ECE385

Spring 2019

Experiment #9
SOC With Advanced Encryption Standard
In SystemVerilog

Austin Lee (sal3)

Zuling Chen (zulingc2)

Patrick Wang, Gene Shiue

Tuesday 3PM ABE

Introduction

Experiment 9 was an expansion on using the parallel of the FPGA board's hardware and software components to recreate the 128bit Advanced Encryption Standard. AES takes a message to encrypt and a key to encrypt it with. Through several rounds of SubBytes, ShiftRows, MixColumns, AddRoundKey, the message gets encrypted. Through the inverse of some of these functions, that same message would get decrypted using the key schedule generated. In week 1, the main task was to create the encryption part of the algorithm on Eclipse using C code. The only connection we made with the hardware in week one was to display data onto the HEX displays. For week two, our objective was to create the decryption part in hardware and then communicate that result back to the software to be displayed on the Eclipse console.

Written Description and Diagrams of the AES encryptor/decryptor

Software Encryption

For the software part of the lab, we were given an initial c file that contained existing code for charToHex, testing and benchmarking. Our task was to write the functions encrypt and decrypt. To do this, we made use of several helper functions representing the different steps of the encryption process. The code first asks the user to input a message of up to 32 characters that is to be encrypted. That message data is sent to the FPGA using the charToHex function provided. The code then prompts the user for an encryption key, which is also sent to the FPGA. The NIOS II then will expand the key with our Key Expansion function to generate the 11 keys to use during the encryption process. The code will go onto AddRoundKey once then then goes into the 9 loops for subBytes, ShiftRows, MixColumns, and AddroundKey. AddRoundKey XOR's the message with the generated key schedule. SubBytes takes the message bytes and switches then out from the provided matrix of Rijndael bytes. ShiftRows will shift the row by the row number. MixColumns uses gf_mul[][] to grab existing matrix multiplication result. One more round is done minus the AddRoundKey and the resulting message is printed to the console. All of these operators are done through the NIOS II processor.

Hardware Decryption

Decryption essentially takes the inverse steps of encryption and performs the steps backwards. Decryption will first AddRoundKey before going into the 9 loops of InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. Then finally, it will go through a final round of InvShiftRows, InvSubBytes, and AddRoundKey before sending the data back to the console to be printed out. We controlled the 9 loops of the algorithm through a state machine using control signals to determine where in the loop we were and to send signals back to the console to tell it whether we were done or not.

Avalon_aes_interface description

The avalon aes interface file serves as a bridge between hardware and software. The input signals consists of many control signals including read, write, chip select, bye enable, address and data. This sv file creates 16 registers of 32 bits long. Depending on what signal is sent, data would either be written to a register, written out to LEDs, or a done signal would be sent to the software. The Software would send a start signal to the register 14 and that holds the bit that tells our state machine to start. While the hardware is running, it'll constantly tell software that it's not done yet and just send low signals for register 15. Once register 15 holds an active high after the hardware is done and holds the right results, that signal will be read by the C code which will then move on to grab that data and display it onto the console.

Block Diagram(s)

