International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-1 Issue-5, June 2012

Image Encryption and Decryption using AES

Manoj. B, Manjula N Harihar

Abstract— In today's world most of the communication is done using electronic media. Data Security is widely used to ensure security in communication, data storage and transmission. We have Advanced Encryption Standard (AES) which is accepted as a symmetric cryptography standard for transferring block of data securely. The available AES algorithm is used for text data and it is also suitable for image encryption and decryption to protect the confidential image data from an unauthorized access. This project proposes a method in which the image data is an input to AES Encryption to obtain the encrypted image, and the encrypted image is the input to AES Decryption to get the original image. In this paper, we implement the 128 bit AES for image encryption and decryption which is synthesized and simulated on FPGA family of Spartan-6 (XC6SLX25) using Xilinx ISE 12.4 tool in Very high speed integrated circuit Hardware Description Language (VHDL) and shall be verified with the help of its simulation result.

Index Terms— Cryptography, AES, Image Encryption, Decryption, FPGA, S-box, Cipher Text, NIST, FIPS.

I. INTRODUCTION

Computer has become an essential device now a day. The main use of computer is to store data and send the data from one location to other. The information that is shared must be transported in a secure manner. So to avoid such situations data may be encrypted to some formats that is unreadable by an unauthorized person. Cryptography is the science of information security which has become a very critical aspect of modern computing systems to secure the data transmission and storage.

The exchange of digital data in cryptography results in different algorithm that can be classified into two cryptographic mechanisms: symmetric key in which same key is used for encryption and decryption and asymmetric key in which different keys are used for encryption and decryptions. Symmetric key algorithms are much faster and easier to implement and generally requires less processing power when compared with asymmetric key algorithms.

The National Institute of Standards and Technology (NIST) declared Rijndael algorithm as the Advanced Encryption Standard (AES) in October 2000 [1]. The Advanced Encryption Standard specifies a Federal Information Processing Standard (FIPS) that has approved cryptographic algorithm which is used to protect sensitive information. The AES algorithm is a symmetric key algorithm that encrypts and decrypts information. Encryption process converts an original message (plain text) into encrypted message (cipher text). Decryption process is to convert the cipher text message back to plain text so that it can be readily understood.

AES algorithm is not only for the text data, it can applied for the images, usually image processing deals with a image, which is composed of many image points. The image points, namely pixels, spatial co-ordinates that indicate the position of the points in the image and intensity values. The applications of the image processing have been commonly found in the Military communication, Forensics, Robotics, Intelligent systems etc. In this paper, we implement the AES algorithm which is an efficient scheme for both hardware and software implementation.

II. AES ALGORITHM

AES comes in three favors, namely AES - 128, AES - 192, and AES-256, with the number in each case representing the size (in bits) of the key used. All the modes are done in 10, 12 or 14 round depends on the size of the block and the key length chosen. AES merely allows a 128 bit data length that can be divided into four basic operation blocks. These blocks operate on array of bytes and organized as a 4*4 matrix that is called the state. The algorithm begins with an Add round key stage followed by nine rounds of four stages and a tenth round of three stages which applies for both encryption and decryption algorithm [1] [2] [4].

These rounds are governed by the following four stages:

- Substitute Bytes
- Shift rows
- Mix columns
- Add round key

The tenth round Mix columns stage is not included. The first nine rounds of the decryption algorithm are governed by the following four stages:

- Inverse Shift rows
- Inverse Substitute Bytes
- Add round key
- Inverse Mix columns

Again the tenth round Inverse Mix columns stage is not included. The Overall flow of the encryption and decryption algorithm of the AES algorithm is show in Figure 1.

Manuscript published on 30 June 2012.
***** Correspondence Author (s)
Manoj B*, Department of Electronics and Communication, School of Engineering and Technology, Jain University, Bangalore, India, (e-mail: manojb.jgi@gmail.com).
Manjula. N. Harihar, Department of Electronics and Communication, School of Engineering and Technology, Jain University, Bangalore, India, (e-mail: manjulaharihar@gmail.com).

