ASSIGNMENT 2
CS666: HARDWARE SECURITY OF INTERNET OF THINGS

Group G2
Aug, 2023

1. Group Members

1. VIKRANT CHAUHAN : MS CYBERSERCURITY : 231110407
2. SOUVIK MUKHERJEE : MS CYBERSECURITY : 231110405
3. VISHAL KUMAR : MTECH CYBERSECURITY : 231110058
4. VALETI LOKESH : MS CYBERSECURITY : 231110406
5. M DHILIPKUMAR : MTECH CYBERSECURITY : 231110026

2. Helping instruction for running the code files

- Kindly do ***cd*** to the specific folder, and give the required inputs of Plaintext and Key in our testbenchfile. **The name of our testbench file is "aes_test.v"**

- Terminal command 1 : ***iverilog -o aes.vvp aes_test.v***
- Terminal command 2 : ***vvp aes.vvp***
- See Output
- You can also see the ***GTKwave*** output by using ***"gtkwave aes.vcd"***

3. Question

Design a Verilog module for AES-128 iterative architecture. The design should have two inputs: plaintext (128-bit), key (128-bit), and one output: ciphertext (128-bit). The key expansion utilized to generate the round keys using the master key and the AES round operation should run parallelly in a non-blocking manner. You can use the instantiation of given modules (subbyte, shiftrow, mixcolumn, sbox and key expansion) to design a 10-round iterative AES hardware. [Hint: Each round of AES would be executed in one clock cycle.]

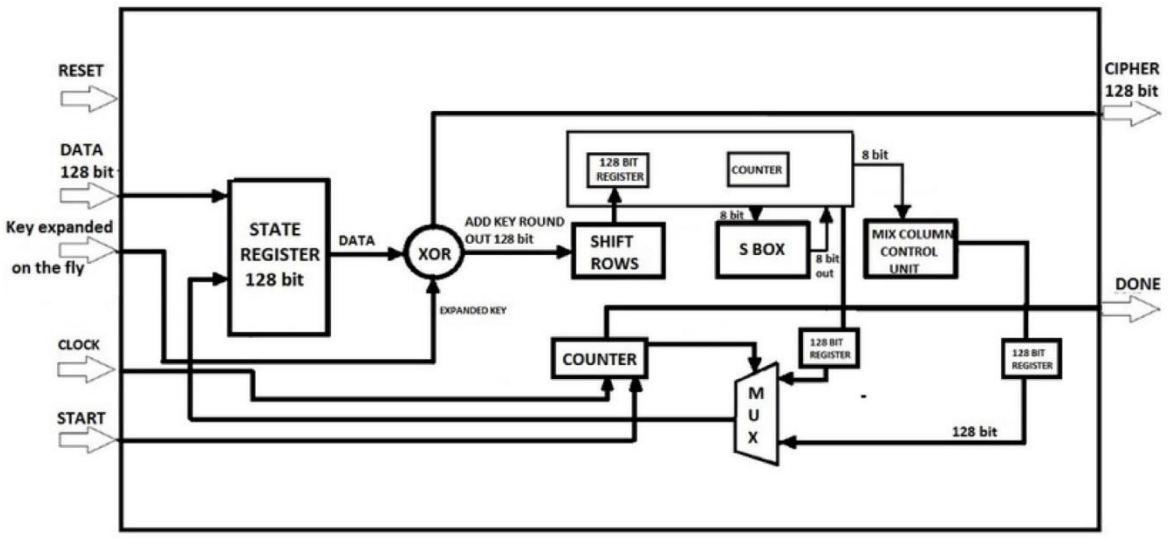

Figure 1: Iterative AES hardware

4. Introduction

AES stands for Advanced Encryption Standard.In AES there are various versions of AES 10 round AES, 12 Round AES, 14 round AES.In 10 round AES we make use of 128 bit key, in 12 round AES we make use of 192 bit key, in 14 rounds AES we make use of 256 bit key.

We shall discuss about 10 round AES which is having 128 bit key. In AES we having mainly 4 parts in each round namely:-

1. **Subbytes**: In Subbytes we make use of sbox.Sbox stands for substitution box.In subbytes what we do is we make use of a lookup table. In subbyte for each byte of data we replace that data with some other byte.This byte is seen from the from the lookup table.

