Grant Lance

ECE 474 Spring 2024

Homework Assignment 4

AES Encryption

Overall this testing confirmation doc is shorter than the last few, namely as there is only one block that needs to be tested/confirmed. This document will contain all relevant testing results, comments, and an appendix where handwritten notes will be saved.

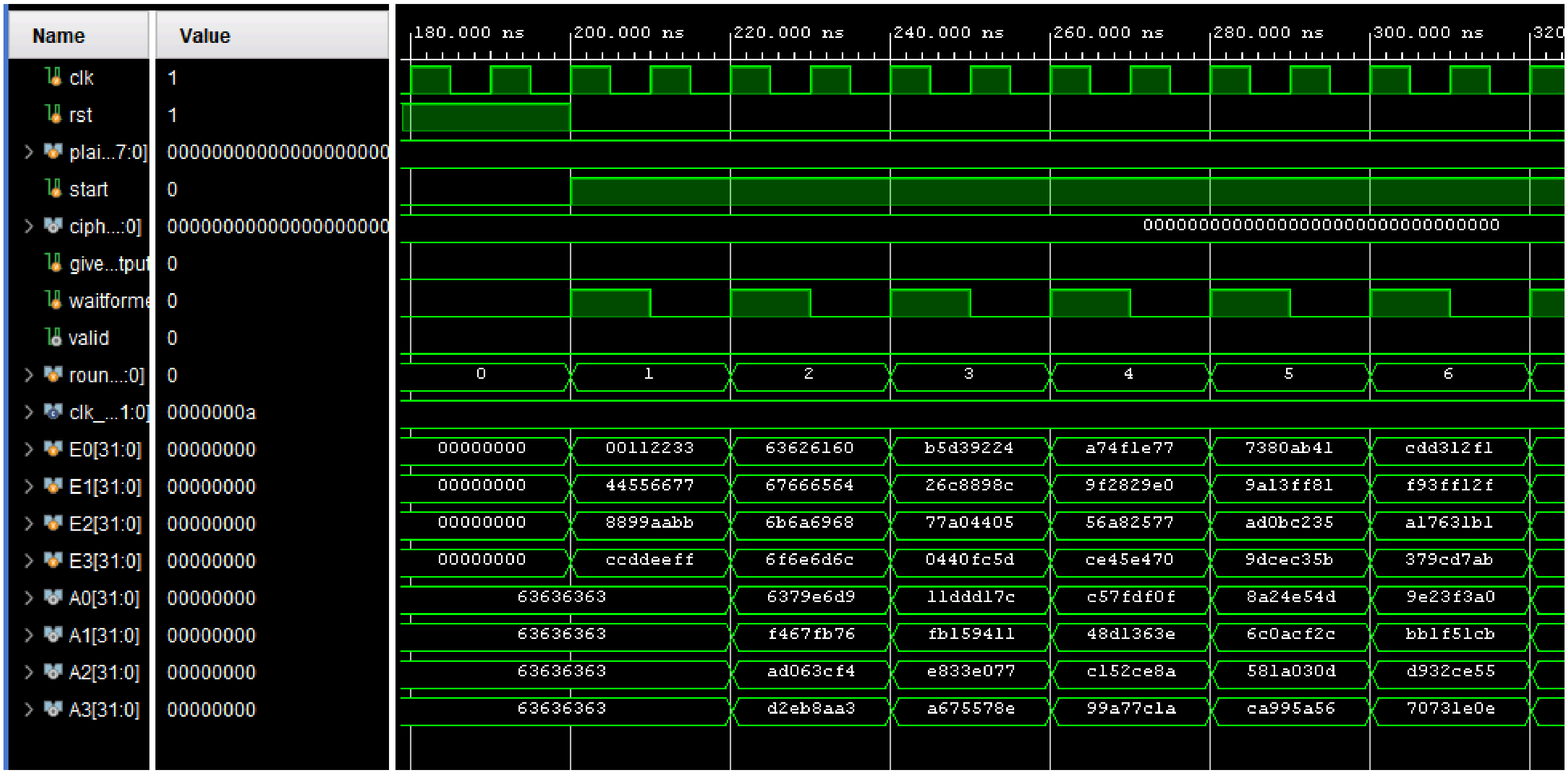
1) **AES.v**
    a) This is a breakdown of the elements that make up the Verilog code for this block. It is copy and pasted from the comments section of AES.v:
        i) In order to do the looping correctly roundCount, finalround, giveoutput, and waitforme flags will be used Essentially, the idea of the waitforme flag is to raise a flag that on the next cycle the AES block can grab. If you nest “if” statements inside of another that looks for this flag you can shift between performing two operations. One operation   when you meet your flag condition and another when the program goes to the else section of that if The two operations that the block will shift between is checking for if cipher output is neccesary/normal round to round operation, and outputing the cipher when done encrypting
        ii) roundCount will be used to monitor the round number and make sure that the correct amount of encryption cycles are ran and that the correct flags get raised so that the ciphertex it output
        iii) giveoutput will be a flag to signal that the ciphertext will be expected as output   that same round. as waitforme will be 0 at the start of round 10, the block will attempt to execute everything nesteed under if(waitforme == 0). When none of the statements pass, the block will then move on to check for the status of round 10. Because this is in the else outside of these if statements, it will be able to respond to this change.

