ISSN 2320-9984 03 >
9 772320 998015

Volume 1, Issue 3, October 2013 ISSN: 2320-9984 (Online)

International Journal of Modern Engineering & Management Research

Website: www.ijmemr.org

FPGA Implementation of Enhanced AES Algorithm for Secret Communication

Bhupendra Badoniya
Research Scholar
Department of Electronics and Communication Engg.
Shriram Institute of Technology, Jabalpur
Engineering, Affiliated to RGPV University, Bhopal
Email : bhupendra.badoniya@gmail.com

Prof. H. Siddharth
Asst. Professor
Department of Electronics and Communication Engg.
Takshshila Institute of Engineering & Technology,
Jabalpur (M.P.) [INDIA]
Email: siddhahari@gmail.com

Prof. Ravi Mohan
Professor & Guide,
Department of Electronics & Communication Engg,
Shri Ram Institute of Science and Technology,
Jabalpur, (M.P.) [INDIA]
Email: ravimohan7677@yahoo.co.in

Prof. Sumit Sharma
Professor & Head of the Department
Department of Electronics & Communication Engg,
Shri Ram Institute of Science and Technology,
Jabalpur, (M.P.) [INDIA]
Email: sharma.sumit3@gmail.com

Abstract**—***Cryptographic technology is an important way to ensure information security, and is the key to information safety. Among all kinds of cryptographic algorithms, Advanced Encryption Standard Algorithm (AES) is preferred as it offers high security, efficiency, convenient usage, flexibility, and comprehensive performance. The aim of this project is to communicate the data secretly using AES Algorithm. We first send the data (plain text) and the key which is of 128 bit into the encryption process. The output of this process will be used to obtain cipher text. This cipher text is then fed into the decryption process and the process is reversed with the same key to get the data (plain text) as output. Since we add the key and shuffle the data it is very hard for the unknown person to find out the original data. For each key there will be a change in the cipher text and so the person has to know the key in order to find out the original data. Thus sensitive information is protected across insecure networks so that even if the data is leaked, it is unusable. AES Algorithm is implemented using FPGA as it gives quicker and more customizable solution. VHDL language is used for coding and all transformations are simulated using an iterative design approach for minimizing the hardware consumption.*

Keywords**—***AES (Encryption & Decryption) Algorithm, FPGA Implementation*

1. INTRODUCTION

AES is a cryptographic algorithm that is used to protect electronic data. The Advanced Encryption Standard (AES) is a National Institute of Standards and Technology specification for the encryption of electronic data. It is expected to become the accepted means of encrypting digital information, including financial, telecommunications, and government data. The AES algorithm is a symmetric block cipher that can encrypt, (encipher), and decrypt, (decipher), information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128,192 and 256 bits to encrypt and decrypt data in the blocks of bits. AES cipher is specified as a number of

repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext.

Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to AES is based on a design principle known as a Substitution permutation networ. Unlike its predecessor, DES, AES does not use a Feistel network. AES operates on a 4 x 4 array of bytes called state in a matrix form. The algorithm consists of performing four different simple operation. These operations are: Sub Bytes, Shift Rows, Mix Columns and Add Round Key.

2. ENCRYPTION (CIPHER GENERATION) USING AES ALGORITHM

AES operates on a 4x4 array of bytes (referred to as "state"). The algorithm consists of performing 4 different operations.

