ECE385 Experiment #9

Eric Meyers, Ryan Helsdingen

Section ABG; TAs: Ben Delay, Shuo Liu

April 13th, 2016

emeyer7, helsdin2

I. Introduction

The purpose of this lab was to explore encryption/decryption techniques using the Advanced Encryption Standard (AES). The team wrote both encryption and decryption algorithms that runs on software and hardware respectively. The advantages/disadvantage of these two techniques will be analyzed in the post-lab section.

II. Description of Circuit

The encryption algorithm will be performed on a NIOS-II processor and programmed in C, whereas the decryption algorithm will be performed on a Cyclone IV FPGA and programmed in System Verilog.

The C program onboard the NIOS-II processor will communicate with the hardware on the Altera DE2-115 board and transfer both the encrypted message and key to the hardware. The hardware will then proceed to decrypt this message using the provided key and display the result on the hex display.

III. Purpose of Modules

The AES encryption/decryption algorithm used several modules. However, these can be broken down into software modules and hardware modules. The only software module was the NIOS system used to create the software encryption algorithm in C. The following were the hardware modules and these will be explained in detail when necessary:

- lab9 (top-level)
- aes_controller
- AES
- io_module
- KeyExpansion
- InvShiftRows
- InvSubBytes
- InvSubBytes_16
- InvShiftRows
- InvMixColumns
- AddRoundKey
- HexDriver

First, the encryption algorithm, as stated before was programmed in C on a NIOS-II processor. This processor is contained within the ”NIOS system”.

The NIOS system developed was similar to that of the one created in lab8 except there are 4 PIO modules to allow communication to/from the hardware (to_sw_sig, to_hw_sig, to_sw_port, to_hw_port). A JTAG UART peripheral was used to allow debugging upon the host computer and to allow user input to the C program.

lab9

This is the top-level module containing the connections from the nios_system, aes_controller, io_module, and the hex display.

aes_controller

[TEXT HERE]

AES

[TEXT HERE]

KeyExpansion

[TEXT HERE]

InvShiftRows

[TEXT HERE]

InvSubBytes

[TEXT HERE]

Inv_Sub_Bytes_16
[TEXT HERE]

InvMixColumns
[TEXT HERE]

AddRoundKey
[TEXT HERE]

HexDriver
This module contains the logic necessary to output values in binary/decimal in hex on the hex displays on board the Altera De2-115.

IV. Software/Hardware State Diagram

The software/hardware state diagram is handled in io_module and is shown in Figure 1.

V. Decryption State Diagram

The decryption state diagram is split up into two FSMs for simplicity. One is referred to as the Calculation FSM and the other is referred to as the Round FSM. There are a total of 10 rounds with an extra round to ensure the key_expansion algorithm performs. Each round a series of calculations are performed as determined in the algorithm outlined in the description PDF of this lab. The Round FSM is shown in Figure 2 and the Calculation FSM is shown in Figure 3.

VI. Schematic/Block Diagram

VII. Annotated Pre-Lab Waveforms

The annotated pre-lab waveforms can be found in Figure 4, 5, 6, and 7 for a 2000ns period. The ending decrypted value given plaintext of ”daec3055df058e1c39e814ea76f6747e” with a key of ”000102030405060708090a0b0c0d0e0f” is ”ece298ece298...dc”.

VIII. Post Lab

1) Which would you expect to be faster to complete encryption/decryption, the software or hardware? Is this what your results show?
   - Answer Question
2) If you wanted to speed up the hardware, what would you do? (Note: restrictions of this lab do not apply to answer this question)
   - Answer Question

Resource	Value
LUT	
DSP	
Memory (BRAM)	
Flip-Flop	
Frequency	MHz
Static Power	mW
Dynamic Power	mW
Total Power	mW

TABLE I: Design Statistics

IX. Conclusion

Overall, this lab proved to be very difficult from both a hardware and software perspective. Programming the encryption algorithm in C was difficult for many reasons. Working with two-dimensional arrays is never an easy task in C, and this lab seemed to put this skill to the test. The notation used throughout the algorithm is difficult to understand fully what should be passed into certain functions. However, once a basic working model of the encryption algorithm was developed, it was optimized for patterns, and condensed and therefore much easier to understand.

