DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4407

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

IMPLEMENTATION OF AES ALGORITHM ON FPGA FOR LOW AREA CONSUMPTION

Gurpinder Kaur
Department of ECE,
Punjabi University, Patiala, Punjab, India

Dr. Amandeep Singh Sappal
Department of ECE,
Punjabi University, Patiala, Punjab, India

***Abstract*:** With the increasing number of internet and wireless communication users the demand for security measures to protect user data transmitted over open channels increases. So cryptography becomes important for such sensitive data which needs to be kept secured. AES can be considered as the most widely used modern symmetric key encryption standard. This paper propounds hardware implementation of AES to achieve less area and high speed. The proposed AES design supports 128 bit key length and 128 bit data blocks. Single register is used to store the round keys in each round of key expansion to reduce area consumption. The AES-128 is implemented on FPGA using Verilog language with the help of Xilinx ISE tool.

Keyword: Advanced Encryption Standard (AES), Field Programmable Gate Array (FPGA)

1. INTRODUCTION

Data security is a major problem for any organization that has valuable information that needs to be kept secured from automated spying/hacking. So to avoid savage attacks on information, cryptography can be used as effective tool. AES can be considered as the most widely used modern symmetric key encryption standard [1].
In 2000 NIST announced that the rijandael algorithm from Belgium has been selected as the Advanced Encryption Standard (AES) algorithm. After that AES algorithm has attracted attentions from various departments since it provides high level of security and can be implemented easily [2].
AES can be implemented both in hardware or software but hardware implementation is used in real time applications from high end computers to low power portable devices. Since there is a set of complex computational steps in AES algorithm so software implementation of AES algorithm is slow and thus consumes large amount of time to complete. Whereas AES hardware implementation is fast, very reliable and conveniently suitable for high speed applications and moreover it does not require system resources used in software implementation. AES hardware implementation can be easily reset and erase data on risk and thus provide better system performance. Also hardware implementation is economically better than software implementation [3]. There are two major platforms for hardware implementation. They are: (1) Application Specific Integrated Circuits (ASICs) (2) Field Programmable Gate Array (FPGA). In this paper FPGA has been used for the implementation of AES as it inhibits parallelism and is reconfigurable. Also FPGA approach proves to be
.
economically better than ASIC implementation. Further FPGA architecture consists of various features such as block memory (BRAM), Digital Signal Processing (DSP) cores[7]. Major problem in hardware implementation of AES is higher area and power consumption. Main goal of AES hardware implementation is to minimize hardware and speed up the Algorithm with minimum increase in hardware. The rest of the paper is organized as follows. Section II presents a brief overview of AES and previously proposed existing work done and section III provides the proposed work. Section IV shows the implementation results and finally, section V concludes the paper.

2. ADVANCED ENCRYPTION STANDARD

The AES is a private or symmetric block cipher which uses the same key for encryption and decryption. AES algorithm can encrypt or decrypt block size of 128 bit using cipher keys of 128, 192, 256 bits [4]. Depending on the key length the each round is repeated Nr number of times. In this paper the key size is 128 and each round is repeated 10 times.

TABLE I. KEY LENGTH AND ROUNDS

Key size(bits)	No. of Rounds
128	10
192	12
256	14

AES Encryption consists of four different transformations – SubBytes, ShiftRows, MixColumns, AddRoundKey. Figure 1 shows the various steps of AES Encryption and Decryption