Encryption process Block Diagram

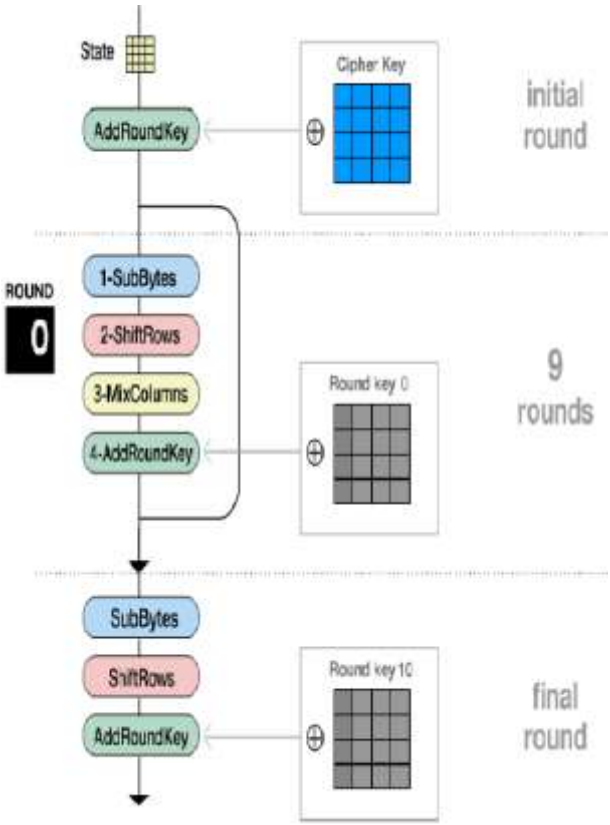

Figure 1. AES Encryption process

1. SubBytes( ) Transformation

The **SubBytes( )** transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). S-box is Invertible and used for decryption purpose.

2. ShiftRows( ) Transformation

In the ShiftRows() transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, $r = 0$, is not shifted. The shift value shift (r, Nb) depends on the row number, r, as follows (recall that Nb = 4):

shift(1,4) =1; shift(2,4) = 2 ; shift(3,4) = 3

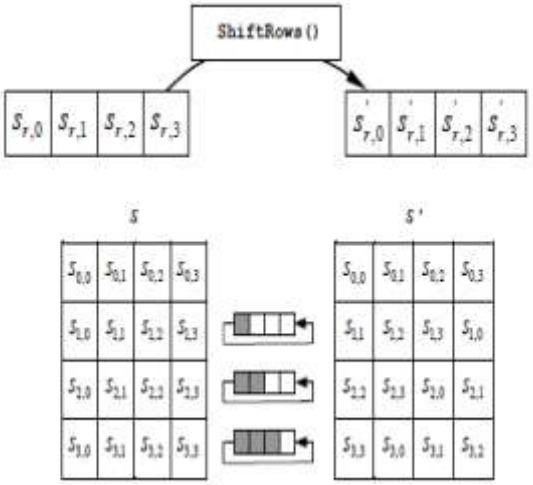

Figure 2 ShiftRows( ) cyclically shifts the last three rows in the state

3. MixColumns( ) Transformation

The MixColumns() transformation operates on the State column-by-column, treating each column as a four-term polynomial. In the MixColumns step, each column of the state is multiplied with a fixed polynomial *a(x)*.

In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes.

Columns are considered with fixed polynomial a(x), given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}. \quad (1)$$

Let

$$s'(x) = a(x) \otimes s(x):$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

for 0 ≤c < Nb MixColumns( ) operates on the state column-by-column (2)

4. *AddRoundKey( ) Transformation*

In the AddRoundKey() transformation, a Round Key is added to the output of MixColumn operation (state) by a simple bitwise XOR operation. For each round of operation, separate key is generated using Key Expansion.

Key Expansion

Round keys are derived from the cipher key using Rijndael's key schedul. The AES algorithm takes the Cipher Key, *K*, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of *Nb*(Nr + 1) words. The expansion of the input key into the key schedule proceeds as per the functions Rotword(), Subword(), Rcon [i/Nk], Xor operations.

3. DECRYPTION (INVERSE CIPHER GENERATION) USING AES ALGORITHM

The process and idea behind the decryption algorithm or inverse cipher is almost identical to that of the Cipher.

