Available online @ https://jjem.jnnce.ac.in
https:www.doi.org/10.37314/JJEM.2020.040103
Indexed in International Scientific Indiexing (ISI)
Impact factor: 1.025 for 2018-19
Published on: 30 November 2020

Implementation of AES Algorithm Using Verilog

Ganesh Gopal Shet , Jamuna V, Shravani S, Nayana H G, Pramod Kumar S

J N N College of Engineering, Shimoga.

ganeshshet800@gmail.com, jamunav014@gmail.com ,
shravanishamsundar@gmail.com, nayanahg16@gmail.com, pramodkumars@jnnce.ac.in

Abstract

Cryptography is very important now-a-days for data security and integrity as the ecommerce and internet applications has increased. But, it has least importance in many cases because of extra memory and other requirements needed for the implementation. The main aim of this work is to implement Advanced Encryption Standard (AES) Encryption using Verilog. To protect data like electronics, cryptographic algorithms are used. The digital information can be encrypted and decrypted by the block cipher of AES algorithm. It can be implemented with the key length 128, 192, 256 bits. Each round of encryption associated with delay can be reduced by AES parallel design.

Keywords: *Implementation, Multi-hop Routing Protocol, Energy Consumption*

1. INTRODUCTION

In this technical era, we have seen a drastic growth in telecommunication. Through internet anybody can access the data present in computer in any corner of the world. Many activities like e-commerce, data sharing etc. will happen through internet. So, data authentication and secured communication become very important. Here the cryptography will play an important role. Now a day's many data encryption algorithms are available. Digital information can encrypt and decrypt by using block cipher by using cryptographic keys of 128, 192 and 256 bits [1].

The main objective of this project is to implement AES algorithm using Verilog and to give optimum circuit with clock frequency, path delay, time required to generate keys and decoding the data [2]. For the secure communication, cryptography is useful in presence of third parties. It mainly deals with the analysing protocols and overcome the influence of information on security. The disciple like mathematics, computer science and electrical engineering inspected by modern cryptography [3].

2. Problem statement

Considering the history of communication, it is not surprising that security has taken the last seat. Because implementing security mechanisms in distributed applications creates extra overheads like more memory, handshaking, more CPU time for calculating keys etc. [4].

The main aim of this work is to provide a solution for the above stated problem with the help of Verilog code using Xilinx. In real time applications the software code takes lot of time to execute the same code again and again [5]. But particular hardware for a repeating code reduces the execution time. So, this work provides solution to the above-mentioned real-time problem.

3. Methodology

The methodology involved in this system is Verilog code. To support both analog and digital circuit designing, Xilinx provides

analog and digital platform. It is interesting to note that any encryption algorithm works in a digital environment and all the blocks in the system will handle digital data.

3.1. Advanced Encryption Standard (AES)

AES is a type of cryptography algorithm. It performs the encryption and decryption operation. The input information is known as plaintext and encrypted form is called as ciphertext. Ciphertext contains the plaintext information, but it is not in a readable form to humans. Encryption procedure is varied to one algorithm to another algorithm. Without the key ciphertext cannot be used to encrypt or decrypt. AES has proven to be more efficient than its encryption processors. AES is mainly used in voice communication, network applications, vertical private network, secured socket layer (SSL) [6-7].

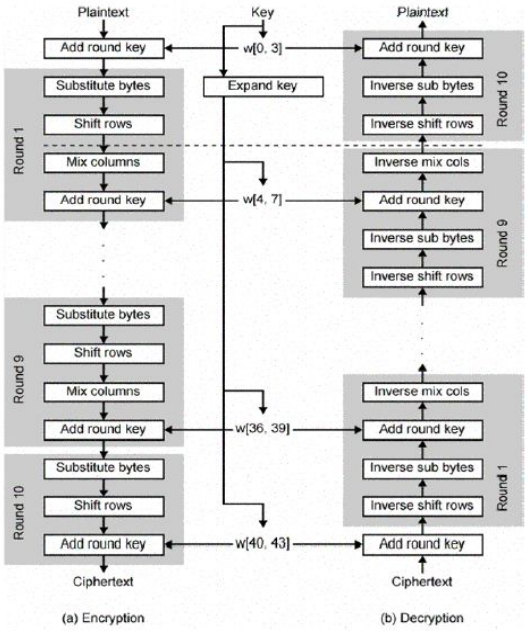

Figure 1: AES-128bit Overall Representation.

This overall representation has 4 operations as show in figure 1:

- SubByte()
- ShiftRow()
- MixColunm()
- AddRound key ()

Table 1: Key-Block-Round combinations.

