Journal of Faculty of Engineering & Technology

journal homepage: www.pu.edu.pk/journals/index.php/jfet/index

FPGA IMPLEMENTATION OF ADVANCE ENCRYPTION STANDARD WITH SINGLE KEY

K. Javed[*], A. Sultan, S. Mehmood

University of Engineering and Technology Taxila, Pakistan

Abstract

Advanced Encryption Standard (AES), is known as most secured encryption standard now a days. Many researchers have implemented it in different languages like java, C and C++ with different algorithms. Recently the AES 128-bit has been implemented using Verilog on FPGA with equipped key being encrypted along with data input in whole process. In this paper the AES 128-bit encryption and decryption process with key which is only used for data input and is not encrypted throughout the encryption/decryption process. Results are same but our algorithm is slightly faster because only data is encrypted in the process of encryption, thus process time and area is optimized.

Keywords: Cryptography, Plaintext, Cipher Text, Decryption, Special Key, Encryption.

1. Introduction

In January1997, the National Institute of Standard Technology planned to move on new technology for securing secret information and confidentiality to supersede Data Encryption Standard []. Advanced Encryption Standard (AES), is known as most secured encryption standard now a day. Many researchers have implemented it in different languages like java, C and C++ with different algorithms [].The new standard called as Advanced Encryption Standard (AES) possesses the following attributes;

i. 128-bit block cipher data with selection of three key sizes of 128-bit,192-bit and 256-bit respectively[3].
ii. A public and plaint design.
iii. At least as secure as two-key triple-DES.
iv. Available royalty-free world wide[3].

This standard is unbreakable. One can trust this Standard by securing secret information like credit cards, bank accounts and credentials. AES is now days used commercially in which internet security standard IPsec, TLS, IEEE 802.11i, Skype and several security products are taking advantage of this standard[4].

The AES algorithm consists of encryption and decryption module. The plain text of 128-bit for this algorithm is given to AES encryption module along with 128-bit key. AES encryption module encrypted the 128-bit plaintext to 128-bit chipper text or secret code. Encryption is a

[*] Corresponding Author: kamran.javed@uettaxila.edu.pk

processes of encoding the messages or baseband signals in digital communications it also can be state as the operation of encoding the storage data that means first encoder encode the input data before storage in such way that the information can be read only by authorize people.

Decryption is the reverse process of encryption. In this process, unreadable text converts into readable plaintext. AES is used for security purpose in;

i. Smart card, identification card and credit card
ii. Securing FTP
iii. Security of web browser
iv. Protect the data stored
v. Virtual private network
vi. Secure communication
vii. To encrypt and decrypt any
viii. Voice processed through the MATLAB
ix. To encrypt and decrypt any image

AES 128-bit has been simulated and implemented with equipped key being encrypted along with data input in the process [5, 6]. In this article AES128-bit encryption and decryption process key is only used for data input and is not encrypted throughout the encryption/decryption process. Results are same but proposed algorithm is slightly faster because only data is encrypted in the process of encryption, thus process time and area is optimised. Proposed technique is simulated on ModelSim and implemented on Xilinx ISE Xilinx [7] using Verilog Hardware.

2. Methodology

This 128-bit cipher text becomes the input of decryption module with the same 128-bit key, which decrypts the 128-bit plaintext data to the required 128-bit plaintext data. This complete AES algorithm for encryption and decryption processes with data flow is shown in Figure 1 [8].

AES has two inputs, one for data that we have to encrypt and other is special key as shown in Figure 2. In this paper, data and key inputs both have 128-bits. The results of AES engine will be 128-bits that will be our original input that we made secure for security purpose.

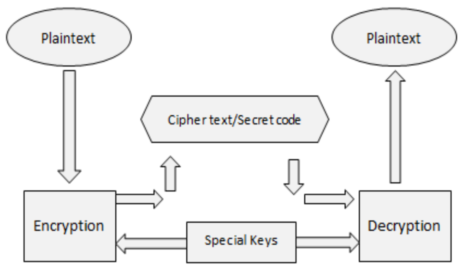


Figure 1. AES Algorithm for encryption and decryption.