2. **ShiftRows** : In Shift Rows we left shift the first row of the state matrix by 0 positions,next we left shift the second row of the state matrix by 1 positions,next we left shift the third row of the state matrix by 2 positions,next we left shift the fourth row of the state matrix by 3 positions.

3. **MixColumns** : In mixcolumn we take each column of the state matrix and mutiply that by a standard matrix. This matrix is given as:

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

4. **Add Round Key:** In add round key we basically perform an XOR operation with our state matrix and an output key which is generated through the key generator which generates the keys for all the 10 round of AES and also one extra key.

5. Working Of AES

In AES we firstly pass our master key to the key generator. This key generator generates 11 keys, 10 keys for 10 rounds of AES and one key for the initial round (Round 1).

Initially we firstly perform an XOR operation with the add round key of the 0th round of the AES and our initial State matrix, that is the plain text.

The State matrix obtained after the first round is passed to the First round of AES. In First to ninth round of AES we firstly perform Subbyte operation, ShiftRows operation, Mixcolumn operation and Add round Key operation for each round of the AES till the ninth round.

For the tenth round of AES we firstly perform subbyte operation, shift row operation and then the add round key operation we don't take the mix column operation in the tenth round. After the tenth round we finally get our final ciphertext and our AES operation has been completed.

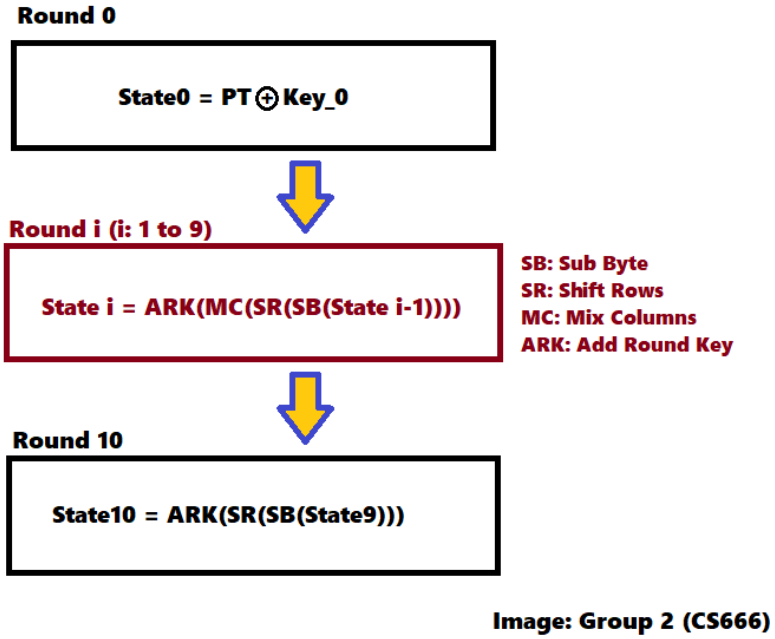

Figure 2: AES Flow Diagram

6. Implementation Of AES

For implementation of AES we firstly generate eleven keys which would be required in execution of the AES usingthe key expansion program.
After generation of the keys then we perform the initial operation of xoring the 0th key with the state matrix(state matrix contains the input plaintext),with the help of a counter we keep a track of which round is currently running. For each round except the tenth round we perform we perform four operations subbytes, shift rows, mixcolumns and add round key then we obtain the resultant state matrix for that particular round. For tenth round we don't perform the mixcolumn operation and perform only subbyte, shift rows and add round key operation.

But how do we segregate the round that is when we have to perform the mix column operation and when we don'thave to perform the mix column operation?

To do this we make use of combinational circuit. What we do is for each round we have a particular counter value. So we pass the 1st and 3rd bit of our counter to the AND gate. The output of the AND gate acts as the select line forour multiplexer if the select line is 0 (which it will be from round 1 to 9) then we take the output of the mixcolumn whereas when the select line is 1 then we take output of the shift rows and not mixcolumn.
After this we pass the output of the multiplexer to the ADD round key part and then obtain our output.This outputis again given into the same hardware and the output acts as an input for the next round.