Substitute Bytes: Is a non linear byte substitution, using a substation table (S-box) each byte from the input state is replaced by another byte. The substitution is invertible and is constructed by the composition of two transformations as described below [2] [8]. The substitute bytes operation is as shown in Figure 3.

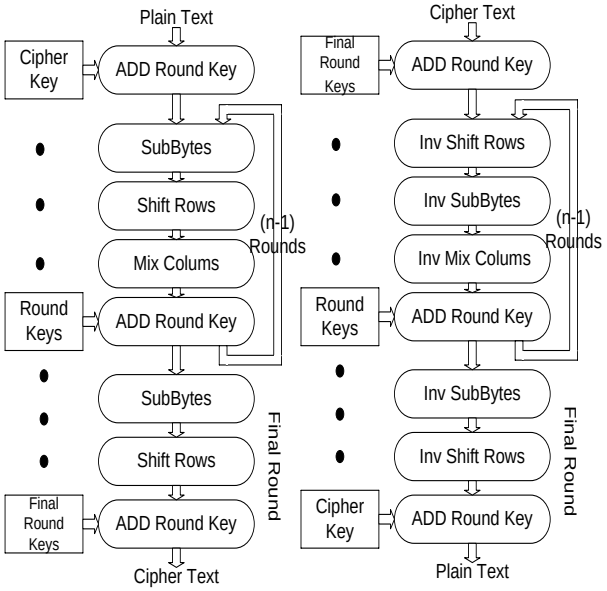


Figure 1: Design flow of AES Algorithm (a) Encryption Process (b) Decryption Process

1. The state bytes are constructed by multiplicative inverse in the finite field GF (2^8) that is used in the AES. This field is derived using the following irreducible polynomial of degree 8:

$$m(x) = x^8+x^4+x^3+x+1. \quad (1)$$

2. Affine transformation is applied on the result of above statement. The fixed matrix and the fixed vector over GF (2) in the affine transformation are shown in the Figure 2.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Figure 2: Affine transformation of Substitute Bytes

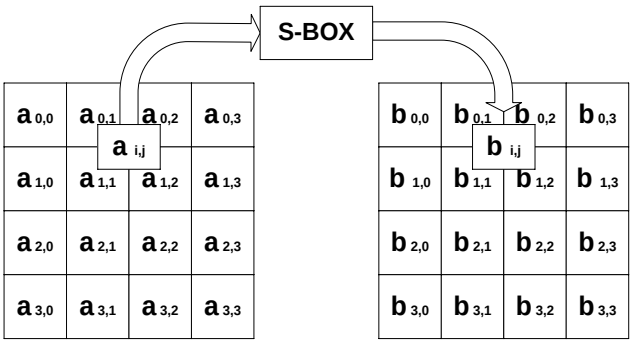


Figure 3: Substitute Bytes Operation

Inverse Substitute Bytes: Is the reverse operation of the Substitute Bytes transformation, in which the inverse S-box is applied to each byte of the State. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF (2^8).

Shift rows: In the Shift Rows transformation, the first row of the state array remains unchanged. The bytes in the second, third and forth rows are cyclically shifted by one, two and three bytes to the left, respectively as shown in Figure 4.

Inverse Shift rows: Is the inverse of the shift rows, the first row of the state array remains unchanged. The bytes in the second, third and forth rows are cyclically shifted by one, two and three bytes to the right, respectively.