AES engine first encrypt as shown in Figure 2**.** The input data by using secret key and coverts it to unreadable text for secure transmission or secure storage data etc. At the receiver end or for read the proper correct data from the storage elements this AES engine decrypts the unreadable data to readable text/data by using same secret key.

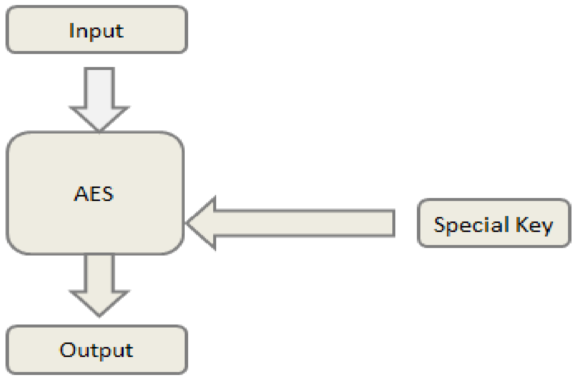


Figure 2. Top module of AES.

In order to make data more secure initialization vector (IV) can be added as an additional box. Top-level module is shown in Figure 3.

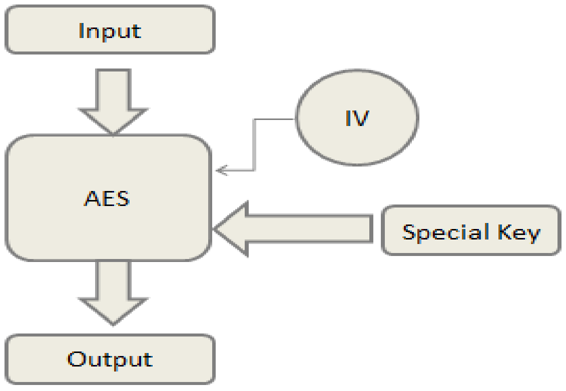


Figure 3. Top module of AES.

The AES algorithm has three standards that are AES-128-bit, AES-192-bit and AES-256-bit. These standards of AES can be easily implemented by varying key length which are used for encryption and decryption processes.

The special key or secret key that is used for encryption and then for decryption, its size can be changed that depending upon our AES standard that we are going to design. If we vary the key length, then the number of rounds in encryption and decryption also vary that obviously more secure the input plaintext. In Table 1, the possible key/block size/round combinations given in tabulated form.

Table 1. Different AES Standards Depending on key Length.

AES Standards	Block size (Nb-words)	Key length (Nk-words)	Rounds (Nr-words)
AES-128	4	4	10
AES-192	4	6	12
AES-256	4	8	14

2.1. Encryption

The encryption types that are commonly used are;

i. Asymmetric Encryption
ii. Symmetric Encryption

These two types of encryption can be distinguished by key that is used for the processing. In symmetric encryption, two keys are used private key and public key while in Asymmetric encryption only one key is used private or public key. The AES encryption process for a 128-bit is shown in Figure 4.

In this figure the input data which is plain text is entered which is then added with add round key stage in which key expansion module is used which yields the series of round keys for encryption. After that substitution box, (sub-byte) state substitutes the byte in each round key. Then Shift Rows stage shifts circularly the substituted data.

After that Mix columns stage in which each column is mixed with columns of data matrix. These four stages repeat nine times in looping manner and then in last stage these all processes are performed once to generate the cipher text, which came into being after all these stages. Encryption process consists of following modules;

i. Substitution Bytes
ii. Shift Rows
iii. Mix Columns

2.2. Decryption

Decryption is the reverse process of encryption. In this process, unreadable text converts into readable plaintext. In Figure 5 all the processes performed in the encryption mode are now reversed using same algorithm. The cipher text passed through all stages described in encryption paragraph. All stages are performed ten times to recover the cipher text to plain text.