	Key Length (NK words)	Block Size (Nb words)	Number of Rounds(Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

For a 128-bit implementation it requires the 10 rounds of operation as shown in table 1. Each round having 4 steps as following:

1. Byte substitution using a substitution Box (S-box).
2. Shifting rows of matrix by different offsets.
3. Mixing the rows with each columns of the matrix.
4. Adding a Round Key to the matrix.

3.2. The AES Algorithm

The AES algorithm has mainly 3 encryption modes: 128 bits, 192 bits, 256 bits. Each encryption mode has a corresponding number of rounds Nr based on length of Nk words. The state array block size term Nb is constant for all encryption modes. Each state has 4 words, each words has 4 bytes.

3.2.1 Encryption Process:

Both the encryption and decryption process consist of a number of various transformations. The number of rounds depends on the length of the key used for the encryption process and decryption process. For 128 bit, state array requires 10 rounds of iteration for ciphertext conversion (Nr=10). Each Nr-1 has 4 different transformation i.e, Subbyte(), ShiftRow(), MixColulnm(), AddRound key().

In Subbyte() Transformation, only a nonlinear component is used in the entire process. Eachbyte operates independently. Each byte is replaced with S-box values. This S-box invert is generated by taking the multiplicative inverse in the finite field GF

(2^8) with irreducible polynomial m(x) = x^8+x^4+x^3+x^1. After this apply the Affine Transformation over GF (2^8).

In ShiftRow() Transformation, it cyclically shifts bytes of rows in state array. Each row is shifted by specified different offset values. This operation is similar in decryption process except to different rotational offset as shown in figure 2.

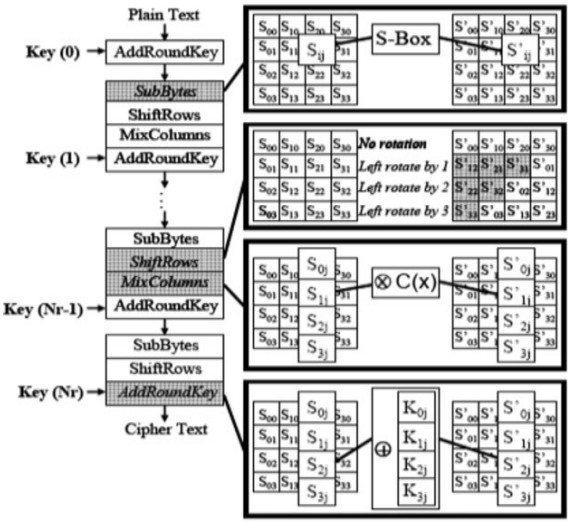

Figure 2: AES Encryption Process.

In Mixcolumn() Transformation, it operates column by column on state array treating each 4 term polynomial. Each column represented as polynomials over GF (2^8) and multiplied by modulo x^4+1 with fixed polynomial as follows:

$$m(x)=\{03\}x^3+\{01\}x^2+\{01\}+\{02\}$$

In AddRoundKey() Transformation, the Round key is XORed with the output of Mixcolumn block. Each Addroundkey consists of Nb words from the key expansion unit. These Nb words are added into the column of state array. This is similar in decryption process. In any expansion units a previous Round key,generates a 4 word array generated as a constant that changes each round and a service of S-box values replacement for each 32 bit word of key. The key scheduling unit generates an overall Nb (Nr+1) words as shown in figure 3.

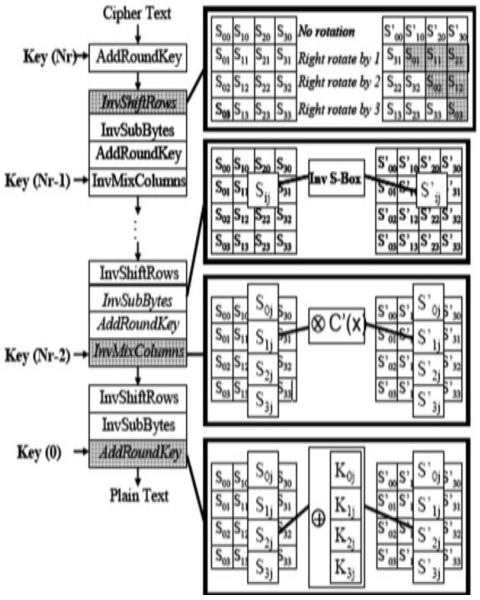

Figure 3: Encryption Using Cipher Feedback (CFB)

3.2.2 Decryption Process:

For decryption, the same process in the reverse manner. It taking ciphertext as a input and plaintext as an output. It also contains 10 rounds of process. Each round has 4 steps similar to encryption process. AddRound key is same for both encryption and decryption and remaining steps is just inverse in decryption i.e. Inverse Subbytes() , InverseShiftRows() and Inverse Mixcolumn().