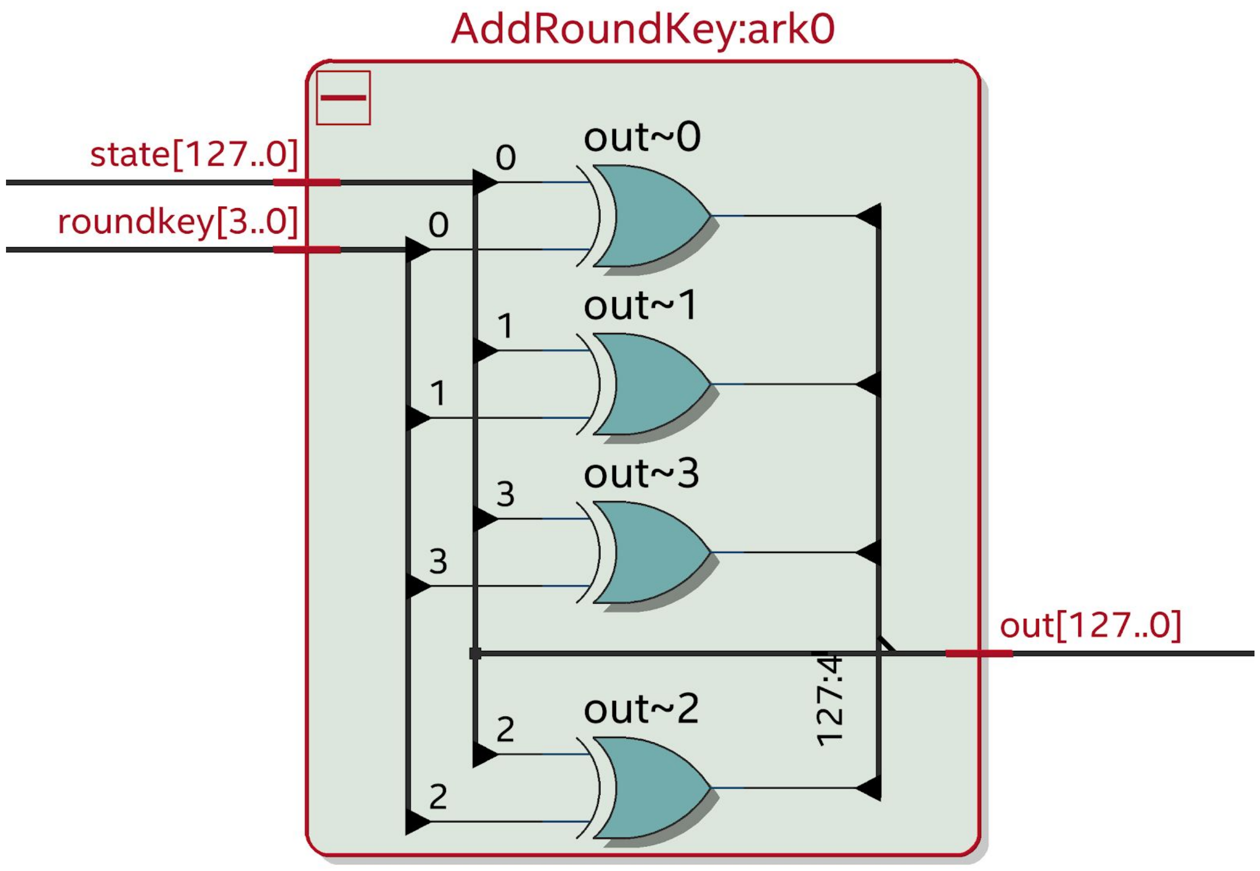


Figure 1: AddRoundKey module

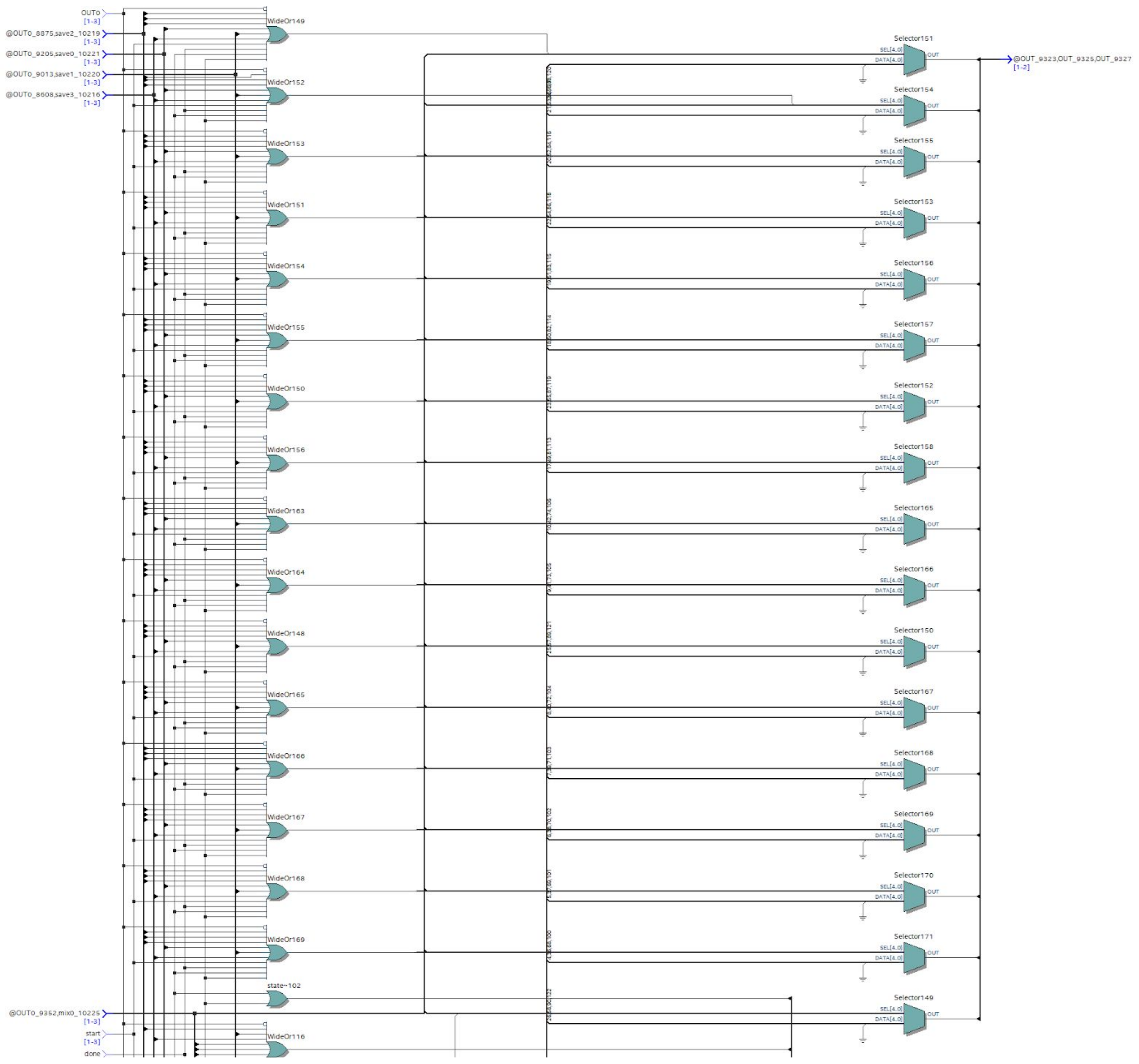


Figure 2: AES.sv (part 1)