The output of the tenth round is our final output of AES. In this way we convert our plaintext to the ciphertextfor the given key.

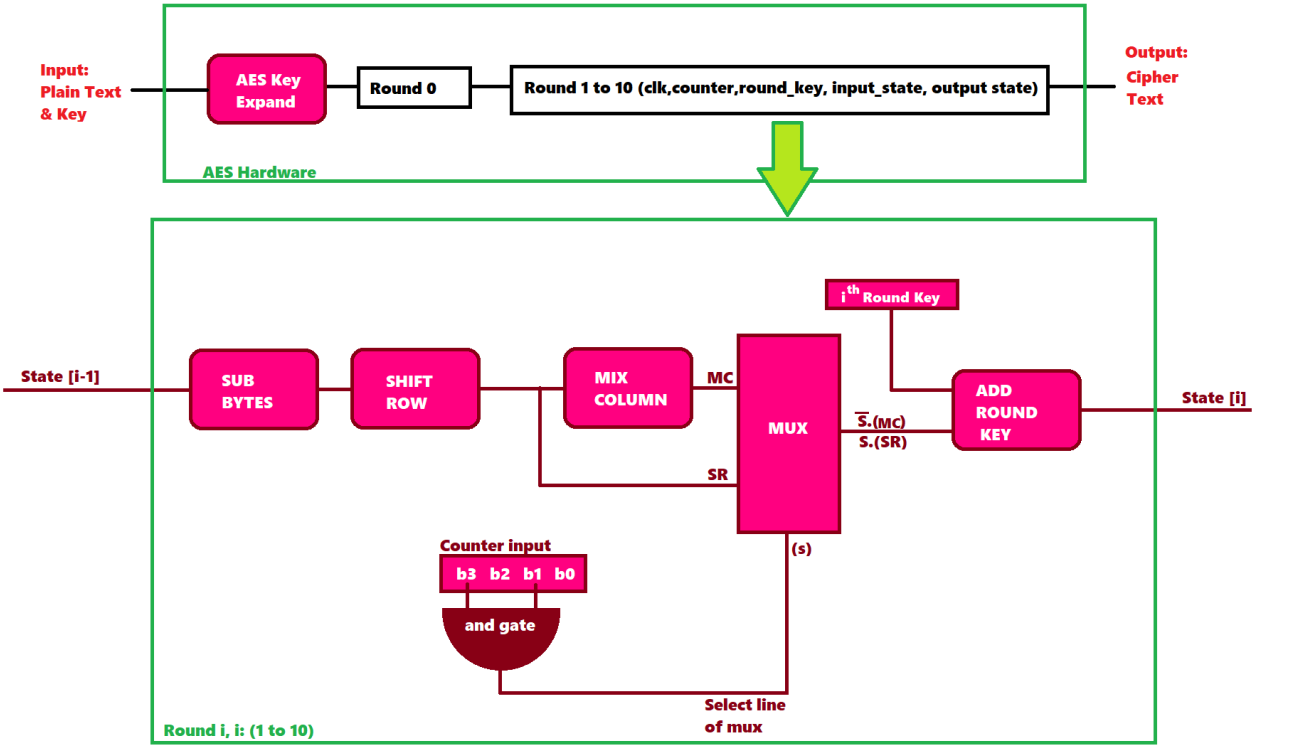

Figure 3: Implementation Of AES

7. Snippets of our Execution of AES

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

vikrantchauhan@Vikrants-MacBook-Air aes_assign2_code % iverilog -o aes.vvp aes_test.v
vikrantchauhan@Vikrants-MacBook-Air aes_assign2_code % vvp aes.vvp
VCD info: dumpfile aes.vcd opened for output.
                   0 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                   5 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                  10 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                  15 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                  20 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 4: iverilog commands