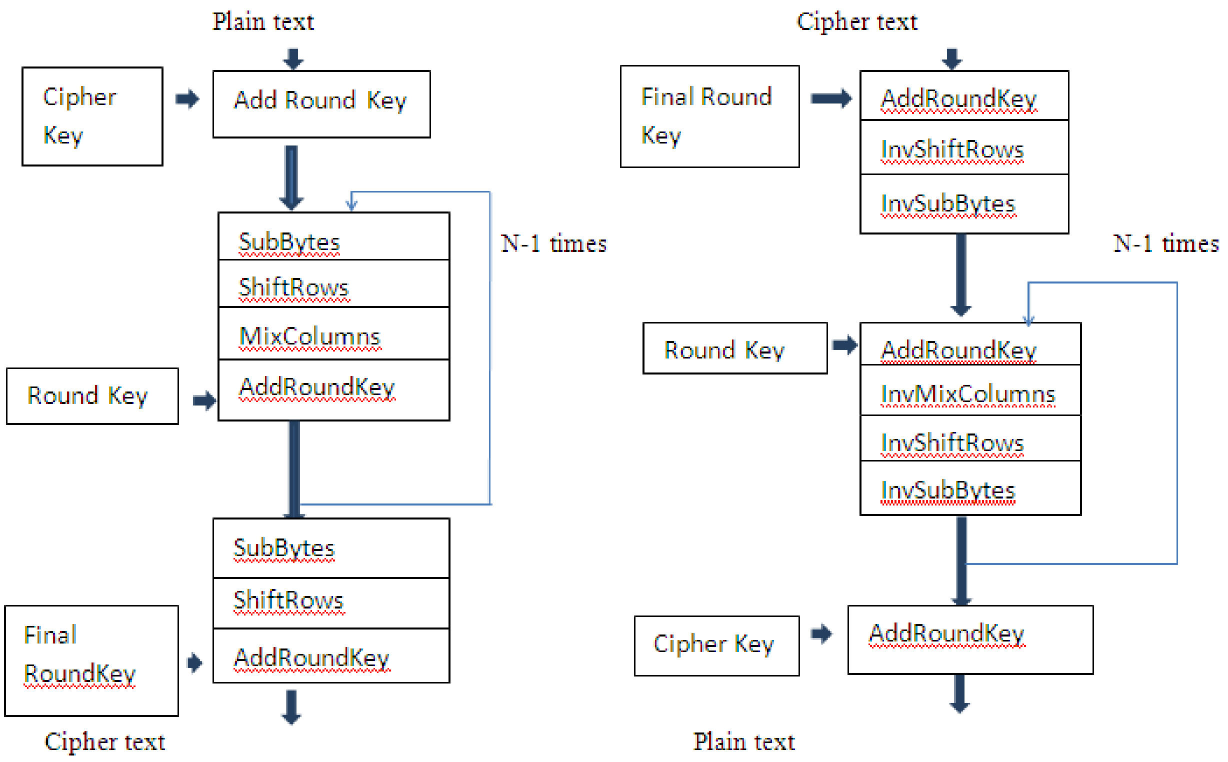

Figure 1. Block diagram of AES Encryption and Decryption.

A. *SubBytes*

In byte substitution layer each element of the state is transformed non-linearly using look up tables with some special mathematical properties. The mathematical operations consists of multiplicative inverse over GF(2^8) and then affine transformation.

1) *Inverse SubBytes:* In order to compute the reverse of S-Box substitution inverse of affine transformation is computed and then inverse operation is reversed by computing the reverse again [5].

B. *ShiftRows*

ShiftRows layer permutes the data on byte level to increase the diffusion properties of AES. The first row of ShiftRowtransformation remains unchanged while the second, third and fourth row of state matrix shifts by three, two and one byte respectively to the right.

2) *Inverse ShiftRows:* in order to reverse the shift rows operation each row of the state matrix is shifted in opposite direction i.e. to the left and the first row remains unchanged [5].

C. *MixColumns*

MixColumn transformation is a linear transformation which mixes each column of state matrix independently. Each column is considered as a polynomial over the GF(2^8) and then multiplied by fixed polynomial modulo $x^4 + 1$. The combination of ShiftRows and MixColumn makes it possible that after only three rounds every byte of state matrix depends on all 16 plaintext bytes. MixColumn layer is not present in last round of AES encryption.

3) *Inverse MixColumn:* in inverse MixColumn each column is considered as polynomial over GF(2^8)and then multiplied by a fixed polynomial modulo $\{0b\}x^3+\{0d\}x^2 \ +\{09\}x^1 \ +\{0e\}$. This layer is not present in first round of decryption [6].

D. *Key Addition layer*

Two inputs for key addition layer are 16 bytes (128 bits) state matrix and a subkey of 16 bytes. These two inputs are combined by a bitwise XOR operation.