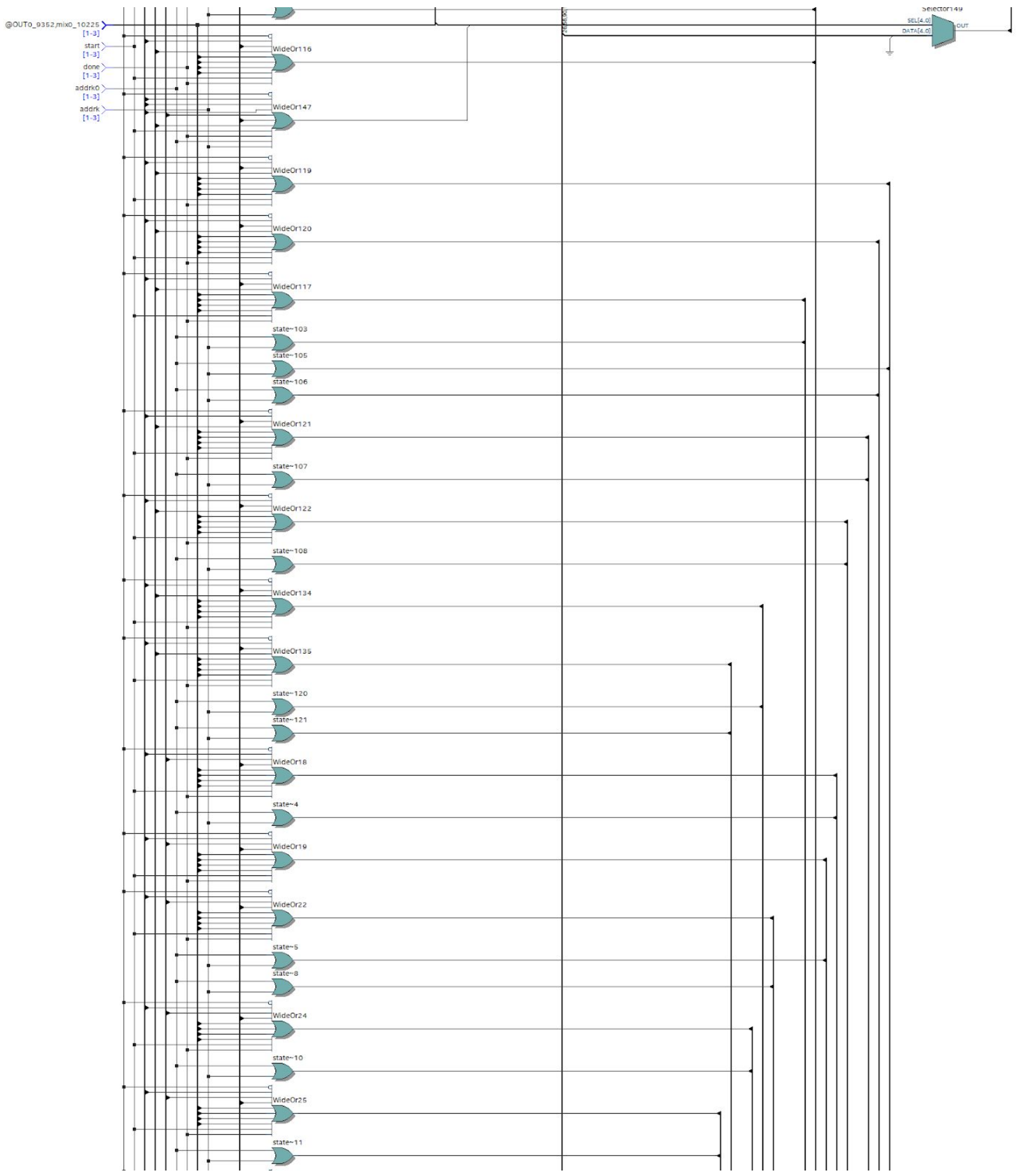


Figure 3: AES.sv (part 2)

