ECE 341 AES Encryption

Chequel McNeil, Addie Wright, Daniel Sciortino

Group #1

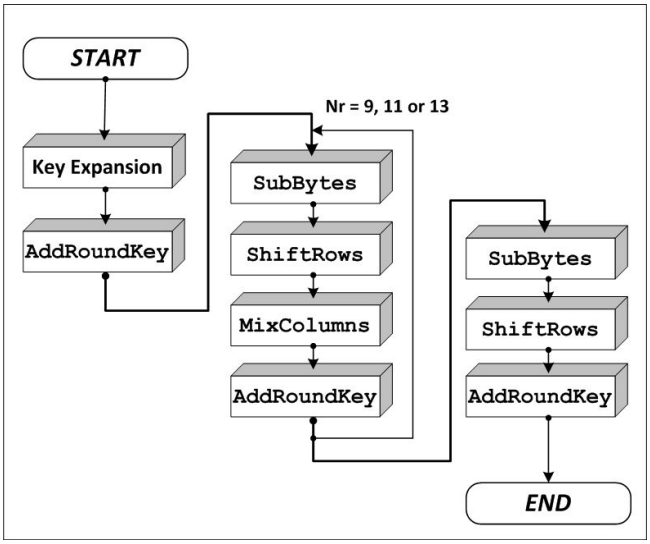
Introduction

Advanced Encryption Standard (AES) became the standard for the Federal Government after Data Encryption Standard started to show its weakness in key sizes. In 1999 The National Institute of Standards and Technology (NIST) started researching the next generation of encryption standards. Once selected the Federal Information Processing Standards made AES the default standard for storing all confidential data. During this project AES was implemented using various design models within the VHDL platform. More details about the designs and actions performed within them are displayed below.

Design

The Advanced Encryption Standard is an iterative process that can be described as having four distinct steps to completing the encryption process. The amount of iterations that it takes to completely encrypt the data is determined by the length of the key that has been chosen. AES can support three different key lengths of 128, 192, or 256. The version that was implemented during this project is a 256 bit key. The longer the key means that the encrypted data will be exponentially more secure. Implementing this version of AES requires the most rounds of the different versions to obtain the most secure protection over the data. Within each of the steps to performing the encryption there are processes that are performed to both the data and the key to make the data less recognizable from the original unencrypted version. The exact details of the processes that occur during each round are explained in detail below.

AES Diagram



To begin the design process we partitioned the entire encryption process into four different categories in order to focus on the processes in a sequential order that the data will travel through. We began by obtaining a flow chart representation of the encryption process. This allowed for a clear direction for the behavioral model.

The behavioral model is intended to be used as the gold standard for any model or version to follow it. This model is implemented on the assumption that everything is ideal. The functionality and performance that the behavioral model is able to output is not realistically achievable with real life applications. When implementing using different techniques such as structural and dataflow the behavioral model will give a good bar to aim for during the design process. It also helps when making different decisions on technologies used to implement the design.

Our approach for the behavioral model was to use a process that would utilize a loop as a vehicle to iterate through a majority of the process. Due to the fact that all of the rounds one through fourteen have the same actions performed for each one a single loop could be used to

represent those rounds. Both the first and the last round of the encryption process has a different set of actions that are being performed and had to be implemented separately from all of the other rounds.

The behavioral model was able to undergo some verification through a few different methods. The model could be tested using testbench functionality of Aldec to confirm correct results. The behavioral model also could be tested by comparing the results of the various steps to the provided AES calculator through paper calculations. We were able to test the encryption process of the behavioral model using the AES calculator and stepping through our code using a hand written approach. With more time to develop the test suite for the model a testbench would be implemented that could fully test the encryption with a properly configured test bench. The key to constructing a valuable testbench is by providing a stimulus that can accurately test many different instances of the model that is being constructed.

After constructing a gold standard behavioral model a structural model could be formulated. The goal of this model is to put the design into terms that could be more easily translated to a physical representation. The structural model is implemented using different components that can be plugged into one another to create the necessary functions. In the case of the AES encryption it is best to break the components according to what makes sense physically in the encryption process. By taking this approach adds flexibility to the model for the future. These components can be used in future versions of the same application or possibly used to implement a different project in the future.

The behavioral model was used as the guide for constructing each part of the structural model. A similar approach was taken to confirm each piece of the process before moving on to

implement the next piece of the encryption. By taking this approach limited the possibility for incorrect output at the end of the design.

Key Expander

The initial step in ciphering the incoming plaintext is expanding the provided key. The key that is taken in can range from 128, to 256 bits. From this the key has to be expanded for each of the different rounds. The first step in extending the key is taking the last 32 bits of the key and rotating them to the left by 8 bits. Next, each 8 bits are grouped together and then substituted from an accompanying substitution box. This rotated and substituted 32 bits is then exclusive or'ed with an the first 32 bits of the array and the first 8 of the round constant. Once the first 32 bits of the new key and the second 32 bits is used to make the second 32 bits of the new key. This process is repeated until a complete round key is created. The process is then repeated. Each round has a unique round key that is based off of the initial supplied key.