The hardware decryption was just as difficult as the software. The finite state machines that were developed took a large amount of time to debug using ModelSim. The io_module used to pass information to/from the software proved to be a large time-sing in and of itself.

In the end, the team did not receive full points for demo due to a glitch in passing the data to/from software. However, partial credit was received for showing simulations...

Overall this lab was one of the most difficult ECE385 labs, however it was one of the most rewarding to understand in the end. AES encryption/decryption is now very clear to the team.

X. Figures

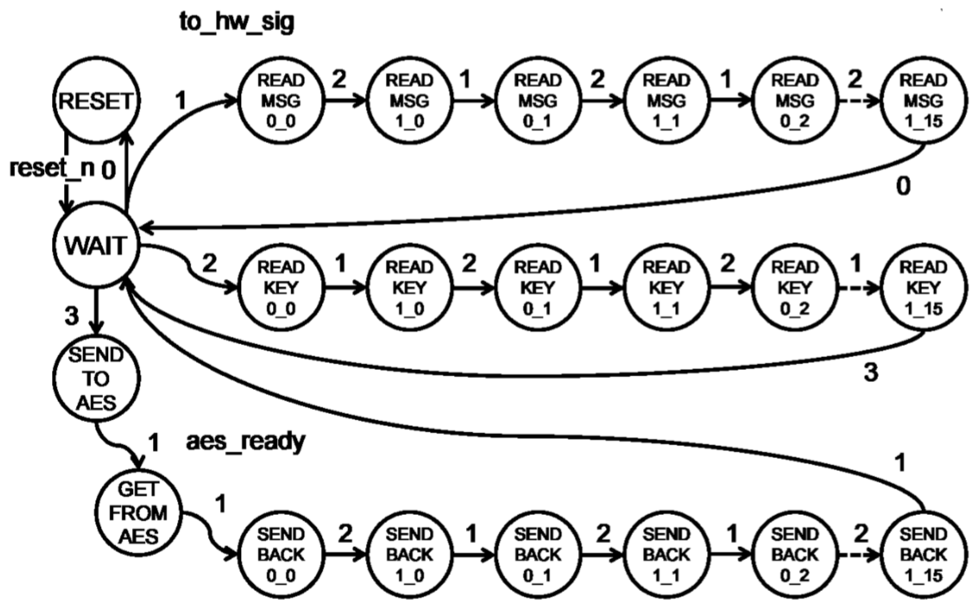


Fig. 1: IO Module Hardware/Software State Diagram

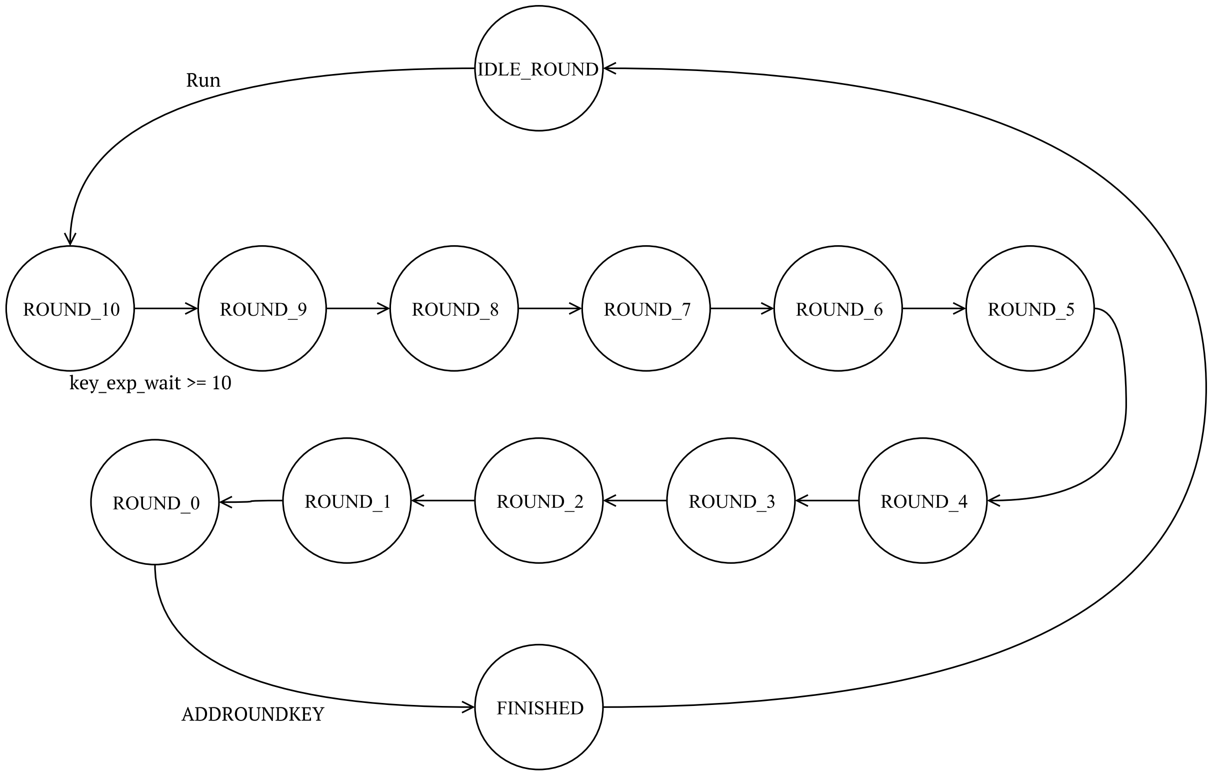


Fig. 2: Decryption Round State Diagram

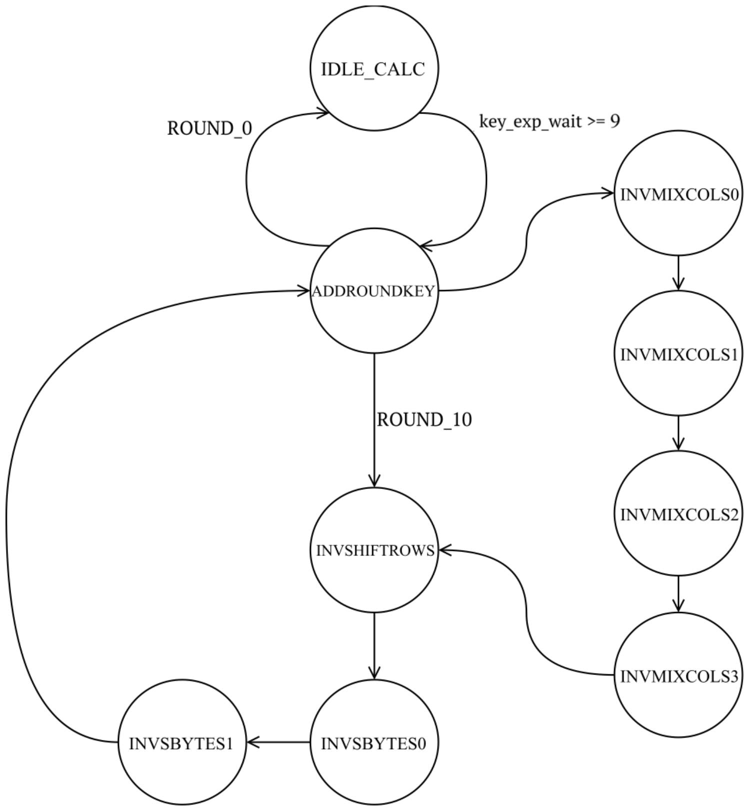
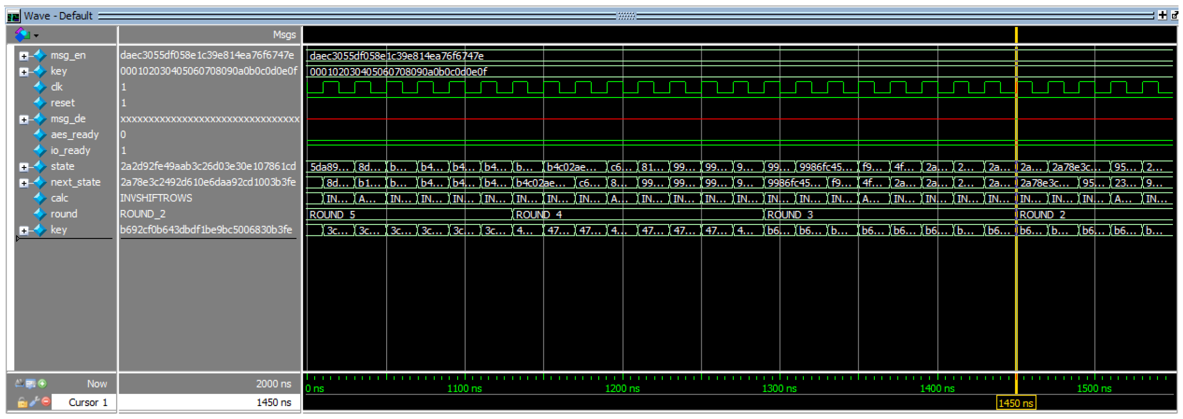
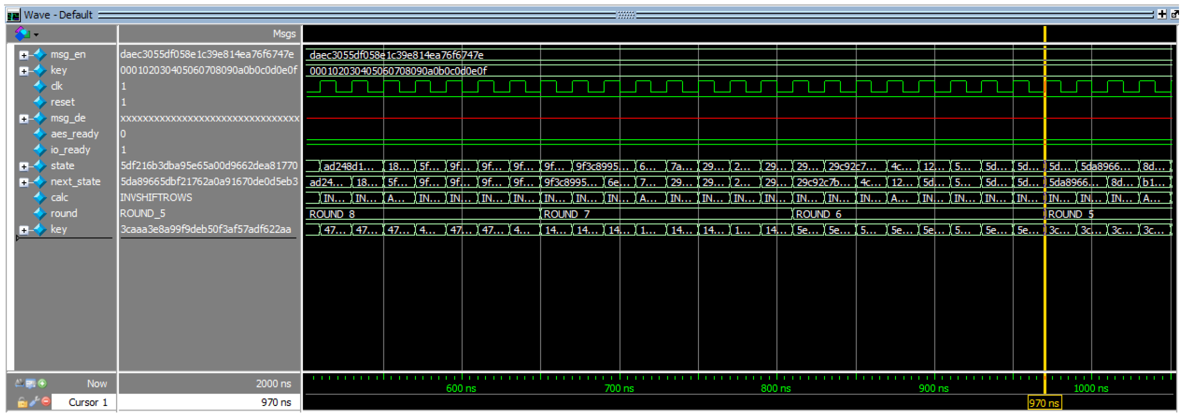