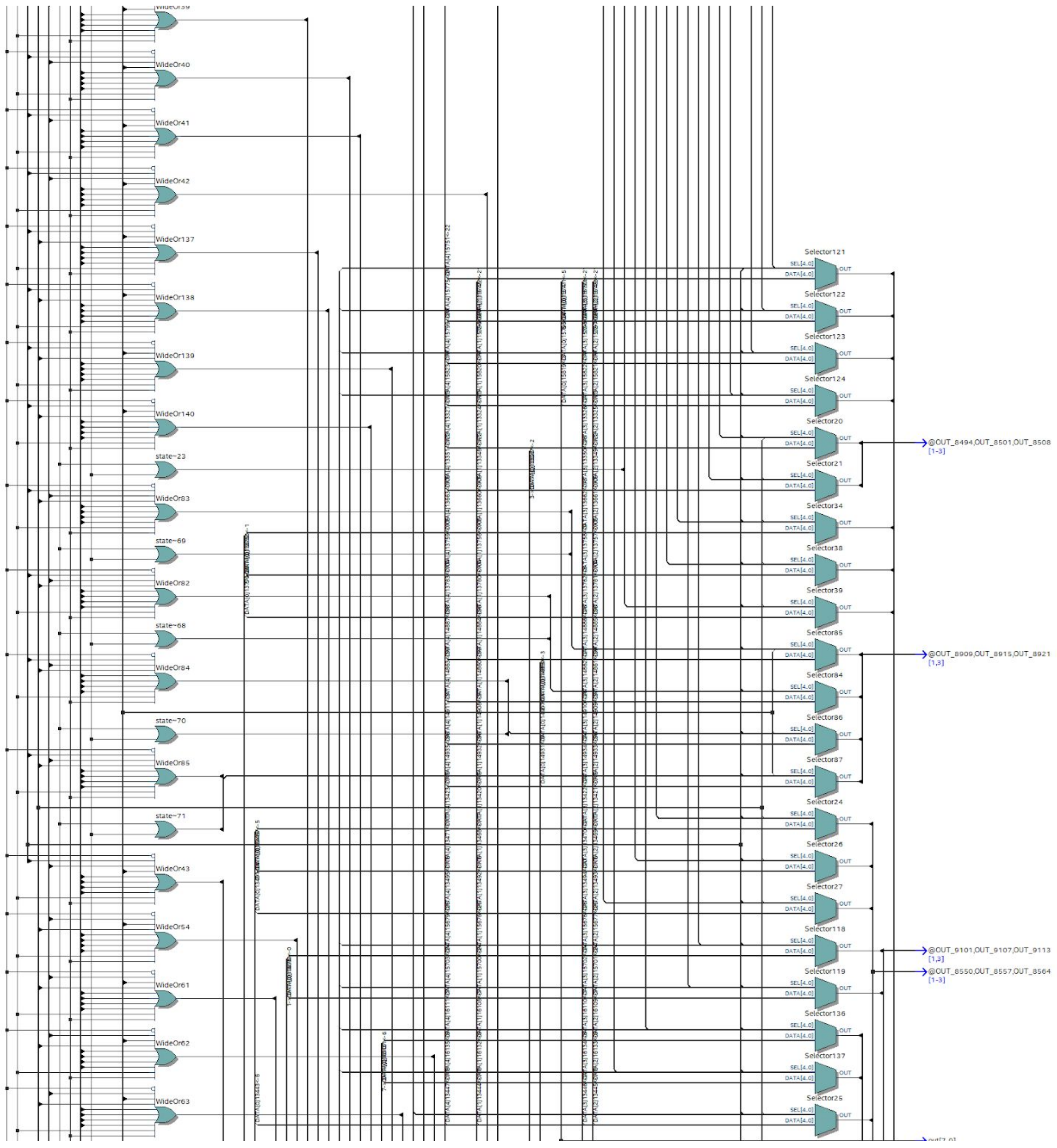


Figure 4: AES.sv (part 3)

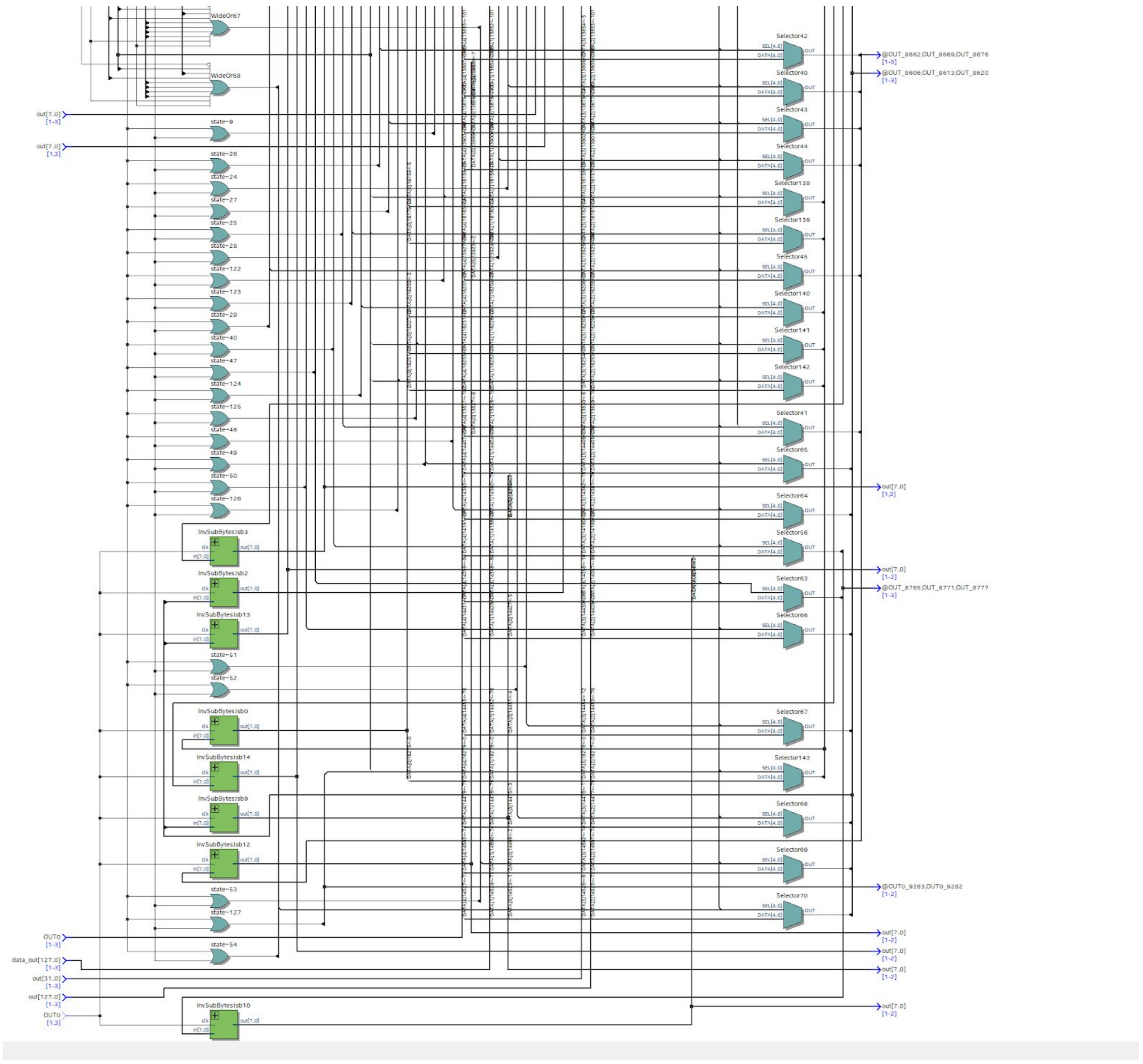
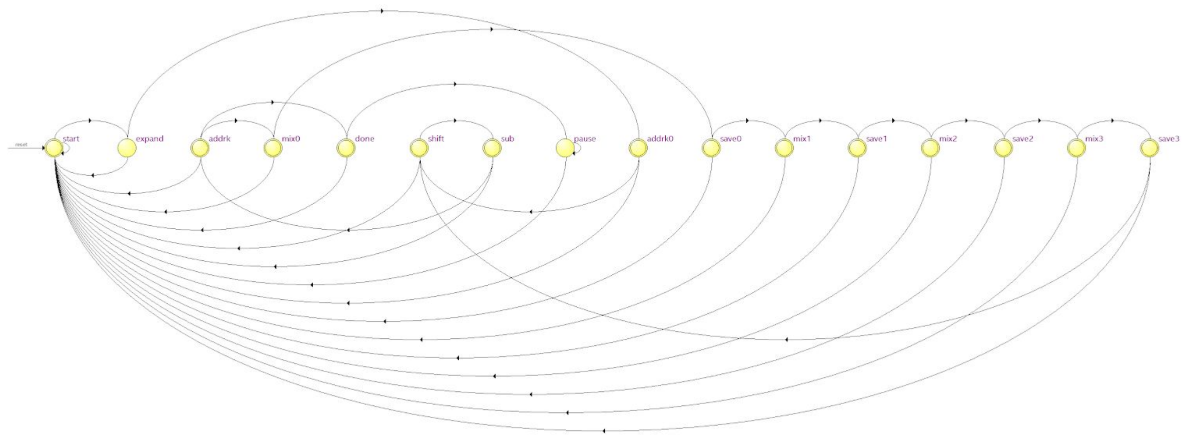