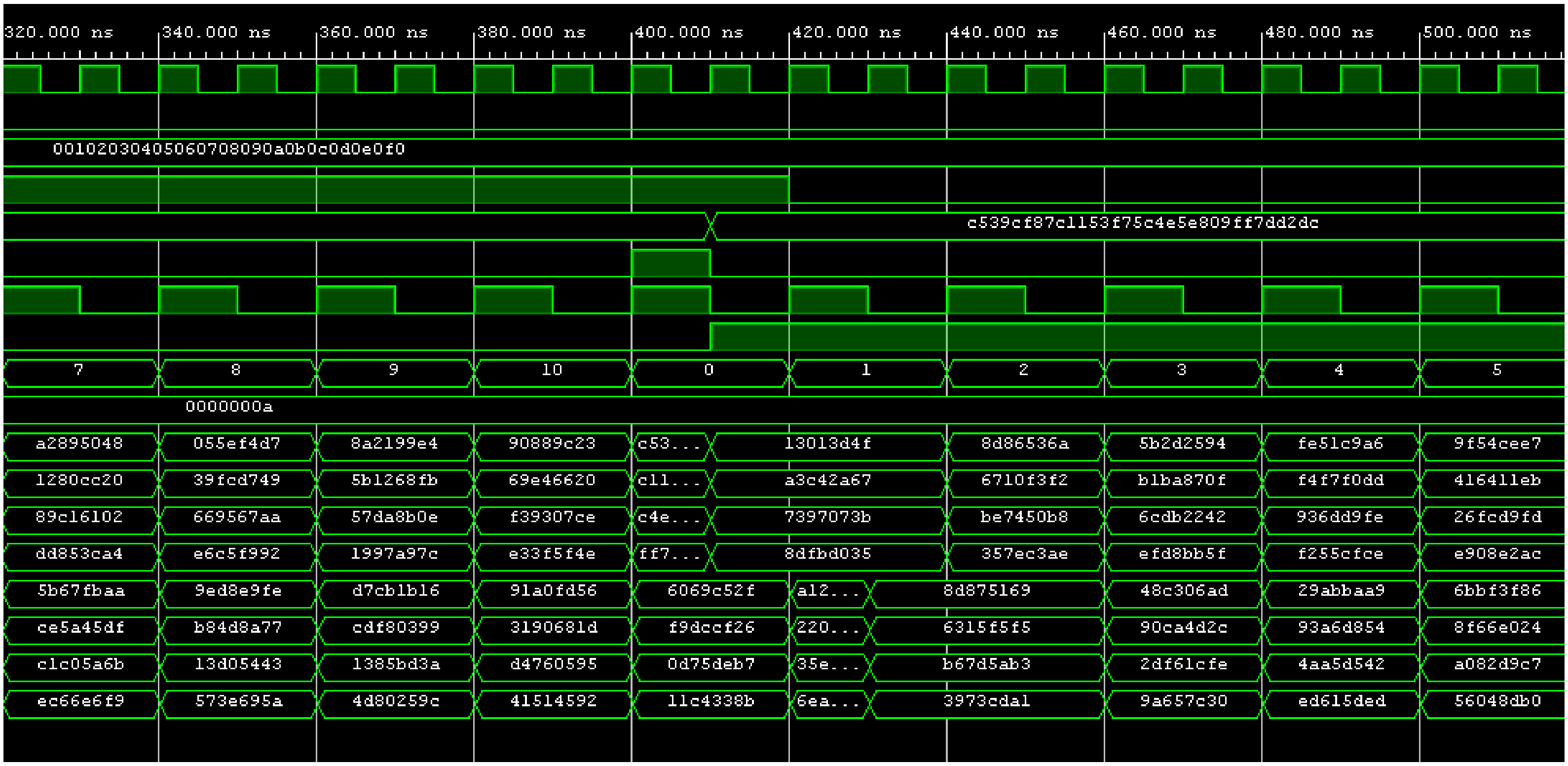
iv) finalround will be a flag that is used in conjunction with giveoutput. Becuase of this, the flag will be raised on round 9 (when roundCount ==9); so that when the ttables lookup the the entries for this round, they grab the correct bits. To do this the finalround is concatonated on the head (the leftmost bit) of TXXin