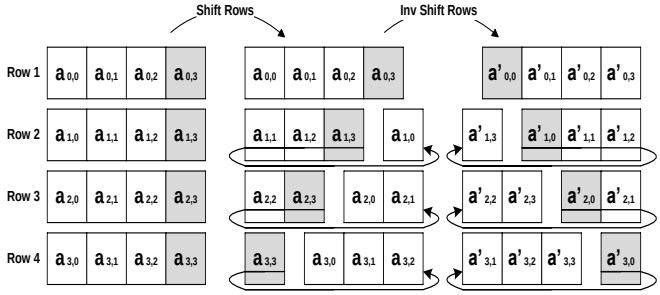


Figure 4: Shift Rows Operation

Mix columns: In the Mix Columns transformation, every column of the state array is considered as polynomial over GF (2^8). After multiplying modulo x^4+1 with a fixed polynomial a(x) [2], the operation of MixColumn is as shown in Figure 5.

$$a(x)=03*x^3 + 01*x^2 + 01*x + 02 \quad (2)$$

The result is the corresponding column of the output state is as shown in Figure 5. As it can be noticed this operation requires multiplication by 'two' and 'three' that is relatively simple shift operation in hardware.

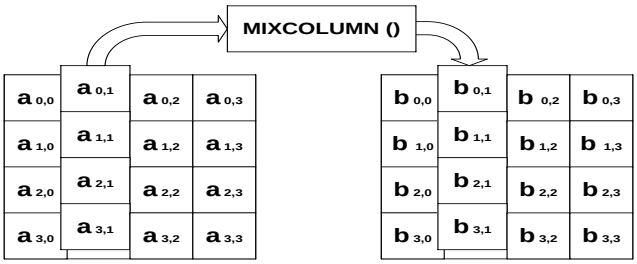


Figure 5: MixColumn Operation

Inverse Mix columns: In the Inverse Mix Columns transformation, every column of the state array is considered a polynomial over GF (2^8). After multiplying modulo x^4+1 with a fixed polynomial b(x),

$$b(x)=0B*x^3 + 0D*x^2 + 09*x + 0E \quad (3)$$

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The result is the corresponding column of the output state. As it not so straightforward hardware implementation as Mix column, so if we compare both, InvMixCol requires more logic resources for implementation.

Addroundkey: The AddRoundKey operation is as shown in Figure 6, which is a simple XOR operation between the State and the Round Key. The Round Key is derived from the Cipher key by means of key schedule process. The State and Round Key are of the same size and to obtain the next State an XOR operation is done per element:

$$b(i, j) = a(i, j) \oplus k(i, j) \qquad (4)$$

Where *a* is the current State, *b* the next State and *k the* round key.

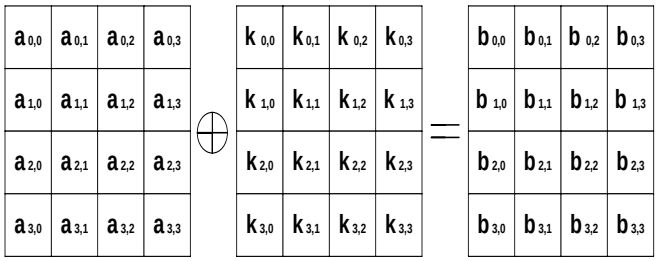


Figure 6: AddRound Key Operation

There are three steps, in each Key schedule round [6].

Keyrotate: The function Keyrotate takes a four-byte word and rotates one byte to the left.

Keysubytes: The Keysubytes operation takes four-byte input word by substituting each byte in the input to another byte according to the S-Box.

KeyRcon: The first byte of a word is XORed with the round constant. Each value of the Rcon table is a member of the Rinjdael finite field.

Add round key is same for the both encryption and decryption.

III. IMPLEMENTATION

A. *AES Encryption*

To implement the AES algorithm using VHDL coding in the Xilinx 12.4 [3], we proceed with the encryption and decryption with 128 bits. The AES Encryption block is as shown in Figure 7, the encryption parameters are the input plaintext, the key of size key size and the output cipher text. First, we have to map the 16 byte input plaintext in the correct order to the 4*4 byte state, calculate the number of rounds based on the key Size and expand the key using our key schedule. At round one plaintext and key is XORed, the remaining nine rounds which has to apply all four operations: Substitute Bytes, Shift rows, Mix columns, Add round key.

The tenth round Mix columns stage is not included and round key was generated during each iteration [7]. Here simple XOR of each byte of the key with the respective byte of the state is done to get cipher text of the Encryption algorithm.