Fig. 3: Calculation State Diagram



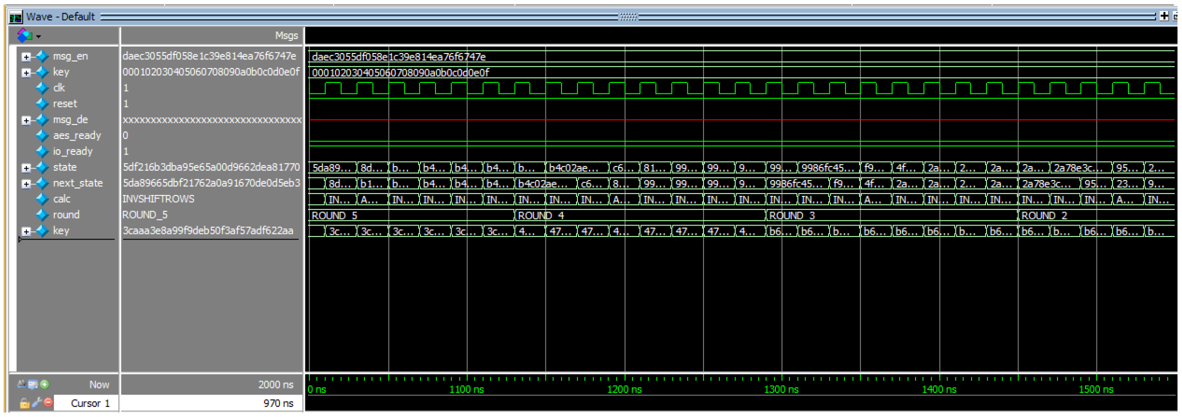
```
Wave - Default
                                                    Msgs
msg_en       daec3055df058e1c39e814ea76f6747e
key          000102030405060708090a0b0c0d0e0f
clk          1
reset        1
msg_de       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
aes_ready    0
io_ready     1
state        2a2d92fe49aab3c26d03e30e107861cd
next_state   2a78e3c2492d610e6daa92cd1003b3fe
calc         INVSHIFTROWS
round        ROUND_2
key          b692cf0b643dbdf1be9bc5006830b3fe

Now          2000 ns
Cursor 1     1450 ns
```