Figure 5: AES.sv (part 4)

All other modules and instances were ***provided*** *modules/files.*

State Diagram of AES decryptor controller



Module Descriptions

Module: SubBytes
Inputs: clk, [7:0] in
Outputs: [7:0] out
Description: This was basically one huge mux.
Purpose: Depending on the input of bytes, SubBytes would go through the table and replace it with the corresponding Rijndael bytes for encryption.

Module: InvSubBytes
Inputs: clk, [7:0] in
Outputs: [7:0] out
Description: This was basically one huge mux.
Purpose: This served as the inverse of SubBytes used in decryption. This also makes use of the matrix of Rijndael bytes to take an input and performs the inverse subbytes operation on it.

Module: Key Expansion
Inputs: clk, [127:0] Cipherkey
Outputs:[1407:0] KeySchedule
Description: This module generated 11 keys from a given cipherkey.

Purpose: This was used in decryption as the inverse of key expansion in the software encryption part of the lab. It essentially takes words from the previous keys and XORs it with subwords generated by SubBytes.

Module: InvShiftRows
Inputs: [127:0] data_in
Outputs: [127:0] data_out
Description: File given by course staff that is meant as an inverse to Shift Rows
Purpose: Performs the inverse of ShiftRows to be used in decryption.

Module: InvMixColumns
Inputs: [31:0] in
Outputs: [31:0] out
Description: File given to us by the ECE385 staff to perform the inverse to MixColumns.
Purpose: Performs the inverse operation of MixColumns by calling on the gfmul operator and XORing the results together.

Module: hexdriver
Inputs: [3:0] In
Outputs: [6:0] Out
Description: This was implemented using a Mux.
Purpose: This module served to change binary values to hex values that could be sent to LEDs to display hex values instead of binary.

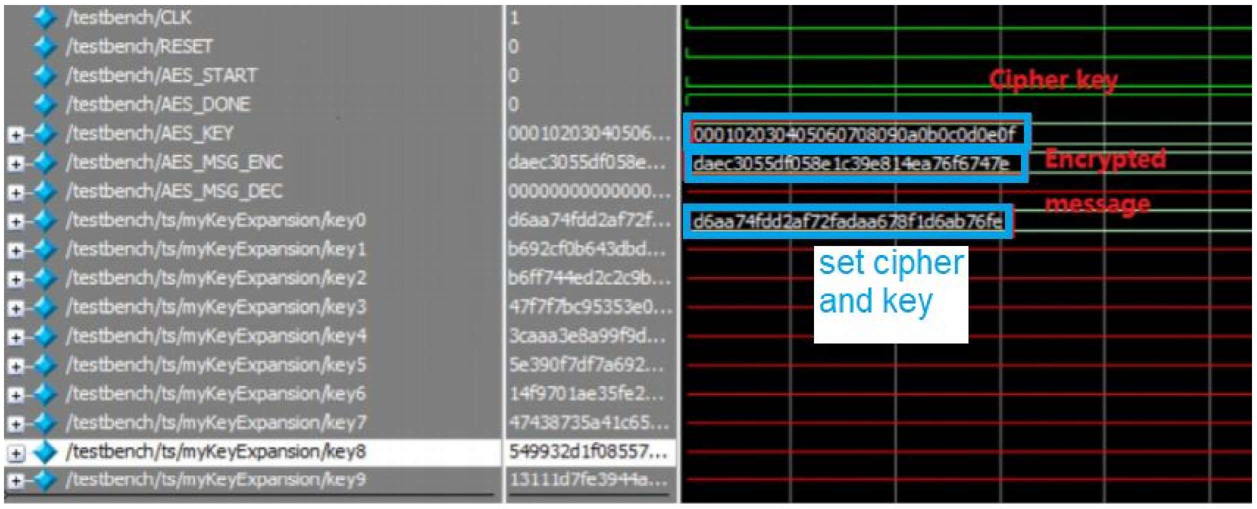
Module: AddRoundKey
Inputs: [127:0] state, [1407:0] key_schedule, [3:0] round
Outputs: [127:0] out
Description: This was a mux alongside an XOR method.
Purpose: Depending on what round we were on, the current state of the message would be XORed by a different 128 bits of the previously generated key schedule. This function is exactly the same as in software.