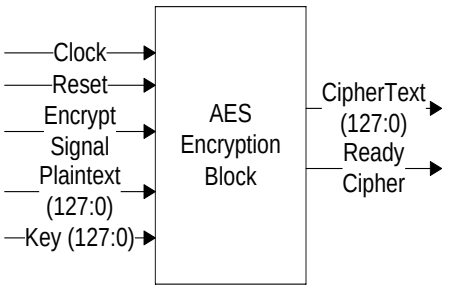


Figure 7: AES Encryption Block

B. *AES Decryption*

For AES Decryption, the same encryption process occurs simply in reverse order. The decryption block is as shown in Figure 8, the encryption parameters are the input cipher text, the key and the output plaintext should be same as encryption input. In decryption the key schedule remains the same; the only operations we need to implement are the Inverse subBytes, shiftRows and mixColumns, while addRoundKey stays the same.

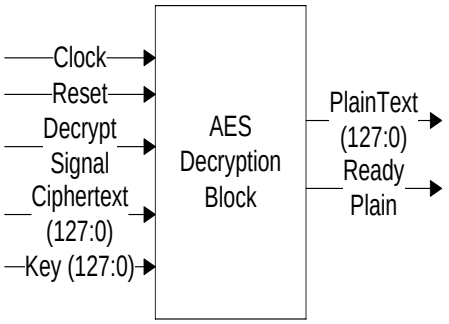


Figure 8: AES Decryption Block

C. *Image Encryption and Decryption Block*

To Encrypt, Decrypt the image pixel, 7- bits is the input and output which communicates with the 128 bit AES algorithm. We have to shift the 7 bit input to 128 bit register and feed it to the AES Encryption algorithm to get the Encrypted 128 bit data and from the 128bit register shift 7 bit data to Image per clock to get the encrypted image. To get back the original image feed the Encrypted image to the AES Decryption algorithm is as shown in Figure 9.

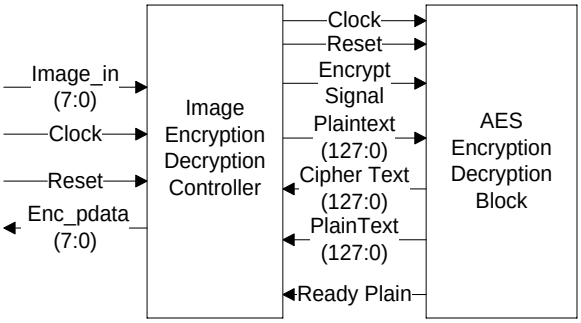


Figure 9: Image Encryption and Decryption Block

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

IV. RESULTS

A. Simulation Results

The waveforms generated for the Image Encryption and Decryption Process is shown in the figure below.

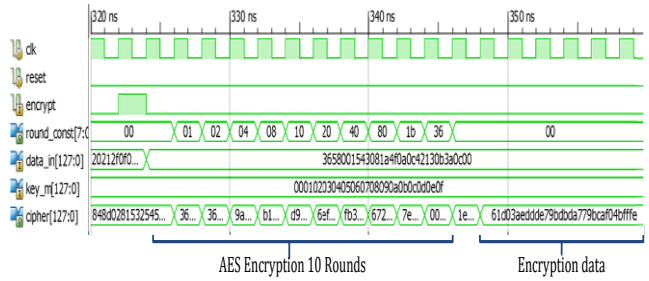


Figure 10: Simulation Waveform of Encryption

Encryption Process (128 bit)
Plain Text: 3658001543081a4f0a0c42130b3a0c00.
Key: 000102030405060708090a0b0c0d0e0f.
Cipher Text: 61d03aeddde79bdbda779bcaf0abfffe.

Decryption Process (128 bit)
Cipher Text: 61d03aeddde79bdbda779bcaf0abfffe.
Key: 13111d7fe3944a17f307a78b4d2b30c5.
Plain Text: 3658001543081a4f0a0c42130b3a0c00.