5. Verilog Implementation

Verilog HDL is used because it is easier to explore different design options, flexibility to exchange among environments. The implementation code is pure without changing the design Verilog code that could easily be implemented on other devices, with. We have used mainly three tools to implement the code Notepad++, Questasim, Xilinx Synthesis and Simulation Tools (ISE14.7). The goal of design implementation is speed optimization, keeping other constraints as minimum as possible. We have implemented CBC Mode of AES Rijndael Algorithm.

5. Tool Details

The editor used for writing the design codes is Notepad++. Questasim 10.0 is used for debugging and optimizing the design codes and simulating. Xilinx ISE 14.7 is used for synthesizing the design to the Zed (Zynq Evaluation and Development) Board as shown in figure 4. The code implementation results are based on Questasim 10.0 simulation results.

Zynq-7000 All Programmable SoC							
	Device Name	Z-7010	Z-7015	Z-7020	Z-7030	Z-7045	Z-7100
	Part Number	XC7Z010	XC7Z015	XC7Z020	XC7Z030	XC7Z045	XC7Z100
Processing System	Processor Core	Dual ARM® Cortex™-A9 MPCore™ with CoreSight™					
	Processor Extensions	NEON™ & Single / Double Precision Floating Point for each processor					
	Maximum Frequency	667 MHz (-1); 766 MHz (-2); 866 MHz (-3)			667 MHz (-1); 800 MHz (-2); 1 GHz (-3)		667 MHz (-1); 800 MHz (-2)
	L1 Cache	32 KB Instruction, 32 KB Data per processor					
	L2 Cache	512 KB					
	On-Chip Memory	256 KB					
	External Memory Support[1]	DDR3, DDR3L, DDR2, LPDDR2					
	External Static Memory Support[1]	2x Quad-SPI, NAND, NOR					
	DMA Channels	8 (4 dedicated to Programmable Logic)					
	Peripherals[1]	2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO					
	Peripherals w/ built-in DMA[1]	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO					
	Security[2]	RSA Authentication, and AES and SHA 256b Decryption and Authentication for Secure Boot					
	Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master 2x AXI 32b Slave 4x AXI 64b/32b Memory AXI 64b ACP 16 Interrupts					

Figure 4: Zed Board Device Specifications

6. Encryption

6.1 Encryption Pre Round

The Simple bitwise XOR operation are performed in Pre Round Operation. .Because of fully parallel architecture output of this stage is registered.

6.2 Encryption Inner Rounds

There are 10 rounds as per 128-bit AES Algorithm. Every round includes 4 sub modules SubByte() Transformation, ShiftRow() Transformations, MixColoumns() Transformation and AddRoundkey() Transformation. Inner round includes 9 rounds remaining one round is implemented as last Round. For implementing 10 rounds, if we instantiate each module 10 times, the overall area requirement increases 10 times and implementing it with Zed Board (XC7Z020) resources utilization exceeds by 100% for each encryption and decryption. .

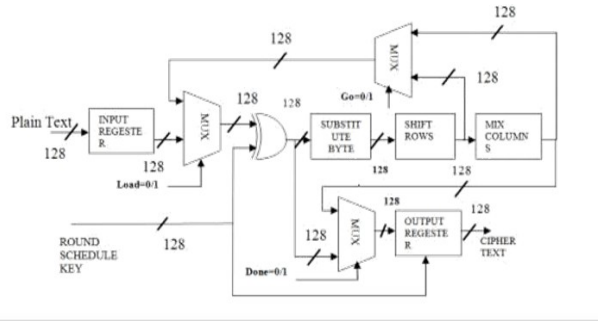

Figure 5: Encryption Module Gate Level.

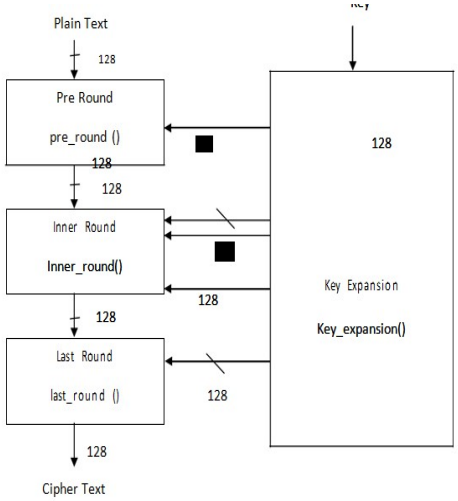

Figure 6: Encryption Process

To overcome this problem we used the concept of reusing the same modules as many times they are required. We used state diagram at the top level [middleround()], which uses the same module each times and registered output is sent to the next state as input. Because of this process, the IOB utilization is reduced to 5% and Slices utilization to 43% as shown in figure 5 and 6.