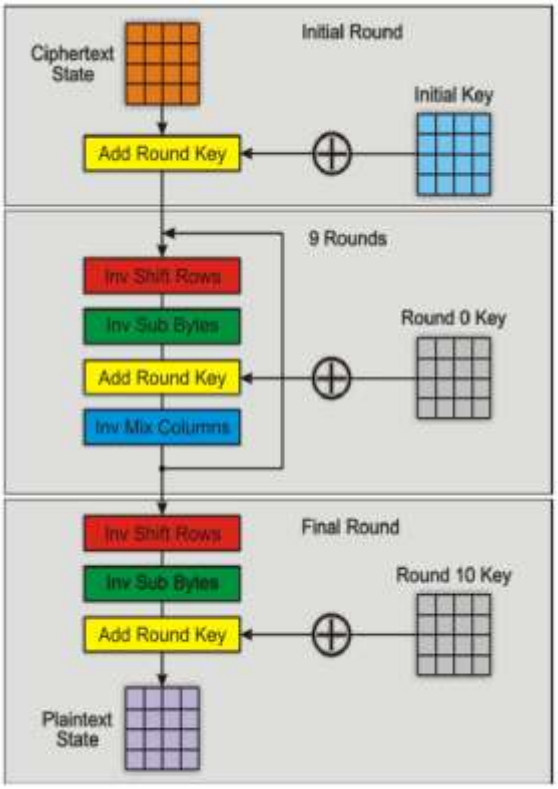

Figure 3 AES Decryption Process

- Only exception is of employing quite the reverse transformations in a different sequence as, Invshiftrows(), InvSubBytes(), Invaddrounkey(), Invmixcols() þ
- The two properties that allow for this Equivalent Inverse Cipher are as follows:
- The SubBytes() and ShiftRows() transformations commute and the same is true for their inverses, InvSubBytes() and InvShiftRows.
- The column mixing operations – MixColumns() and InvMixColumns () – are linear with respect to the column input, which means,
- InvMixColumns(state XOR Round Key) = InvMixColumns(state) XOR InvMixColumns(Round Key).

These properties of the AES algorithm allow for an Equivalent Inverse Cipher that has the same sequence of transformations as the

Cipher (with the transformations replaced by their inverses) that can greatly reduce the algorithm design burden and allows for reusability. This is accomplished with a change in the key schedule by taking invmixcols to each round key.

4. SIMULATION WAVEFORM FOR ENCRYPTION

Encryption Process(Cipher)

AES Block Length/Plain Text=128 bits(Nb=4)

Key Length=128 bit(Nk=4)

No. of rounds=10

Plain Text:00112233445566778899aabbccddeeff

Key:000102030405060708090a0b0c0d0e0f

Cipher text:69c4e0d86a7b0430d8cdb78070b4c55a

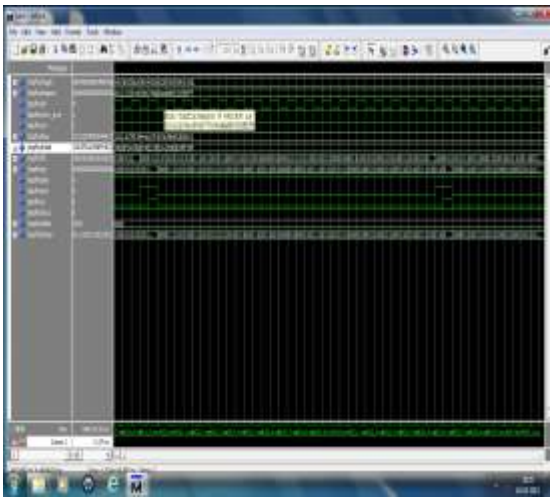

Figure 4 : Simulation Waveform for Encryption

5. SIMULATION WAVEFORM FOR DECRYPTION

Decryption Process (Decipher)

AES Block Length/Plain Text=128 bits (Nb=4)

Key Length=128 bit(Nk=4)