Module: AES
Inputs: Clk RESET, AES_START, [127:0] AES_KEY, AES_MSG_ENC
Outputs: AES_DONE, AES_MSG_DEC
Description: This served as our initiation module for the AES state machine

Purpose: Call upon the AES state machine and pass on the data needed to be encrypted along with the key and decrypt it. Once the message is done, AES_DONE is set to a high so the software will know that it's done.

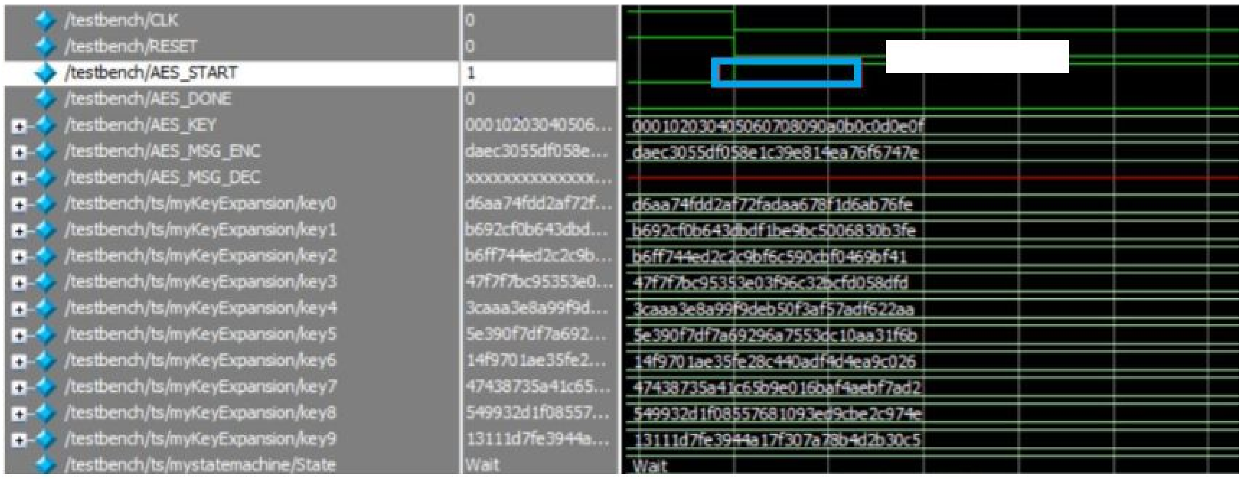
Module: **avalon_aes_interface**
Inputs: CLK, RESET, AVL_READ, AVL_WRITE, AVL_CS, [3:0] AVL_BYTE_EN, AVL_ADDR,[31:0] AVL_WRITEDATA
Outputs: [31:0] AVL_READDATA, EXPORT_DATA
Description: This serves as the interface between software and hardware.
Purpose: 16 registers are generated with 32 bits each. This module was explained above.

<u>Annotated Simulation of the AES decryptor</u>



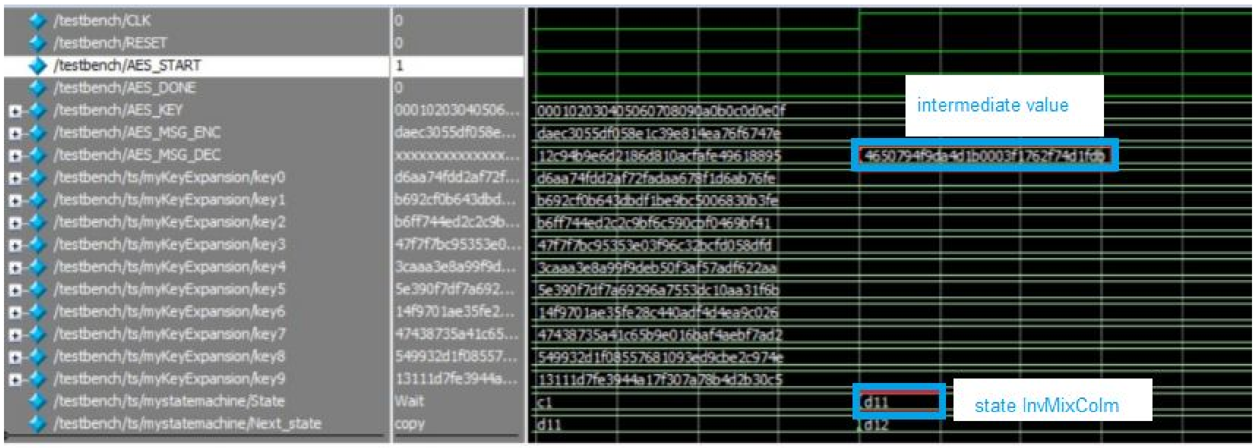
```
/testbench/CLK                         1
/testbench/RESET                       0
/testbench/AES_START                   0
/testbench/AES_DONE                    0
/testbench/AES_KEY                     000102030405 06...   000102030405060708090a0b0c0d0e0f
/testbench/AES_MSG_ENC                 daec3055df058e...     daec3055df058e1c39e814ea76f6747e
/testbench/AES_MSG_DEC                 0000000000000...
/testbench/ts/myKeyExpansion/key0      d6aa74fdd2af72f...    d6aa74fdd2af72fadaa678f1d6ab76fe
/testbench/ts/myKeyExpansion/key1      b692cf0b643dbd...
/testbench/ts/myKeyExpansion/key2      b6ff744ed2c2c9b...
/testbench/ts/myKeyExpansion/key3      47f7f7bc95353e0...
/testbench/ts/myKeyExpansion/key4      3caaa3e8a99f9d...
/testbench/ts/myKeyExpansion/key5      5e390f7df7a692...
/testbench/ts/myKeyExpansion/key6      14f9701ae35fe2...
/testbench/ts/myKeyExpansion/key7      47438735a41c65...
/testbench/ts/myKeyExpansion/key8      549932d1f08557...
/testbench/ts/myKeyExpansion/key9      13111d7fe3944a...
```