Fig. 4: Simulation 0 - 500ns



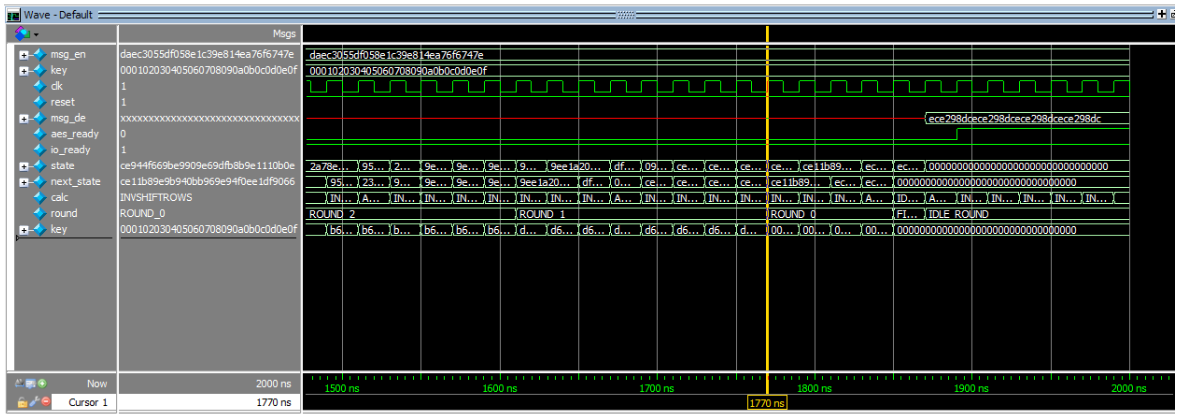
```
Wave - Default
                 Msgs
msg_en       daec3055df058e1c39e814ea76f6747e
key          000102030405060708090a0b0c0d0e0f
clk          1
reset        1
msg_de       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
aes_ready    0
io_ready     1
state        5df216b3dba95e65a00d9662dea81770
next_state   5da89665dbf21762a0a91670de0d5eb3
calc         INVSHIFTROWS
round        ROUND_5
key          3caaa3e8a99f9deb50f3af57adf622aa
Now          2000 ns
Cursor 1     970 ns
```

Fig. 5: Simulation 500 - 1000ns



```
Wave - Default
                 Msgs
msg_en       daec3055df058e1c39e814ea76f6747e
key          000102030405060708090a0b0c0d0e0f
clk          1
reset        1
msg_de       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
aes_ready    0
io_ready     1
state        5df216b3dba95e65a00d9662dea81770
next_state   5da89665dbf21762a0a91670de0d5eb3
calc         INVSHIFTROWS
round        ROUND_5
key          3caaa3e8a99f9deb50f3af57adf622aa
Now          2000 ns
Cursor 1     970 ns
```

Fig. 6: Simulation 1000 - 1500ns



```
Wave - Default
                 Msgs
msg_en       daec3055df058e1c39e814ea76f6747e
key          000102030405060708090a0b0c0d0e0f
clk          1
reset        1
msg_de       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
aes_ready    0
io_ready     1
state        ce944f669be9909e69dfb8b9e1110b0e
next_state   ce11b89e9b940bb969e94f0ee1df9066
calc         INVSHIFTROWS
round        ROUND_0
key          000102030405060708090a0b0c0d0e0f
Now          2000 ns
Cursor 1     1770 ns
```

Fig. 7: Simulation 1500-2000ns

APPENDIX