Like encryption process four stages repeat nine times in looping manner and then in last stage these all processes are performed once to generate the plain text. In decryption process, same key is used that was used for encryption process. Decryption process consists of following modules;

i. Inverse Substitution Bytes
ii. Inverse Shift Rows
iii. Inverse Mix Column

In encryption and decryption of AES-128-bit algorithm last round does not contain the mix column block and inverse mix column block respectively while for 9 rounds substitute bytes, shift rows and mix column block executes.

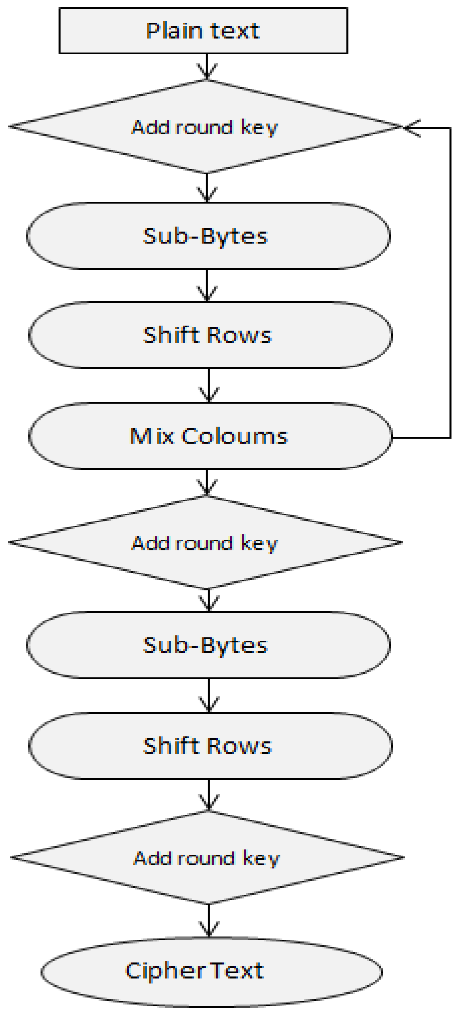


Figure 4. Block diagram of encryption module.

There is also key added after each round which is obtained by key expansion that is input entered key 128-bit key is further expanded and adds to the data that is regulated for ten times. This key expansion block also used the substitution bytes block in order to expand. AES algorithm is design in HDL using Verilog [9, 10] on ModelSim. AES algorithm is implemented on FPGA hardware Spartan-III 3S1000 board [11] using Xilinx ISE synthesis tool, in two ways first by interfacing keyboard through PS2 for user defined plaintext data input to AES engine and display the data outputs of the AES module on termite 2.9 (PC) through serial output of XST-3.0 Board by using UART transmitter. Secondly, the complete algorithm of AES is implemented on XSA-3S1000 board by using UART transmitter and receiver. In this way, AES receives 128-bit plaintext data and key as inputs through receiver AES work on these and gives the 128-bit data output to the transmitter finally output is display on termite 2.9. AES hardware implemented results matched with the ModelSim 6.1 SE simulated results.

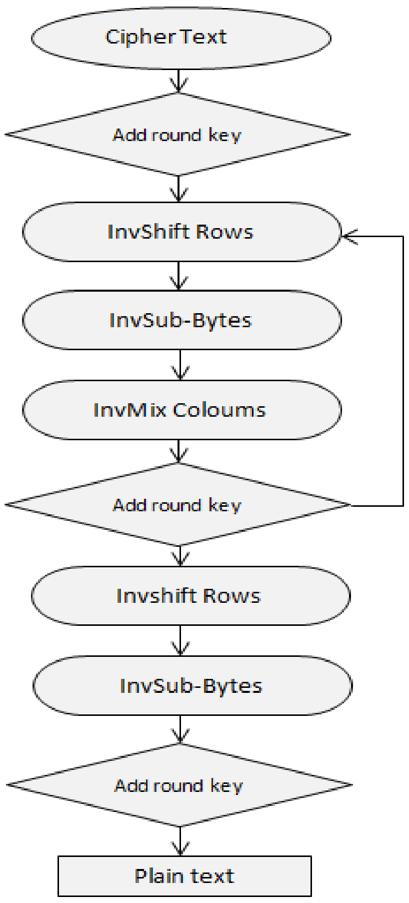


Figure 5. Block diagram of decryption module.