No. of rounds=10

Cipher Text:69c4e0d86a7b0430d8cdb78070b4c55a

Key:000102030405060708090a0b0c0d0e0f

Plain text:00112233445566778899aabbccddeeff

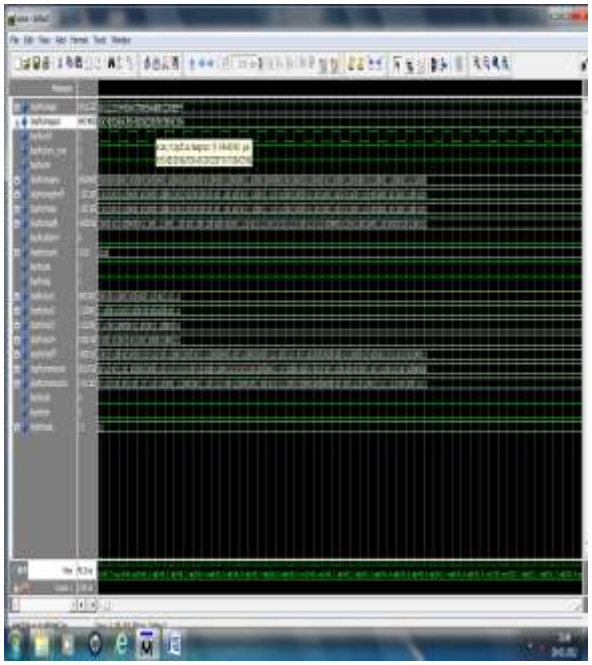

Figure 5 : Simulation Waveform for Decryption

6. HARDWARE IMPLEMENTATION

The performance and safety of hardware implementation is much better than software. Optimized VHDL code is developed for 128 bit plaintext and 128 bit cipher key for encryption and decryption. Synthesis is done using Xilinx ISE 10.1 and implemented on a high performance and cost effective SPARTAN 3AN FPGA Kit.

7. CONCLUSION

Through this project I have tried to implement secure data communication between any two users based on the realization of advanced Symmetric-key Cryptographic algorithm called Advanced Encryption Standard (AES) on an FPGA based processor.

Starting with the selection of highly-structured and immensely secure Advanced Encryption Standard Algorithm, and making suitable modifications in the AES algorithm to improve the Speed and throughput of instruction execution, which is designed selectively in a superior Hardware Description Language—VHDL, simulated with a powerful debugging tool from Mentor Graphics, ModelSim XE III 6.3C, and then synthesized in Xilinx ISE 10.1 with Speed as an

optimization goal aimed at reducing the unrelated logic and improving the maximum clock-rate particularly targeted on a low cost, high speed and highly efficient architectural FPGA chip SPARTAN-3AN, I try to achieve the proven tremendous performance and cost-effective parameters of the hardware implementation of the Advanced Encryption Algorithm (AES) that suits the greatest security demands from a wide variety of users and applications.

REFERENCES :

[1] Atul M Borkar, Dr R.V.Kshirsagar, M.V.Vyawahare,(2011) "*FPGA Implementation of AES Algorithm*".

[2] ChangYuan Yan, RuoHui Xiao (2011) "*Study of Block Algorithms Implement on Hardware in Information Security System*".

[3] Atul Kahate, (2003)"*Cryptography and network security* ".

[4] http://csrc.nist.gov/publications/fips/fips197/fips-197.pd

[5] http://en.wikipedia.org/wiki/Advanced_Encryption_Standar

[6] Joan Daemen, Vincent Rijmen,(2002) "The *Design of Rijndael: AES - The Advanced Encryption Standard*".

[7] Neil H.E. Weste and Kamran Eshraghian, (2000) "*Principles of CMOS VLSI Design*".

[8] Nicolas Courtois, Josef Pieprzyk, (2002) "*Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*".

[9] O P Verma, Ritu Agarwal, Dhiraj Dafouti, Shoba Tyagi,(2011) "*Performance Analysis of Data Encryption Algorithms*".