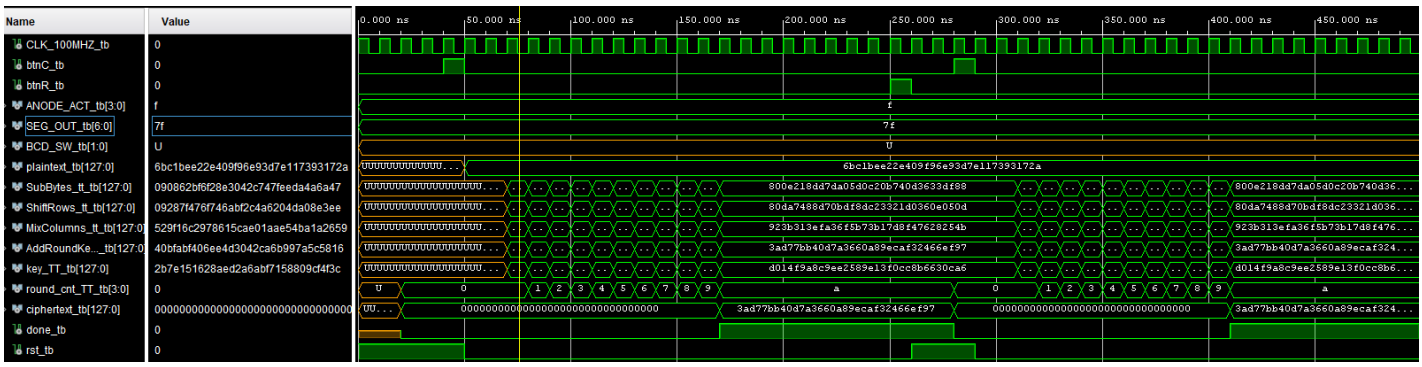

Figure 12: TopModule Test Bench

At last we programmed our FPGA and got the expected result (figures 13, 14).

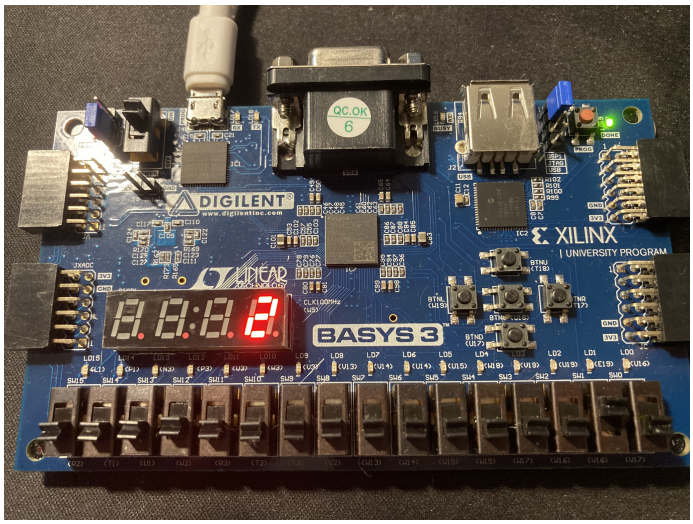

Figure 13: Plain text selection with the two right most switches

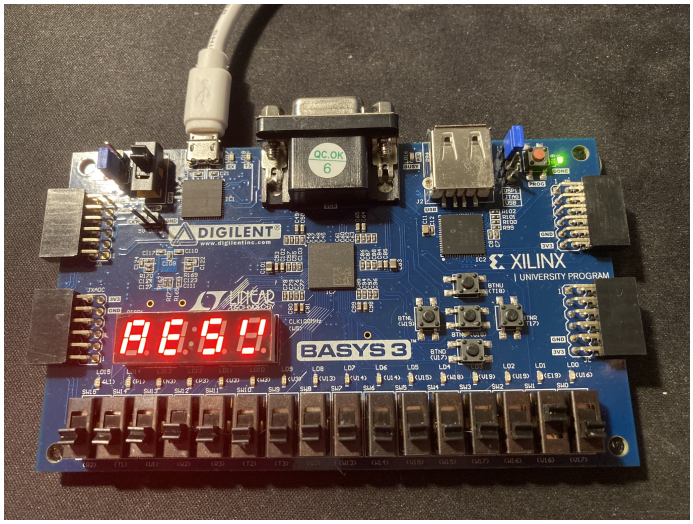

Figure 14: AES on the seven segment display once AES is done