Introduction

The AES (Advanced Encryption Standard) is the most common symmetric encryption algorithm. In the Lab 9, we realize an AES-128 encryption/decryption on the NIOS 2 soc with FPGA board. The encryption part is done in software while the description part is done in the hardware. We also write benchmarks of the whole progress and we can find that the hardware description is much faster than the software encryption.

1. Summarize the operation of the AES encryption and decryption.

The AES is a symmetry encryption algorithm, as a result, both the sender and the receiver should know the Cipher key. In order to use the AES, you should prepare a Cipher key and the Plaintext, that is the message to be encrypted, and load it to the AES encryption, and then we get the encrypted message called Ciphertext. The receiver who has the Cipher key and the Cyphertext can goes through the reverse algorithm to produce the original meaningful Plaintext message.

1. Description and the operation of the AES encryptor/decryptor.
   1. Software encryption

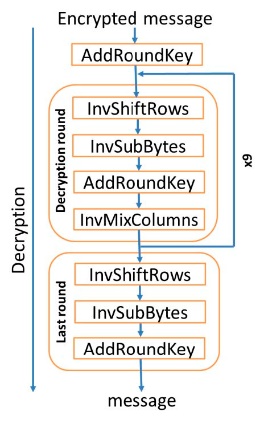
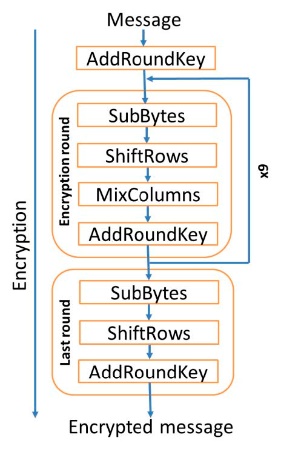
The message is encrypted by a C program running on the NIOS 2. With the JTAG UART module, we can get a scanf function, so we can interact to it with the console to send Plaintext and the Cipher key that we prefer. Once you send a valid message and a key, the main function will call the function CharToHex to convert the strings to the hex values, Then the key expansion function will expand the key to 10 round keys, and then we add the first Round Key with XOR to the message. Then we will loop the following steps for 9 times to encrypted the message totally: execute the SubBytes, ShiftRows, MixColumns and then addRoundKey. The SubBytes function will substitute the value based on the look up table with High four bits and the low 4 bits of the original value. The ShiftRows function will left shift the values in each row based on the number of the row. The MixColumns function will multiply each column with a fix matrix in the GF(2^8) to get the fixed value. After the loop, we call SubBytes, ShiftRows, and addRoundKey again to output the final Ciphertext.

* 1. Hardware description

The Ciphertext will be decrypted by hardware logics written by System Verilog on the FPGA. When we type in the CipherKey and the Ciphertext in the console, the software will write them to the register file that defined in the AES decryption module. Then we initiate the AES module and all the linked module and then define states to decryption step by step. According to the diagram, we should start with the key expansion module to expand 9 round keys. Then we run addRoundKey module to do XOR to the keys. After that we will loop the invSubBytes, InvShiftRows, AddRoundKeys, and InvMixColumns 9 times to completely reverse the encryption. Then We will do InvShiftRows, InvSubBytes and AddRoundKeys again to get the Plaintext again.

* 1. Description of the hardware/software interface

We connect the hardware and software with the module Avalon\_aes\_interface.sv. We create a 32-bit reg file with 16 slot to hold the intermediates. After the software finish the encryption. The key will be stored in the first 4 registers and the encrypted message will be stored in the next 4 registers. After the hardware finish the description, the decrypted message will be store in the next 4 registered. Then we will use the 12 and 13 slot to hold signals that sent through registers, and register 14 will be used to hold START signal that given by the C program. To control the registers, the NIOS 2 will write pointers to sent messages to the target slot.



MIDDLE

Post Lab Question

1. The result of the compile:

|  |  |
| --- | --- |
| LUT |  |
| DSP |  |
| Memory |  |
| Flip-Flop |  |
| Frequency |  |
| Static Power |  |
| Dynamic Power |  |
| Total Power |  |

Q1. 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: The expected is that the hardware decryption is much faster than software encryption because the NIOS have low data performance while the FPGA can process all the bits at the same time through hardware logic.

The result is below:

|  |  |
| --- | --- |
| Software encryption | 0.580383 KB/s |
| Hardware Decryption | 76.923077 KB/s |

Q2. If you wanted to speed up the hardware, what would you do? (Note: restrictions of this lab do not apply to answer this question)

A: We can speed up by modifying the InvMixMatrix part. In the lab, we use 4 states to calculate the inverse mix matrix column by column. As a result, if we can reduce the calculation state, we can save much clock edge to decrypt the message.

Conclusion

The project works correctly on both the weeks, but it was hard to design the decryption states for hardware decryption since there are a lot of modules to design. This lab help me understand the hardware can process data much faster than the software when the date procession can be design in logics. I think we can do some different encryption next time like RSA.