v) the buffers Ax/Ex will contain 32 bit chunks of the 128 bit plaintext. this follows the encryption operations outlined in the paper given.more notes on the operations of encryption included in testing doc appendix

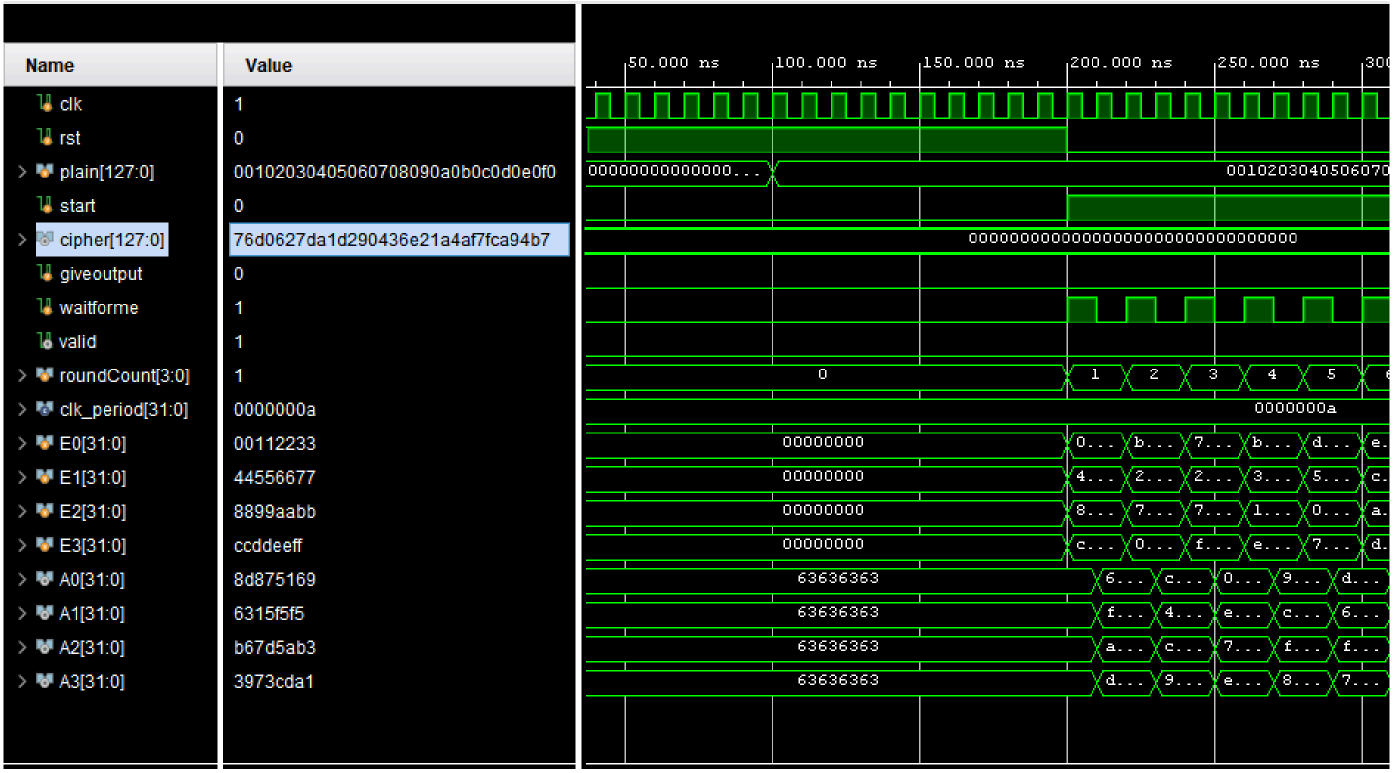
2) tb_AES.v

a) The screenshots below show a successful run of AES verified with the testbench. When the valid flag is raised E0,1,2,3 are concatenated and pushed to the ciphertext output. The correct output can be seen - 0c539cf87c1153f75c4e5e809ff7dd2dc.

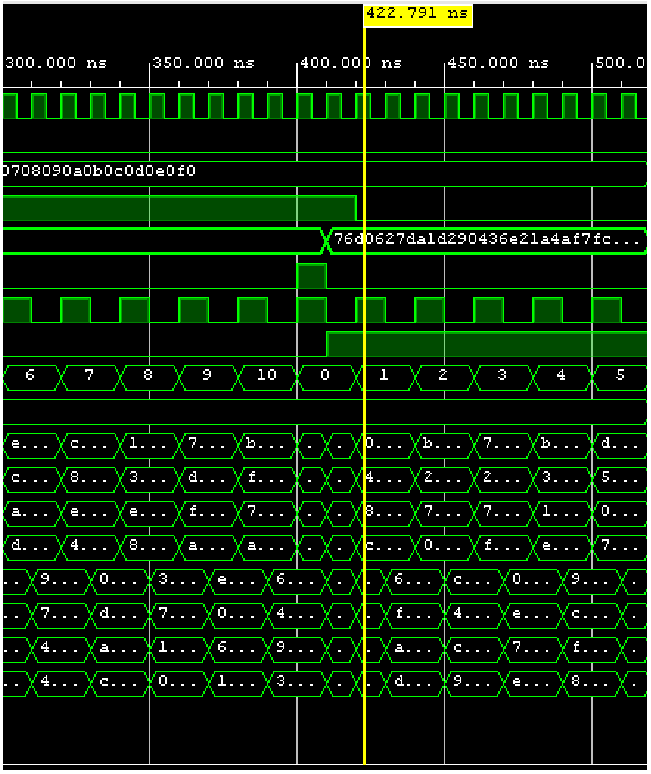




b) After meeting with Jun, he said that the output was incorrect. This was due to some timing issues that I knew I had, but was unfamiliar with how to solve. With output registers attached the BRAM takes two clock cycles to deliver the correct output. Either removing them or modifying the clock speed given to the BRAM will fix this. For simplicity I opted to just remove the output registers. Below are screenshots showing the correct output- 76d0627da1d290436e21a4af7fca94b7. Note they also show the timing issue has been resolved.



Name	Value
clk	1
rst	0
plain[127:0]	00102030405060708090a0b0c0d0e0f0
start	0
cipher[127:0]	76d0627da1d290436e21a4af7fca94b7
giveoutput	0
waitforme	1
valid	1
roundCount[3:0]	1
clk_period[31:0]	0000000a
E0[31:0]	00112233
E1[31:0]	44556677
E2[31:0]	8899aabb
E3[31:0]	ccddeeff
A0[31:0]	8d875169
A1[31:0]	6315f5f5
A2[31:0]	b67d5ab3
A3[31:0]	3973cda1