6.3 Encryption Last Round

The last round contains three operations namely

- SubByte()Transformation
- ShiftRow()Transformation()
- AddRoundKey()Transformation
- But MixColumn()Transformation operation is excluded.

6.4 Unit of Key Expansion

From the original input round key, we are generating all round keys. In encryption will the original key will be last group of the generated key i.e. Expansion keys in case of decryption. Key Expansion module includes sub modules rotate word() Transformation, Sub Word() Transformation, Round constant XORing(), and key round Module().

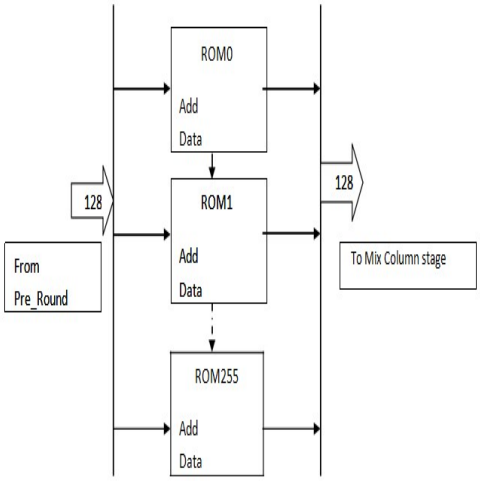

Figure 7: S-ROM Implementation

Rotate word () Transformation: The cyclic rotation can be performed by each word from [a0,a1,a2,a3] and returns the word [a1,a2,a3,a0]. SubWord() Transformation (key s-box) it is same as SubByte() Transformation module only the difference is that it processes bits. Round Constant () Transformation (key rcon()). Predefined round 32 bits constants of GF are fixed for each round. A 4-bit round number and 32-bit output of Sub Word () Transformation is taken as input. Values corresponding to round key are fetched from ROM and XORed with keysbox(). AddRound() Transformation (key round()). Key round is XORed with previous round keys and output of key rcon() as shown in figure 7.

7. Decryption

The Inverse Ciphertex transformations can be performed by the reverse order of the Ciphertext transformation.The transformations used in the Inverse Ciphertext are: InvShiftRow(),InvSubByte(), InvMixColumn() and AddRoundKey().

As the decryption is inverse of encryption the operations are performed in the inverse manner of encryption. The last round of the encryption becomes the first round indecryption Process and the expanded key generated in Key Expansion () is feedback instead of cipher key as shown in figure 8 and 9.

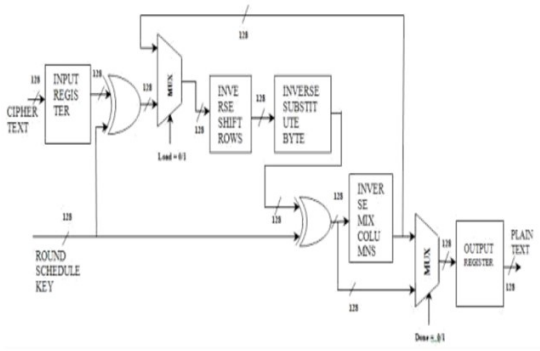

Figure 8: Decryption Module - Gate Level

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	22531	106400	21%
Number of Slice LUTs	23309	53200	43%
Number of fully used LUT-FF pairs	8166	37674	21%
Number of bonded IOBs	10	200	5%
Number of BUFG/BUFGCTRL/BUFHCEs	2	104	1%

Figure 9: Encryption Output Waveform

8. Result Analysis

Simulation results are based on test performed on Questa Sim10.0. The verification is done using Verilog HDL.

a) Test Case1 - Reset Case: Reset is asserted (active high), all signals are assigned to zero.

b) Test Case2 - Encryption: The AES Module inputs are driven for encryption and expected outputs are obtained. All the sequences below are in hexadecimal.

Plaintext:
11111111111111111111111111111111

CipherKey:
3C4FCF098815F7ABA5D2AE2816157E2B

Expected Ciphertext:
A806164898CFC9ACC0546CB8DDFABF89

c) Test Case3 - Decryption: The AES Module inputs are driven for decryption and expected outputs are obtained. All the sequences below are in hexadecimal.