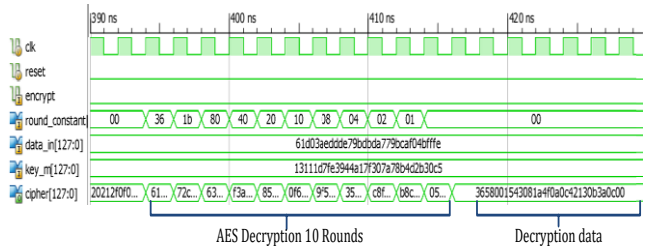


Figure 11: Simulation Waveform of Decryption

The Synthesis report of Image Encryption and Decryption are illustrated in table below. The device utilization report generated from the ISE 12.4 for FPGA Spartan-6 (XC6SLX25) is described in the percentages.

The core performance (Mbits/sec) can be calculated as [7]

Performance = fmax *(128 bits/block) / (n clocks)

For AES Encryption and Decryption, n= 24 clocks: 882.464Mbits/sec.
For Image Encryption and Decryption, n= 84 clocks: 252.132Mbits/sec.

Table 1: Synthesis Utilization

Slice Logic Utilization	
Number of Slice Registers	1658 out of 30064 5%
Number of Slice LUTs	3456 out of 15032 22%
Slice Logic Distribution	
Number of LUT Flip Flop pairs used	4360
Number with an unused Flip Flop	2702 out of 4360 61%
Number with an unused LUT	904 out of 4360 20
Number of fully used LUT FF pairs	754 out of 4360 17%
Number of unique control sets	40
IO Utilization	
Number of bonded IOBs	21 out of 186 11%
Number of BUFG/BUFGCTRLs	1 out of 16 6%
Maximum Frequency (fmax)	165.462MHz
Image Encryption and Decryption Throughput	252.132Mbits/sec
AES Encryption and Decryption Throughput	882.464Mbits/sec

The waveform of On-Chip Power over Vccint – (Typical, 26C), On-Chip Typical vs Maximum Power, On-Chip Power over Temperature– (Typical, 1.2V) are shown below for the Image Encryption and Decryption using AES in FPGA

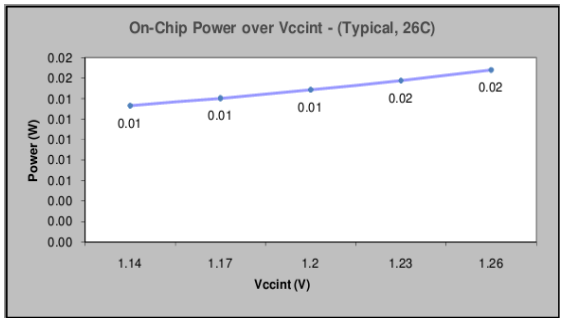


Figure 12: On-Chip Power over Vccint

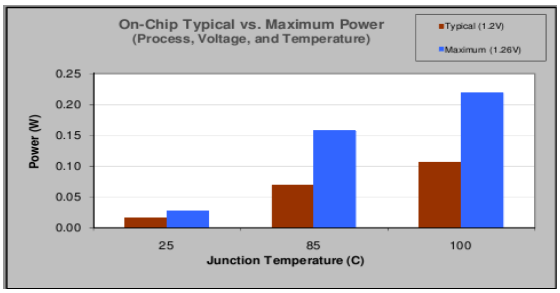


Figure 13: On-Chip Typical vs Maximum Power

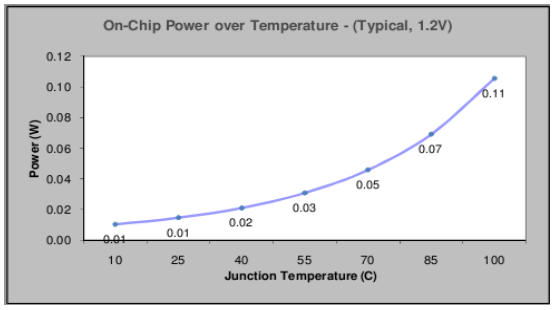
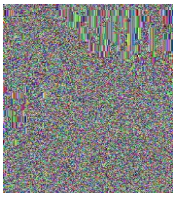