3) Appendix

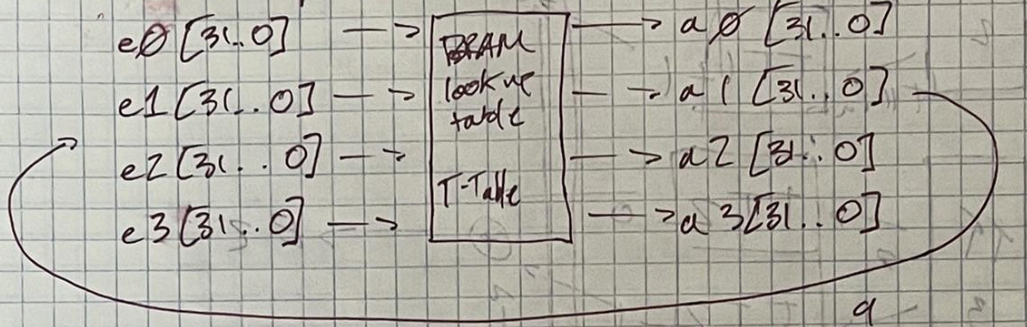
474 HA4 BRAM AES Encryption

10 encryption rounds

128 bit block (plaintext → ciphertext)

Key → 32 char from BRAM } → 32 char In/Out

four signals for each round



do a total of ~~10~~ 9 times
(~~10~~1 rounds)

e0_1 = In[31..0] e2_1 = In[95:64]

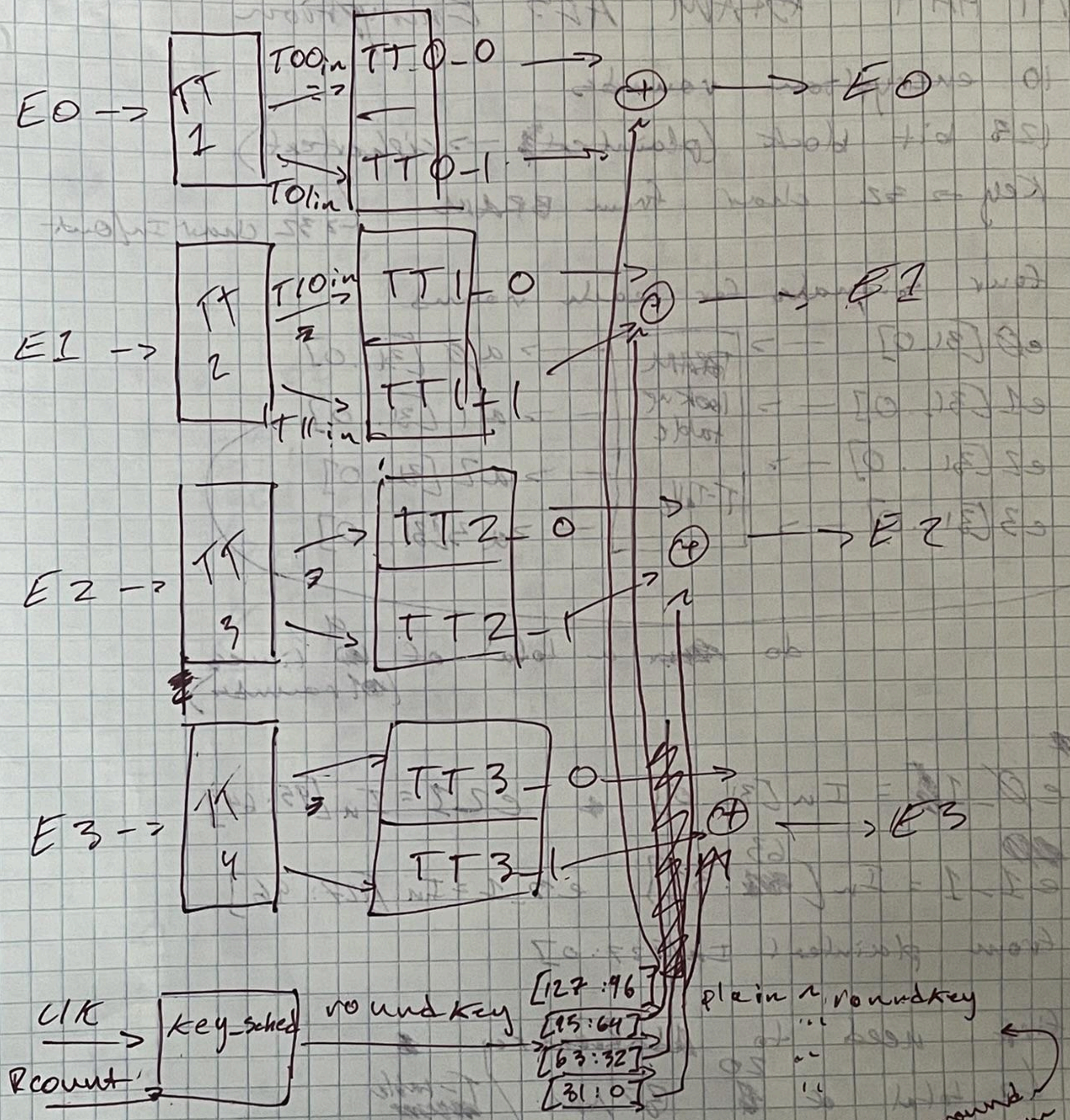
e1_1 = In[63:32] e3_1 = In[127:96]