E. *Key Scheduling*

The subkeys are derived in the key scheduling. The key schedule takes the original input key(of length 128, 192 or 256 bits) and derives the remaining subkeys. Total number of subkeys derived are number of rounds plus one. AES key schedule is word oriented, where 1 word = 32 bits. for 128 bit key, first it accepts 4 bytes input word and thus performs cyclic permutations. It takes 4 byte words as input and then via byte substitution each byte is substituted by another byte and then xoring is performed using a round constant word array.

Since the AES is a symmetric cipher .i.e. same key is used for both encryption and decryption, so the key computed in last round of encryption is applied to first round of decryption, the second to last computed key is applied to second round of decryption and so on.

3. PROPOSED WORK

This section describes the proposed method for low area consumption along with maintaining high speed on a single hardware.

A. *Hardware Efficient Architecture*

The substitution box used in SubByte and InvSubByte can be implemented by two methods, they are BRAM implementation and combinational logic [3]. In this paper S-Box architecture based on combination logic is present. In this paper single register is used for temporary storage of round key values computed in each round. Use of single register instead of individual register for each round saves hardware and thus leads to lower area consumption.

B. *Pipelined Architecture*

In this paper pipelined architecture is used for achieving high speed as compared to the previous work. In pipelining when one round is completed simultaneously, the bytes are accepted for the next round. To increase the speed pipelining architecture proves to be very effective [6].

C. *AES data path improvement*

In AES algorithm control is observed to be independent of data. In this paper finite state machine technique (FSM) is used to easily understand the data path during encryption and decryption process. The finite state machine technique is used to represent and control execution flow. This technique provides detailed path to be followed to reach the output.

D. *FSM Encryption*

Initially when reset is on, no data will be processed and the system is in pre_idle state waiting for the data to encrypt. When reset is set low .i.e. reset = 0, then the system goes to idle state. The system fetches the 16 byte input data block .i.e. plain text in idle state. When the data block is ready the system moves to next state(s0). The s0 state indicates the first round of encryption in which the original data .i.e. plain text is xored with original input key. This xored data is applied to S-Box and then after ShiftRow transformation MixColumn is performed. So after the first round algorithm moves to second state s1 where the output of first round is xored with the subkey generated using original key and thus applied to the four transformations. Similarly for the next eight rounds this state is repeated. And in the last round all the process remains same except the MixColumn transformation is excluded. Once the last round .i.e. 10th round is completed the encryption is done for the first 16 byte data block and thus the done signal goes high indicating that the encryption is done and the system is ready to encrypt the next 16 byte data block.

E. *FSM Decryption*

Decryption is the reverse process for encryption. Reset and pre_idle state are similar to those used in FSM Encryption. When reset is low the system moves to idle state and waits for the data to decrypt. When done signal is high it indicates that data is encrypted and is thus ready to decrypt. So system moves to s0 state where first round of decryption is performed which is similar to the last round of encryption and thus switches to next state which repeats in the same manner for the next eight rounds and includes InvMixColumn transformation and in the last round after completion of last round .i.e. when system decrypts the cipher text back to plain text(input data) it provides a decrypt signal to indicate that decryption is done and the system moves to pre_idle state thus waiting for the new data to arrive.

4. RESULTS AND COMPARISONS

A Xilinx ISE 13.2 tool has been used for synthesis and implementation and Xilinx Isim(0.61xd) for testing and verification of simulation results. The experimental results has been observed using FPGA family XC6SLX16-3-CSG324. Simulation results of AES Encryption and decryption are shown in figure 2 and figure 3. The proposed design area utilization for encryption and decryption are summarized in table 2 and table 3 along with their performance in comparison to the previous work.