3. Simulations & Results

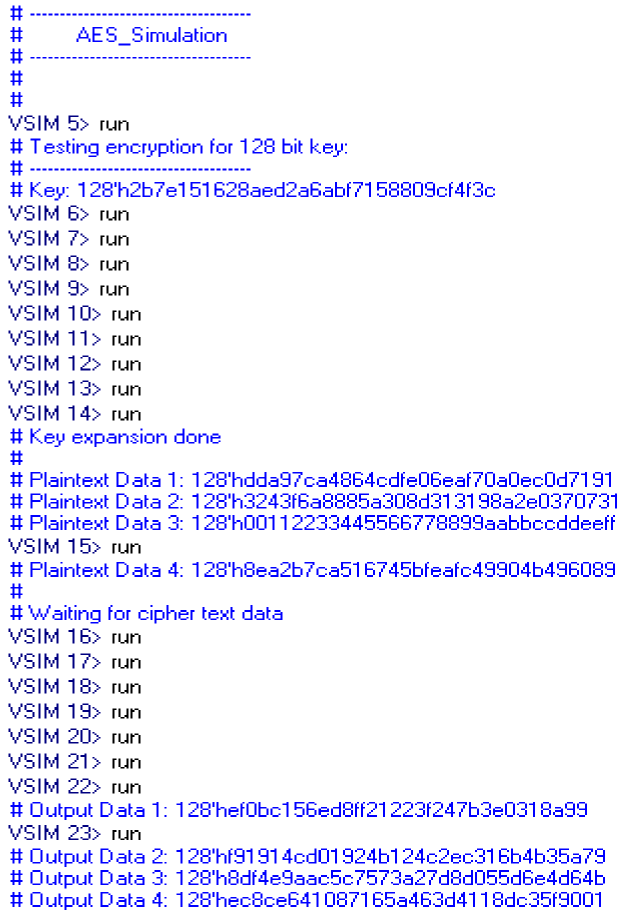
In Table 2 results for the encryption process in which, four Plaintexts as data inputs along with the special key are given to the encryption module and corresponds output data as chipper text is shown in tabulated form.

Table 2. Encryption of four data sets.

Plaintext Data inputs	
Data_1	128’hdda97ca4864cdfe06eaf70a0ec0d7191
Data_2	128’h3243f6a8885a308d313198a2e0370731
Data_3	128’h00112233445566778899aabbccddeeff
Data_4	128’h8ea2b7ca516745bfeafc49904b496089
Key input	
Key	128’h2b7e151628aed2a7abf7158809cf4f3c

Cipher text Data output	
Data_1	128'hef0bc156ed8ff21223f247b3e0318a99
Data_2	128'hf91914cd01924b124c2ec316b4b35a79
Data_3	128'h8df4e9aac5c7573a27d8d055d6e4d64b
Data_4	128'hec8ce641087165a463d4118dc35f9001

ModelSim test bench simulations results for encryption module.