Input cipher key as
000102030405060 8090a0b0c0d0e0f
message input as
daec3055df058e1c39e814ea76f6747e



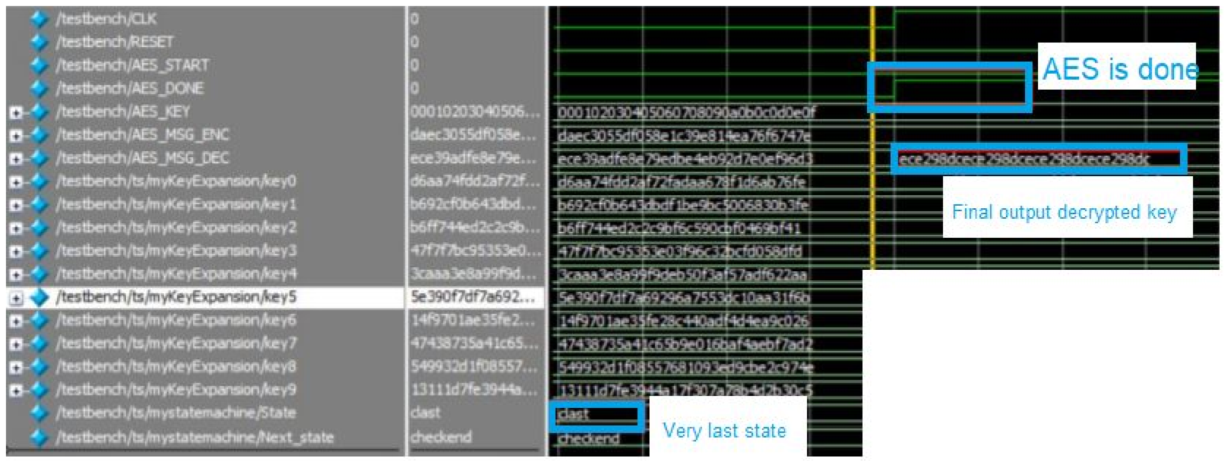
```
/testbench/CLK                              0                   
/testbench/RESET                            0                   
/testbench/AES_START                        1                   
/testbench/AES_DONE                         0                   
/testbench/AES_KEY                          00010203040506...   000102030405060708090a0b0c0d0e0f
/testbench/AES_MSG_ENC                      daec3055df058e...   daec3055df058e1c39e814ea76f6747e
/testbench/AES_MSG_DEC                      xxxxxxxxxxxxxx...   
/testbench/ts/myKeyExpansion/key0           d6aa74fdd2af72f...  d6aa74fdd2af72fadaa678f1d6ab76fe
/testbench/ts/myKeyExpansion/key1           b692cf0b643dbd...   b692cf0b643dbdf1be9bc5006830b3fe
/testbench/ts/myKeyExpansion/key2           b6ff744ed2c2c9b...  b6ff744ed2c2c9bf6c590cbf0469bf41
/testbench/ts/myKeyExpansion/key3           47f7f7bc95353e0...  47f7f7bc95353e03f96c32bcfd058dfd
/testbench/ts/myKeyExpansion/key4           3caaa3e8a99f9d...   3caaa3e8a99f9deb50f3af57adf622aa
/testbench/ts/myKeyExpansion/key5           5e390f7df7a692...   5e390f7df7a69296a7553dc10aa31f6b
/testbench/ts/myKeyExpansion/key6           14f9701ae35fe2...   14f9701ae35fe28c440adf4d4ea9c026
/testbench/ts/myKeyExpansion/key7           47438735a41c65...   47438735a41c65b9e016baf4aebf7ad2
/testbench/ts/myKeyExpansion/key8           549932d1f08557...   549932d1f08557681093ed9cbe2c974e
/testbench/ts/myKeyExpansion/key9           13111d7fe3944a...   13111d7fe3944a17f307a78b4d2b30c5
/testbench/ts/mystatemachine/State          Wait                Wait
```

AES_Start gets a high and decryption starts, 9 keys generated



```
/testbench/CLK                              0
/testbench/RESET                            0
/testbench/AES_START                        1
/testbench/AES_DONE                         0
/testbench/AES_KEY                          000102030405 06...  000102030405060708090a0b0c0d0e0f
/testbench/AES_MSG_ENC                      daec3055df058e...   daec3055df058e1c39e814ea76f6747e
/testbench/AES_MSG_DEC                      xxxxxxxxxxxxxx...   12c94b9e6d2186d810acfafe49618895   4650794f9da4d1b0003f1762f74d1fdb
/testbench/ts/myKeyExpansion/key0           d6aa74fdd2af72f...  d6aa74fdd2af72fadaa678f1d6ab76fe
/testbench/ts/myKeyExpansion/key1           b692cf0b643dbd...   b692cf0b643dbdf1be9bc5006830b3fe
/testbench/ts/myKeyExpansion/key2           b6ff744ed2c2c9b...  b6ff744ed2c2c9bf6c590cbf0469bf41
/testbench/ts/myKeyExpansion/key3           47f7f7bc95353e0...  47f7f7bc95353e03f96c32bcfd058dfd
/testbench/ts/myKeyExpansion/key4           3caaa3e8a99f9d...   3caaa3e8a99f9deb50f3af57adf622aa
/testbench/ts/myKeyExpansion/key5           5e390f7df7a692...   5e390f7df7a69296a7553dc10aa31f6b
/testbench/ts/myKeyExpansion/key6           14f9701ae35fe2...   14f9701ae35fe28c440adf4d4ea9c026
/testbench/ts/myKeyExpansion/key7           47438735a41c65...   47438735a41c65b9e016baf4aebf7ad2
/testbench/ts/myKeyExpansion/key8           549932d1f08557...   549932d1f08557681093ed9cbe2c974e
/testbench/ts/myKeyExpansion/key9           13111d7fe3944a...   13111d7fe3944a17f307a78b4d2b30c5
/testbench/ts/mystatemachine/State          Wait                c1                                 d11
/testbench/ts/mystatemachine/Next_state     copy                d11                                d12
```