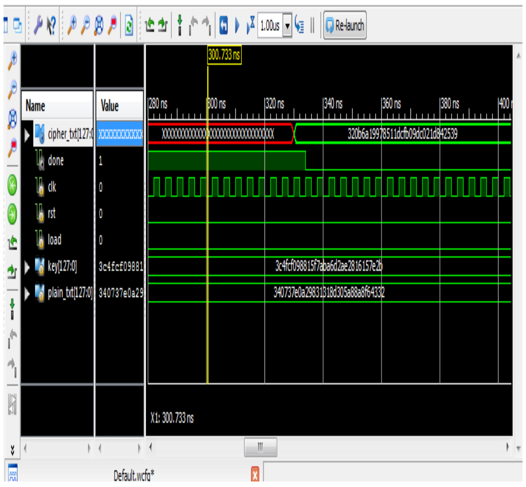

Figure 2: Simulation results of AES Encryption

TABLE 2. Comparison of proposed Encryption design with existing design[3]

Parameter	Proposed work	Existing design[3]
Data path(bit)	128	128
No. of rounds	10	10
Slice Registers	509	564
Slice LUTs	1329	3559
Fully used LUT-FF pairs	403	459
Block RAM	1	4
Combinational path delay(ns)	0	0
Max. operating Frequency(Mhz)	285.759	273.997
Throughput(Mbps)	2085.56	855.61

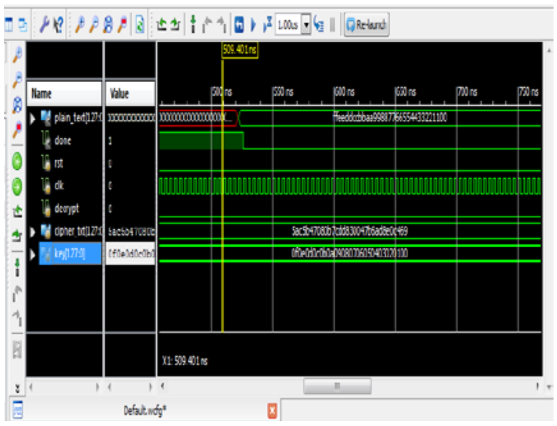

Figure 3: Simulation results for AES Decryption

TABLE 3. Comparison of proposed Decryption design with the existing design

Parameter	**Proposed work**	**Existing design[3]**
Data path(bit)	128	128
No. of rounds	10	10
Slice Registers	580	607
Slice LUTs	2197	3531
Fully used LUT-FF pairs	548	426
Block RAM	1	20
Combinational path delay(ns)	0	0
Max. operating Frequency(Mhz)	282.319	223.157
Throughput	2111.196	696.712

5. CONCLUSION

AES plays a very important role in security applications. Since the software implementation of AES is unsatisfactory for real time applications so hardware implementation of AES is used. High speed implementation is achieved using pipelined approach. The speed needs to be increased being very careful towards area as with increase in speed area also increases. So a trade-off needs to be maintained between speed and hardware such that with lowest increase in area the speed is increased. The proposed design shows improvement in terms of hardware and speed.

REFERENCES

[1] Gulroz Singh, Mankirat Singh Lamba, “Efficient Hardware Implementation of Image Watermarking Using DWT and AES Algorithm”, 39th National System Conference, pp 1-6, Dec 2015.

[2] Standard N F, Announcing the advanced encryption standard(AES), Federal Information Processing Standards Publications, 2001.

[3] Pritamkumar N. Khose and Vrushali G. Raut, “Implementation of AES Algorithm on FPGA for Low Area Consumption”, proc. 2015 International Conference on Pervasive Computing (ICPC).

[4] FIPS 197, Advanced Encryption Standard http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[5] Christof Paar and Jan Pelzl, “ Understanding Cryptography”, pp 87-117, DOI 10.1007/978-3-642-04101-3 1,_c Springer-Verlag Berlin Heidelberg 2010.

[6] Pournima U. Deshpande and Smita A. Bhosale, “AES Encryption Engines of Many Core Processor Arrays on FPGA by using Parallel, Pipeline and Sequential Technique”, proc. International Conference on Energy Systems and Applications(ICESA), pp 75-80, 30 Oct – 01 Nov 2015.

[7] Rajpreet Singh, Tripatjot Singh Panag, Amandeep Singh Sappal, “FPGA implementation of Optimized Decimation Filter for Wireless Communication Receivers”, International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, issue 4, April 2014.