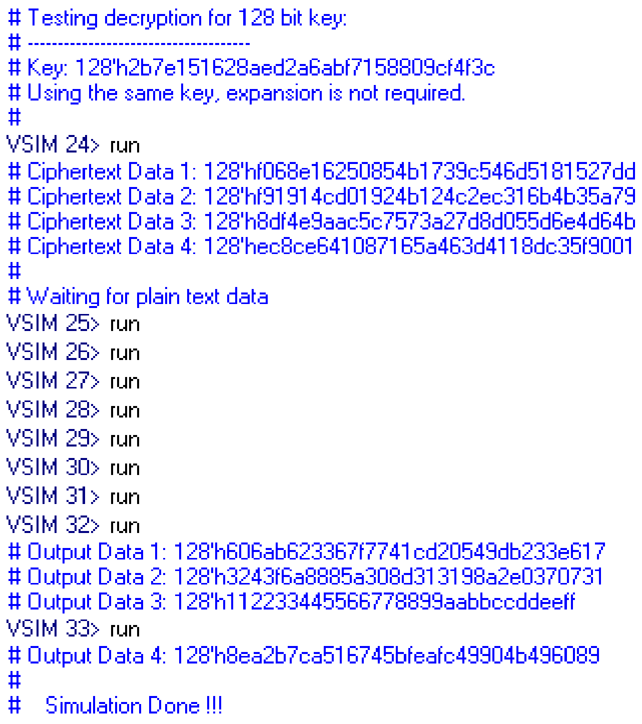
```
# ------------------------------------
#       AES_Simulation
# ------------------------------------
#
#
VSIM 5> run
# Testing encryption for 128 bit key:
# ------------------------------------
# Key: 128'h2b7e151628aed2a6abf7158809cf4f3c
VSIM 6> run
VSIM 7> run
VSIM 8> run
VSIM 9> run
VSIM 10> run
VSIM 11> run
VSIM 12> run
VSIM 13> run
VSIM 14> run
# Key expansion done
#
# Plaintext Data 1: 128'hdda97ca4864cdfe06eaf70a0ec0d7191
# Plaintext Data 2: 128'h3243f6a8885a308d313198a2e0370731
# Plaintext Data 3: 128'h00112233445566778899aabbccddeeff
VSIM 15> run
# Plaintext Data 4: 128'h8ea2b7ca516745bfeafc49904b496089
#
# Waiting for cipher text data
VSIM 16> run
VSIM 17> run
VSIM 18> run
VSIM 19> run
VSIM 20> run
VSIM 21> run
VSIM 22> run
# Output Data 1: 128'hef0bc156ed8ff21223f247b3e0318a99
VSIM 23> run
# Output Data 2: 128'hf91914cd01924b124c2ec316b4b35a79
# Output Data 3: 128'h8df4e9aac5c7573a27d8d055d6e4d64b
# Output Data 4: 128'hec8ce641087165a463d4118dc35f9001
```

In Table 3 results for the decryption process for four input chipper texts data correspond to plaintexts data which uses the same key that earlier used for encryption process is shown in tabulated form.

Table 3. Decryption of four data sets.

Cipher Text Data input	
Data_1	128'hef0bc156ed8ff21223f247b3e0318a99
Data_2	128'hf91914cd01924b124c2ec316b4b35a79
Data_3	128'h8df4e9aac5c7573a27d8d055d6e4d64b
Data_4	128'hec8ce641087165a463d4118dc35f9001
Key	
Key	128’h2b7e151628aed2a7abf7158809cf4f3c
Plaintext Data outputs	
Data_1	128’hdda97ca4864cdfe06eaf70a0ec0d7191
Data_2	128’h3243f6a8885a308d313198a2e0370731
Data_3	128’h00112233445566778899aabbccddeeff
Data_4	128’h8ea2b7ca516745bfeafc49904b496089

ModelSim test bench simulations results for decryption module.



```
# Testing decryption for 128 bit key:
# ------------------------------------
# Key: 128'h2b7e151628aed2a6abf7158809cf4f3c
# Using the same key, expansion is not required.
#
VSIM 24> run
# Ciphertext Data 1: 128'hf068e16250854b1739c546d5181527dd
# Ciphertext Data 2: 128'hf91914cd01924b124c2ec316b4b35a79
# Ciphertext Data 3: 128'h8df4e9aac5c7573a27d8d055d6e4d64b
# Ciphertext Data 4: 128'hec8ce641087165a463d4118dc35f9001
#
# Waiting for plain text data
VSIM 25> run
VSIM 26> run
VSIM 27> run
VSIM 28> run
VSIM 29> run
VSIM 30> run
VSIM 31> run
VSIM 32> run
# Output Data 1: 128'h606ab623367f7741cd20549db233e617
# Output Data 2: 128'h3243f6a8885a308d313198a2e0370731
# Output Data 3: 128'h112233445566778899aabbccddeeff
VSIM 33> run
# Output Data 4: 128'h8ea2b7ca516745bfeafc49904b496089
#
#    Simulation Done !!!
```