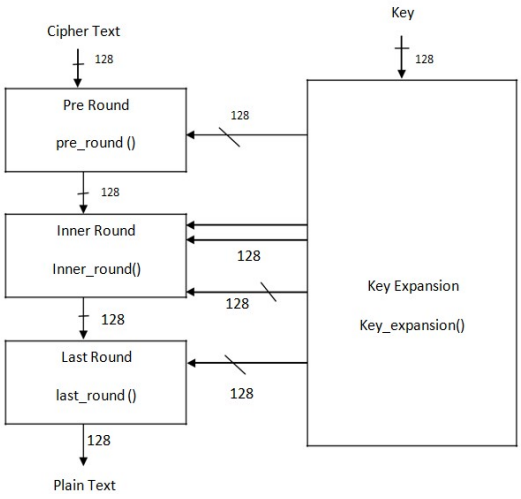

Figure 10: Decryption Process

Encrypted Cipherext:
A806164898CFC9ACC0546CB8DDFABF89

Cipher Key:
3C4FCF098815F7ABA5D2AE2816157E2B

Expected Plain Text:
11111111111111111111111111111111
Associated to this results are depicted in figure 9 to 11.

9. Synthesis report

Overall implementation of parallel AES Algorithm used resources as shown below:
Timing Parameters:
• Speed Grade: -1
•Minimum period: 4.171ns (Maximum Frequency: 239.733MHz).
•Minimum input arrival time before clock: 3.719ns.
•Maximum output required time after clock: 1.219ns.
•Maximum combinational path delay: 1.261ns.

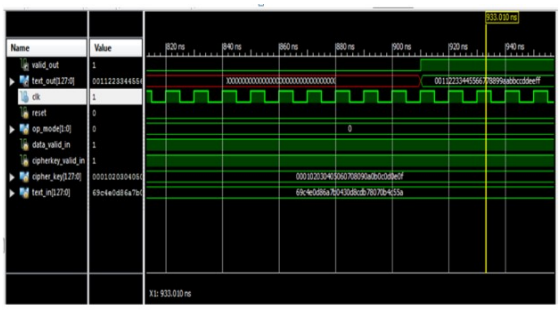

Figure 11: Decryption Output Waveform

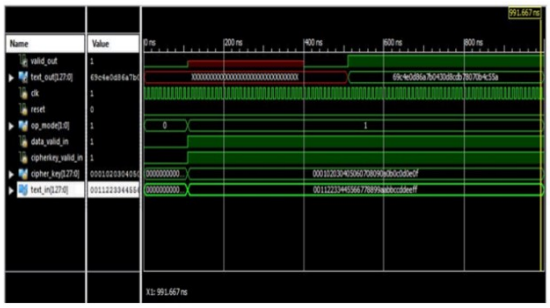

Figure 12: Resource Utilization

10. Conclusion

Each round of encryption associated with the delay that can be reduced by the parallel design of AES encryption that operates in higher frequency than non-pipelined, non-parallel design. In time critical encryption applied this type of encryption Sub title in message throughput. The hardware implementation of AES provides faster speed than software implementation like secure key in encryption and higher throughput.

The work has been extended in order to increase the security for more severe attacks since the encryption time has been reduced.

There has been further scope to optimize the utilization of resources. The implementation can be further improved to achieve more efficient usage of the resources and increase the maximum clock frequency. The key length can be reduced, maintaining the same security, in order to optimize there source utilization. The few gaps have been covered but still a lot can be done to achieve the security of data along with the optimization of resources.

References

1. K. Xinmiao Zhang, High speed VLSI architectures for the AES algorithm ,IEEE transactions on VLSI systems, Tech. Rep., sep2004.
2. Kumar, Mamatha MS Pramod, and M. Mamatha. "FPGA Implementation Of Low Area Single Precision Floating Point Multiplier."International Journal of Science Technology and Engineering, Vol.2, no. 2 (2016): 560-566.
3. csrc.nist.gov.Publicationsfips-197 National Institute of Standards and Technology, Advanced Encryption Standard, Federal Information Processing Standards 197, November 2001.
4. M.Natheera Banu, FPGA Based Hardware Implementation of Encryption Algorithm, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 8958, Volume-3, Issue-4, April 2014.
5. M.Pitchaiah, Philemon Daniel, Praveen, Implementation of Advanced Encryption Standard Algorithm, International Journal of Scientific Engineering Research.
6. Kumar, Pramod, T. V. Narendra, and N. A. Vinay. "Short Hand Recognition using Canny Edge Detector." International Journal 7, no. 5 (2017).
7. Deguang Le, Jinyi Chang , Xingdou Gou , Ankang Zhang ,Conglan Lu ,Parallel AES Algorithm for Fast Data Encryption on GPU, IEEE journal on AES 2010.