from plaintext In[127:0]

first need to AddRoundKey

→ total of 20 BRAM / T-table

- the first round will be the 128 bit plaintext with K1 (the first round key) added
- the next 9 rounds will perform SubBytes, ShiftRows, Mix Columns then AddRoundKey
- the final round will only compute add Round Key, ShiftRows, SubBytes



This is the whole 128 bit key
↳ need to parse into corresponding
~~certain~~ 32 bit sections

T00 in

T10 in

T20 in

T30 in

$$\begin{bmatrix} \text{T00in} & \text{T01in} \\ \text{T10in} & \text{T11in} \end{bmatrix}\begin{bmatrix} \text{T02in} & \text{T03in} \\ \text{T12in} & \text{T13in} \end{bmatrix}$$

$$\begin{bmatrix} \text{T20in} & \text{T21in} \\ \text{T30in} & \text{T31in} \end{bmatrix}\begin{bmatrix} \text{T22in} & \text{T23in} \\ \text{T32in} & \text{T33in} \end{bmatrix}$$

~~E0' = K ⊕ T00out ⊕ T01out ⊕ T02out ⊕ T03out~~

E0':	T00in	T11in	T22in	T33in
E1':	T03in	T10in	T21in	T32in
E2':	~~T02in~~ T02in	T13in	T20in	T31in
E3':	T01in	T12in	T23in	T30in

E0[31:24] E1[23:16] E2[15:8] ~~E2~~ E3[7:0]

E3[31:24] E1[

$E0,1,2,3 \oplus K,0,1,2,3 \rightarrow E'0,1,2,3$

~~$E0[31:24]$ $E0[23:16]$~~

$E'0[31:24]$	$E'1[31:24]$	$E'2[31:24]$	$E'3[31:24]$
$E'0[23:16]$	$E'1[23:16]$	$E'2[23:16]$	$E'3[23:16]$
$E'0[15:8]$	$E'1[15:8]$	$E'2[15:8]$	$E'3[15:8]$
$E'0[7:0]$	$E'1[7:0]$	$E'2[7:0]$	$E'3[7:0]$

for new E (E for the next round)

$$E0 = \overset{T00in}{E'0[31:24]} \,||\, \overset{T10in}{E'1[23:16]} \,||\, \overset{T20in}{E'2[15:8]} \,||\, \overset{T30in}{E'3[7:0]}$$

$$E1 = \overset{T01in}{E'1[31:24]} \,||\, \overset{T11in}{E'2[23:16]} \,||\, \overset{T21in}{E'3[15:8]} \,||\, \overset{T31in}{E'0[7:0]}$$

$$E2 = \overset{T02in}{E'2[31:24]} \,||\, \overset{T12in}{E'3[23:16]} \,||\, \overset{T22in}{E'0[15:8]} \,||\, \overset{T23in}{E'1[7:0]}$$

$$E3 = \overset{T03in}{E'3[31:24]} \,||\, \overset{T13in}{E'0[23:16]} \,||\, \overset{T23in}{E'1[15:8]} \,||\, \overset{T33in}{E'2[7:0]}$$