Program has performed addroundkey and will perform InvMixColumns and the AES_MSG_DEC gets intermediate values as the program runs through the loops



```
/testbench/CLK                            0
/testbench/RESET                          0
/testbench/AES_START                      0
/testbench/AES_DONE                       0
/testbench/AES_KEY                        000102030405060...   000102030405060708090a0b0c0d0e0f
/testbench/AES_MSG_ENC                    daec3055df058e...    daec3055df058e1c39e814ea76f6747e
/testbench/AES_MSG_DEC                    ece39adfe8e79e...    ece39adfe8e79edbe4eb92d7e0ef96d3   ece298dcece298dcece298dcece298dc
/testbench/ts/myKeyExpansion/key0         d6aa74fdd2af72f...   d6aa74fdd2af72fadaa678f1d6ab76fe
/testbench/ts/myKeyExpansion/key1         b692cf0b643dbd...    b692cf0b643dbdf1be9bc5006830b3fe
/testbench/ts/myKeyExpansion/key2         b6ff744ed2c2c9b...   b6ff744ed2c2c9bf6c590cbf0469bf41
/testbench/ts/myKeyExpansion/key3         47f7f7bc95353e0...   47f7f7bc95353e03f96c32bcfd058dfd
/testbench/ts/myKeyExpansion/key4         3caaa3e8a99f9d...    3caaa3e8a99f9deb50f3af57adf622aa
/testbench/ts/myKeyExpansion/key5         5e390f7df7a692...    5e390f7df7a69296a7553dc10aa31f6b
/testbench/ts/myKeyExpansion/key6         14f9701ae35fe2...    14f9701ae35fe28c440adf4d4ea9c026
/testbench/ts/myKeyExpansion/key7         47438735a41c65...    47438735a41c65b9e016baf4aebf7ad2
/testbench/ts/myKeyExpansion/key8         549932d1f08557...    549932d1f08557681093ed9cbe2c974e
/testbench/ts/myKeyExpansion/key9         13111d7fe3944a...    13111d7fe3944a17f307a78b4d2b30c5
/testbench/ts/mystatemachine/State        dlast                dlast
/testbench/ts/mystatemachine/Next_state   checkend             checkend
```

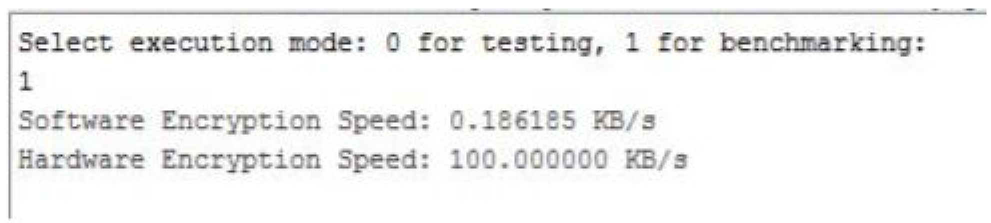
AES_Done is now set to one at the very last state and the decrypted message is stored in AES_MSG_DEC

<u>PostLab Table</u>

LUT	2970
DSP	0
Memory(BRAM)	55296
Flip-Flop	2790
Frequency	148.92 MHz
Static Power	105.40 mW
Dynamic Power	25.66 mW
Total Power	190.24 mW

<u>PostLab Questions</u>

1. Which would you expect to be faster to complete encryption/decryption, the software or hardware? Is this what your results show? (**List your encryption and decryption benchmark here).**

   a. Because decryption uses hardware, we expect it to be faster. It was stated in the lab handbook that software can be used for small tasks like input-put reading, but hardware would be faster in calculations. Hardware optimizes delays and placements for faster operations.
   b. 

```
Select execution mode: 0 for testing, 1 for benchmarking:
1
Software Encryption Speed: 0.186185 KB/s
Hardware Encryption Speed: 100.000000 KB/s
```

2. If you wanted to speed up the hardware, what would you do?
   a. The restriction of the lab required use to only have 1 instance of the InvMixedColumns Module. If we were allowed to implement more instances of this module, our calculation times would be much faster since the module would no longer depend on the previous intermediate output of the InvMixedColumns instance.

<u>Conclusion</u>

We got part of the project working for week 1. We were able to input a message and a key to receive a unique ciphertext in the console. However, our ciphertext did not match up to the solutions provided to us on the course website. To fix this issue, we could’ve added tons of print statements to fix our code bit by bit through each step of the encryption process. In part 2, we were not able to get any functional decryption working. We were able to connect the data from hardware to software, and we showed this through a simple print function. Our program however got stuck in an infinite loop, meaning it never received the done signal from the hardware. We tried backtracking through our state machine through simulations but we were unable to find the source of the problem. It could’ve been chalked up to syntax errors or just missing a signal in a single state. Overall, it would’ve been cool to get this whole algorithm working. The labs we’ve done in the class were interesting, but we never saw any real life applications of them minus video games or a calculator. This lab was the first one which somewhat resembled what FPGA is used for in real life applications. Better documentation of the AES decryption process however would've helped in starting the hardware portion of this lab.