For our implementation we took the 256 bit key and conceptually saw it as being vertical matrix. For example the key that we used was "000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f" when we implemetined VHDL to handle the key it was like the following, where each column is its own array.

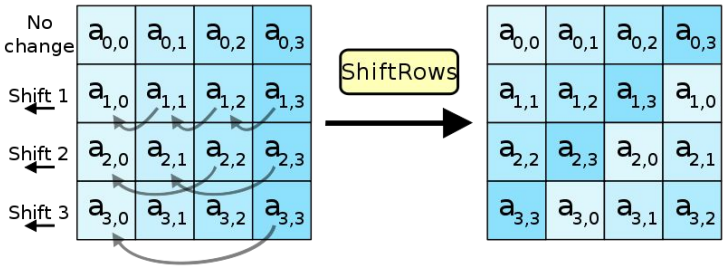
00	05	08	0c	10	14	18	1c
01	06	09	0d	11	15	19	1d
02	07	0a	0e	12	16	1a	1e
03	08	0b	0f	13	17	1b	1f

From this we would take the last 32 bits “1c1d1e1f” and rotate it left 8 bits. This would become “1d1e1f1c”. After that we would take the rotated key and substitute them with the substitution box. This would make the rotated key to “af72c09c”. Once the rotation and the substitution was completed then this would get exclusive or’ed to the first column of the previous expanded array, if this is the first round key it would use the initial key, and the round key. “af72c09c” + “00010203” + “01000000” = “ae73c29f”. This new 32 bits would be the start of the next 256 bit key.

Rounds

For the first round had a simple implementation. This required us to take the initial plaintext array and exclusive or the initial cipher key to it. The test that we used was the string “00112233445566778899aabbccddeeff”. This was then exclusive or’ed to the cipher “000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f”. The output of which was “102030405060708090a0b0c0d0e0f100030205040706090 80b0a0d0c0f0e0”. This initial string of text would then passed from the initial round to the rounds of the AES encryption. It would be here where the diffusion of the plaintext would happen.

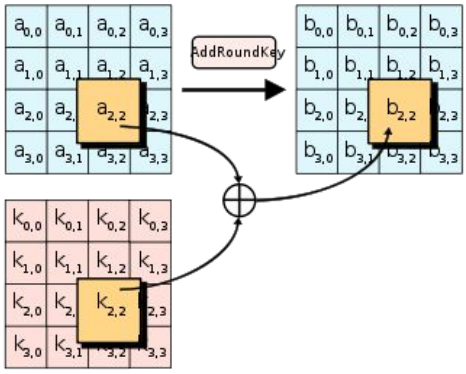
The first step of each round was to substitute 8 bits at a time with the substitution box. This is the same substitution box that is used from the key expansion. This substituted string is then used in another matrix format to shift the rows. The image below illustrates this concept. The first row of the matrix does not shift, the second row shifts by one, the third row shifts by two, and the last row shifts by three.



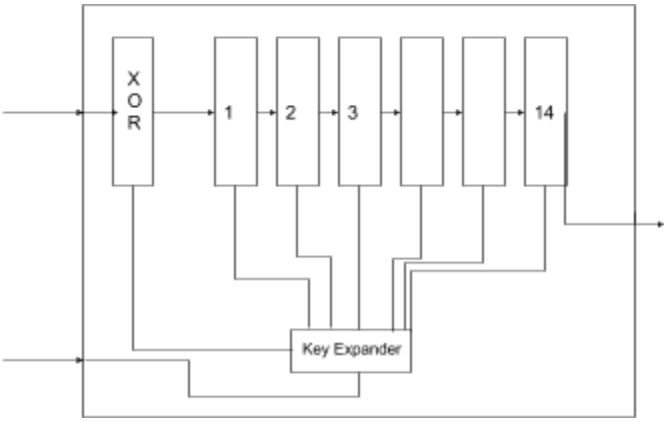
Following the shifting of the rows comes mixing columns. This is the last part of the diffusion of the AES encryption. This step each a column of the matrix is multiplied by a fixed matrix. This step was used in every round except for the last when it goes from the shift

$$\begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \end{bmatrix} \times \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

The last step in the round is exclusive or the round key to to the substituted, shifted, mixed string. This process would repeat any where from 10 to 14 times until the ciphered text would be outputted.



The encryption condensed into a single block would look like the the block diagram below. This would have an input of plaintext and key and would output the would be the ciphertext.



Decryption

Once the encryption was done the next step was to decrypt the ciphered text. This would allow us to verify the AES encryption works in both directions. Due to needing more time to understand the AES encryption we were unable to to start decryption.

Testbench

Part of the verification process for the VHDL implementation of the AES was to do a testbench. This was to show that the encryption was working as expected and the logic was

correct. Do to a time constraint we were not able to verify the functionality of the code with a testbench.

Conclusion

Though the Advanced Encryption Standard has been the federal government standard for the more than a decade it has a very big role in securing information. The benefit of the encryption can be seen by the implementation in both software and hardware. Intel has an implementation in its processor family since 2010, and Advanced Micro Devices has had it since Late 2011. With many different hypnotized attacks, but none being practical AES remains to be a standard for encrypting data.