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

                 210 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 215 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 220 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 225 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 230 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 235 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 240 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 245 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 250 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 255 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 260 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 265 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 270 clock=1,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 275 clock=0,Ciphertext=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 280 clock=1,Ciphertext=856f9f522bc1c5fa95401beca5db1046
                 285 clock=0,Ciphertext=856f9f522bc1c5fa95401beca5db1046
                 290 clock=1,Ciphertext=856f9f522bc1c5fa95401beca5db1046
                 295 clock=0,Ciphertext=856f9f522bc1c5fa95401beca5db1046
                 300 clock=1,Ciphertext=856f9f522bc1c5fa95401beca5db1046
Plain Text = 01234567890123456789012345678901, Input Key = 00000000009876543210987654321098

aes_test.v:19: $finish called at 302000 (1ps)
vikrantchauhan@Vikrants-MacBook-Air aes_assign2_code %
```

Figure 5: final output

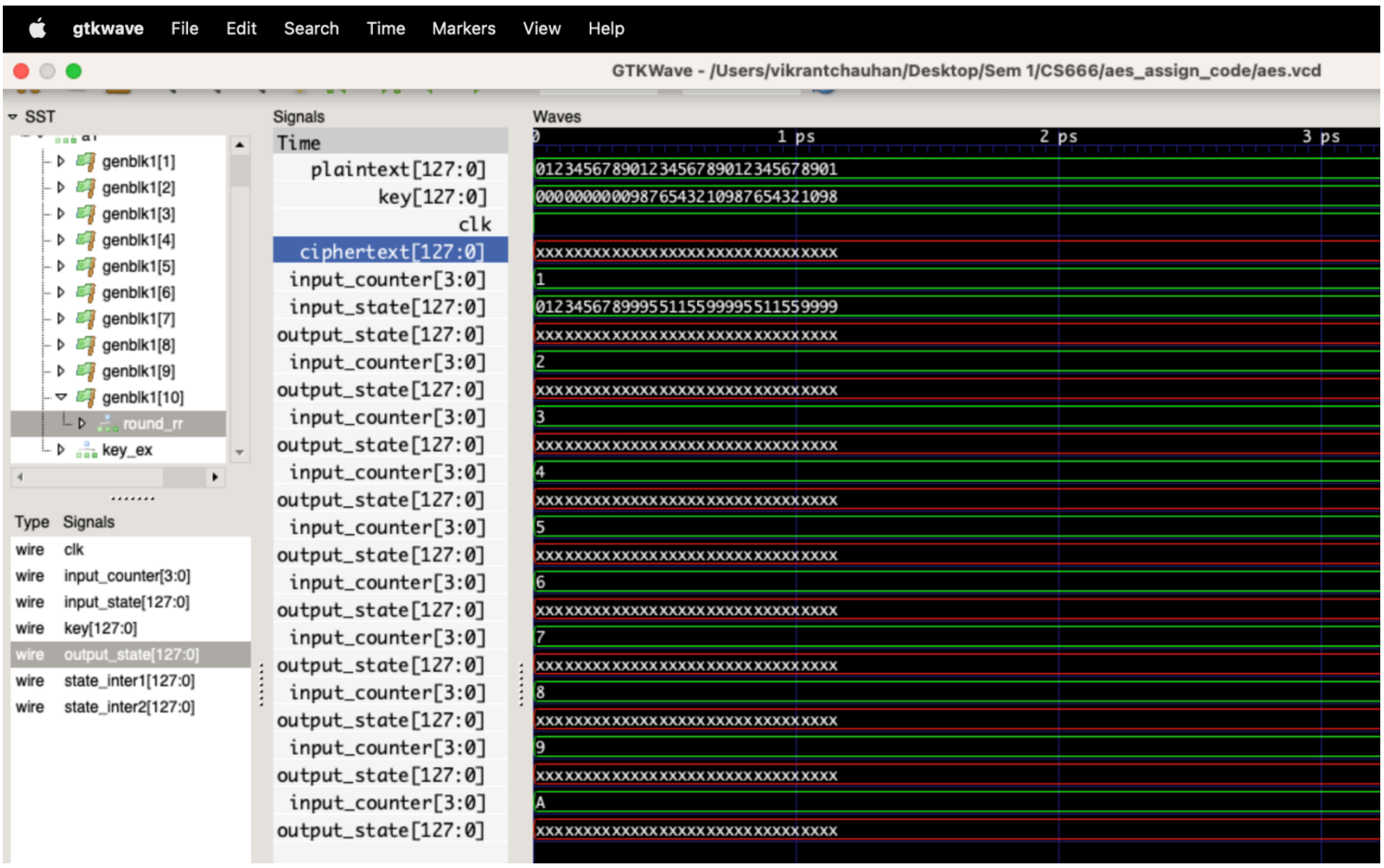

Figure 6: initial configuration, gtkwave (t=0)

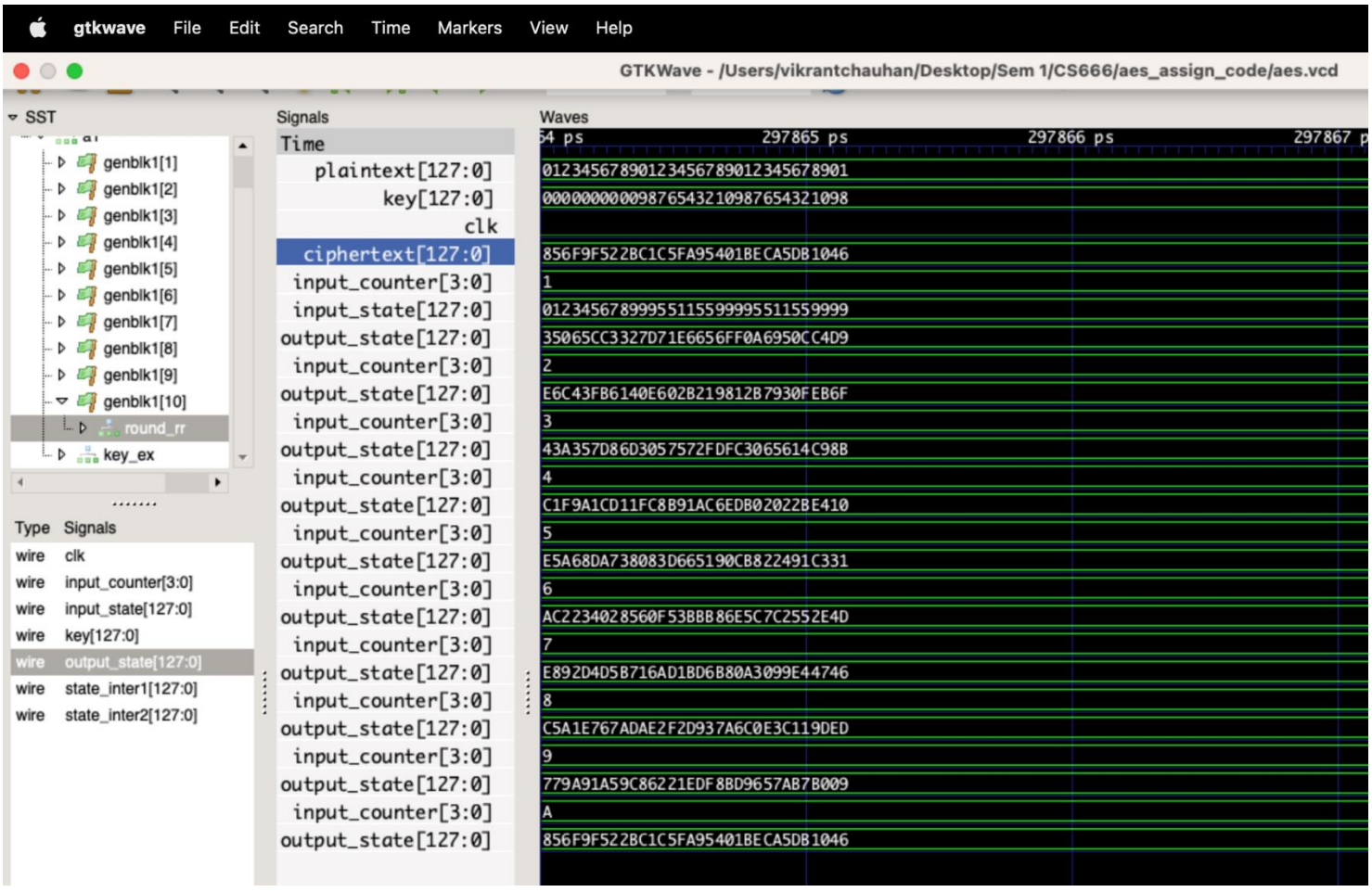

Figure 7: final configuration, gtkwave (t=280)