Device Utilization results are shown in Table 4 calculated using Xilinx ISE. It shows that proposed algorithm is area optimised as compare to conventional AES algorithm. Power analysis results compared with reference design are shown in Table 5.

Table 4. Device Utilization Summary.

Device Utilization Summary		
Slice Logic Utilization	Proposed	Conventional
Number of Slice Register	1,229	5,992
Numbered used as Flip Flop	1,229	5,864
Numbered used as Latches	0	128
Number of Slice LUTs	3,519	5,230
Number of LUT-FF pairs	979	3,203
Number of bounded IOBs	395	388

Table 5. Power Analysis Compared with Reference Design.

Power Analysis				
On Chip	Power (W)		Used	
Parameters	Proposed	Conventional	Proposed	Conventional
Clocks	0.008	0.008	1	130
Logic	0.007	0.005	3,519	5,230
Signals	0.007	0.005	4,495	9,765
BRAM	0.007	0.005	-	-
IOs	0.007	0.005	395	388
Leakage	0.075	0.064		
Total	0.076	0.065		

4. Conclusions

The Verilog implementation of AES is simulated on Xilinx 10.1. The proposed AES algorithm is faster and contains less area on chip as compared to previous AES algorithms used. This algorithm is verified by creating test bench in ModelSim SE 6.1c for both 128-bit data encryption and decryption modules. This algorithm of AES is verified for four input plain text data, 98% of text is recovered accurately. This algorithm is fast compared to the previous algorithms implemented on FPGA.

The proposed algorithm can be further modified by using 192-bit and 256-bit key. It can also be implemented on FPGA by LCD display or through UART to PC. The results of this implemented algorithm of AES can be transmitted and received through GSM by using any controller circuitry in order to secure communication.

References

[1]. S. Kramer, " Announcing Development Of A Federal Information Processing Standard For Advanced Encryption Standard".1997, http://csrc.nist.gov/archive/aes/pre-round1/aes_9701.txt [cited 2016 April 6].

[2]. M. Welschenbach, "Cryptography in C and C++", 2nd ed. 2006: Apress.

[3]. H. Dobbertin, V. Rijmen, A. Sowa, "Advanced Encryption Standard-AES", 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers. Vol. 3373. 2005: Springer Science & Business Media.

[4]. C. Paar, J. Pelzl, "Understanding Cryptography: A Textbook for Students and Practitioners" 2009: Springer Science & Business Media.

[5]. R. Kohli, M. Kumar, "FPGA Implementation of Cryptographic Algorithms using Multi-Encryption Technique", International Journal of Advanced Research in Computer Science And Software Engineering, 2013.

[6]. M. Pitchaiah, P. Daniel, "Implementation of Advanced Encryption Standard Algorithm", International Journal of Scientific & Engineering Research, 2012, 3(3).

[7]. Xilinx. http://www.xilinx.com/. [cited 2016 April 3].

[8]. S. Sutherland, S. Davidmann, P. Flake, "SystemVerilog for Design: A Guide to Using SystemVerilog for Hardware Design and Modeling" 2nd ed. 2006: Springer Science & Business Media.

[9]. S. Palnitkar, "Verilog HDL: A Guide to Digital Design and Synthesis", Vol. 1. 2003: Prentice Hall.

[10]. M.D. Ciletti, "Advanced Digital Design with the Verilog HDL" Vol. 1. 2003: Prentice Hall Upper Saddle River.

[11]. C. Spear, "SystemVerilog for Verification: A Guide to Learning the Testbench Language Features", 2008: Springer Science & Business Media.