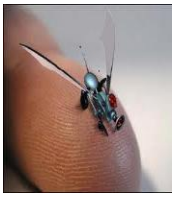
Figure 14: On-Chip Power over Temperature

The Image which is encrypted and decrypted is shown below,
Plain Text Image Cipher Text Image Plain Text Image

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)


Input Image


Encrypted Image


Decrypted Image

The time taken to simulate for the different images (.bmp) is as shown in the table [5].

Table 2: Time taken for Different Images

Image Size	Image Size on Disk	Time taken to Encrypt and Decrypt
256*256	66KB	0.707322ms
512*512	258KB	2.796286ms
1024*1024	3.07MB	167.11ms

V. CONCLUSION

In this paper, Image Encryption and Decryption using AES is designed and implemented to protect the confidential image data from an unauthorized access. A Successful implementation of AES algorithm is one of the best encryption and decryption standard available in market.

It helps to explore the path to implement such an algorithm using VHDL code that is synthesized and simulated using the ISE 12.4 in Xilinx Family Spartan-6 (XC6SLX25). The Maximum Frequency achieved from the design is 165.462MHz and the throughput reaches the value of 252.132Mbit/sec for Image Encryption and Decryption.

REFERENCES

[1] National Institute of Standards and Technology, "Federal Information Processing Standard Publication 197, the Advanced Encryption Standard (AES)," Nov. 2001.
[2] William Stalling, Cryptography and Network Security: Principles and Practices, Principles and Practices, 4th ed. Prentice Hall, 2006.
[3] Charles H Roth, Jr. Digital Systems Design Using VHDL, Thomson, India Edition 2007.
[4] Atul Kahate, Cryptography and Network Security, Second Edition, Tata McGraw-Hill Edition 2008.
[5] Abdulkarim Amer Shtewi, Bahaa Eldin M. Hasan, Abd El Fatah .A. Hegazy "An Efficient Modified Advanced Encryption Standard (MAES) Adapted for Image Cryptosystems" IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.2,pp.226-232 February 2010.
[6] P.Karthigaikumar, Soumiya Rasheed "Simulation of Image Encryption using AES Algorithm" IJCA Special Issue on "Computational Science - New Dimensions & Perspectives" NCCSE, pp166-172, 2011.
[7] Mr. Atul M. Borkar, Dr. R. V. Kshirsagar, Mrs. M. V. Vyawahare "FPGA Implementation of AES Algorithm" IEEE, pp.401-405, 2011.
[8] Xinmiao Zhang, Keshab K. Parhi, Fellow, "High-Speed VLSI Architectures for the AES Algorithm" IEEE Transactions on vlsi systems, vol. 12, no. 9, pp.957-966, September 2004.

Manoj B Obtained his Bachelor degree in Electronic and Communication Engineering from Vemana Institute of Technology, VTU, Bangalore in 2009 and now pursuing M.tech (SP&VLSI) in Electronic and Communication Engineering, Jain University, Bangalore. My research interests include efficient VLSI, Image Processing, Nano Technology and digital signal processing systems.

This paper is dedicated to my parents for their love, endless support and encouragement. I would also thank my batch meats, friends and constant support and guidance from S.Subramanian, Prabul Kanth P M and Sanjeeva nayaka.

Manjula. N. Harihar is a Assistant Professor in the Department of Electronics and Communication Engineering, School of Engineering, Jain University, Bangalore. She obtained her Bachelor degree in Electronics and Communication Engineering from S.T.J Institute of Technology, Ranebennur and Master degree in Communication Systems from P.D.A College of Engineering, Gulbarga, Karnataka, India. She is pursuing Ph.D in Electronics and Communication Engineering, Jain University, Bangalore. Her research interest includes Image Processing, VLSI, Neural Networks and Wireless Communication.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)