Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Secured High throughput implementation of AES Algorithm

Manjesh.K.N*
Dept. of Electronics & Communication Engg.
BIT Bangalore, India

R K Karunavathi
Dept. Of Electronics & Communication Engg
BIT Bangalore, India.

Abstract— Cryptography is the study of Mathematical techniques for secured communication in the presence of adversaries and also it deals with the aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication. A high speed security algorithm is always necessary and important for wired/wireless communication. The symmetric block cipher plays a major role in the bulk data encryption. One of the best existing symmetric security algorithms to provide data security is advanced encryption standard (AES). AES has the advantage of being implemented in both hardware and software. Hardware implementation of the AES has lot of advantage such has increased throughput and better security level. In this paper, Hardware Implementation of different stages of pipelining for 128 bit AES encryption and Decryption has been made using Verilog hardware descriptive language .Results for the above implementation has been tested and tabulated. This paper also gives an overview idea of variation of Power, Throughput & Area with different pipelining stages .The pipelined architecture of the AES algorithm is proposed in order to increase the throughput of the algorithm.

Keywords— AES, Cryptography, Pipelined AES, Security, Encryption, Decryption

I. INTRODUCTION

The large and growing number of internet and wireless communication users has led to an increasing demand of security measures and devices for protecting the user data transmitted over the unsecured network so that unauthorized persons cannot access it. The increasing need for Secured data communication has led to development of several cryptography algorithms. Two types of cryptographic systems are mainly used for security purpose, one is symmetric-key crypto system and other is asymmetric-key crypto system. Symmetric-key cryptography (DES, 3DES and AES) uses same key for both encryption and decryption. The asymmetric-key cryptography (RSA and Elliptic curve cryptography) uses different keys for encryption and decryption. Symmetric crypto system has advantages over asymmetric crypto system. Symmetric key Algorithms are in general much faster to execute electronically than asymmetric key algorithms. Smaller key length is the major disadvantage of DES crypto system.

In November 2001, the National Institute of Standards and Technology (NIST) of the United States choose the Rijndael algorithm as the suitable Advanced Encryption Standard (AES) to replace previous algorithms like DES, 3DES algorithm. The AES encryption is considered to be efficient in both hardware and software implementations. Compared to software, hardware implementation is more secured and reliable. Software implementation of AES algorithm is slower process So the focal approach of our design on hardware platform is to attain speed. So this paper addresses efficient hardware implementation of the AES (Advanced Encryption Standard) algorithm and describes the design and performance evaluation of Rijndael algorithm. A strong focus is placed on high throughput implementations, which are required to support security for current and future wide bandwidth applications .This implementation will be useful in wireless security like military communication, Cellular networks, Web servers, Mobile networks, Smart cards etc.

II. AES ALGORITHM

In cryptography, the Advanced Encryption Standard (AES), also known as Rijndael, is a block cipher adopted as an encryption standard by the US government. The cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen and submitted to the AES selection process under the name "Rijndael", a portmanteau comprised of the names of the inventors. AES is a symmetric iterative private key block cipher algorithm that can process data of different length say 128, 192 or 256 bits with a private key of same length. Each iteration in an algorithm is called as a round and it has 10, 12 or 14 rounds for processing data blocks of 128,192 or 256 bits respectively. The key could be generated and scheduled in each round to get the encrypted data. Table 1 shows the number of rounds as a function of key length.

TABLE I. Different AES specifications

AES Version	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr rounds)
AES126	4	4	10
AES192	6	4	12
AES256	8	4	14

There are four basic operations carried out in each round of the AES algorithm, they are
i) Sub-byte
ii) Shift row
iii) Mixed column
iv) Add Round Key
The last round of the encryption alone is different in a way that the mixed column operation will not be carried out. A 128-bit data block is divided into 16 bytes and these 16 bytes are mapped as 4X4 matrix and each entry are called as states. These states undergo all mathematical operation carried out in each rounds of the AES algorithm. Every State variable is to considered in the element of GF(2). Although there are different irreducible polynomials that could be used for GF (28), this AES algorithm uses $P(x) = x^8 + x^4 + x^3 + x + 1$ as its irreducible polynomials. Decryption can be done by the inverse process of encryption operation. The AES encryption and the equivalent decryption structures are shown in Fig.1.

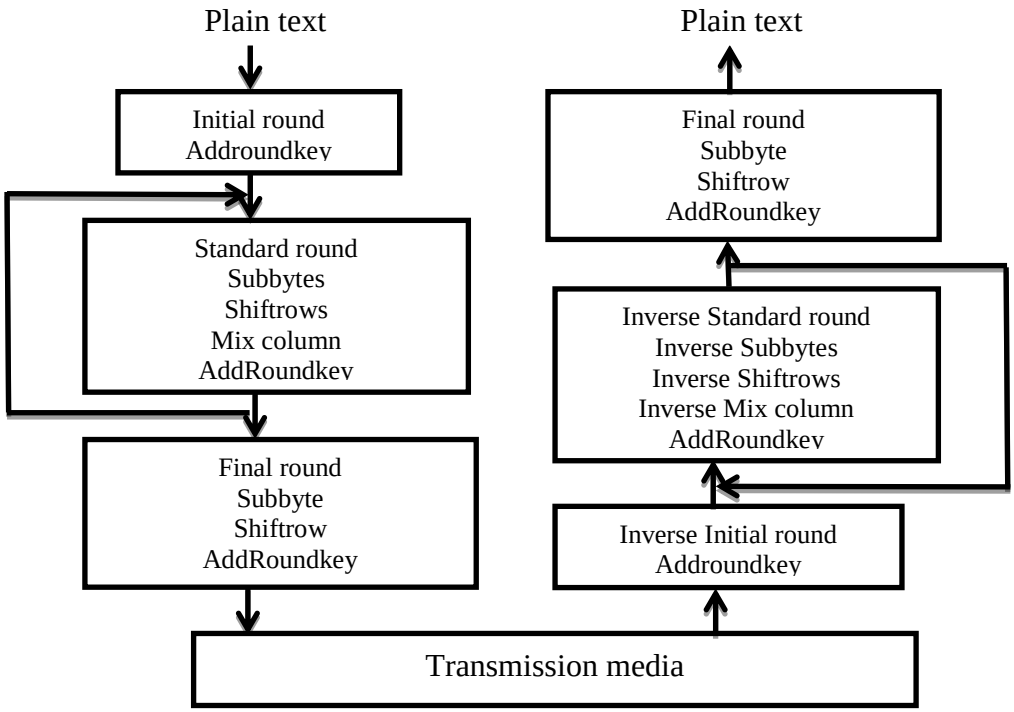

Fig.1 Encryption and Equivalent Decryption Structure

A. Sub bytes transformation

The Sub Bytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box which is invertible is constructed by composing two transformations:
1. Take the multiplicative inverse in the finite field GF (28), the element {00} is mapped to itself.
2. Apply the affine transformation over GF (2)
Similarly inverse sub bytes implemented by using inverse affine transform followed by multiplicative inverse.

B. Shift rows transformation

In the Shift Rows transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row is not shifted at all, the second row is shifted by one the third row by two, and the fourth row by three bytes to the left. In the InvShiftRows, the first row of the State does not change, while the rest of the rows are cyclically shifted to the right by the same offset as that in the Shift Rows transformation.

Fig.2 Shift rows transformation

C. Mixcolumns transformation

The MixColumns transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial a(x), given by $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. In hardware, the multiplication by the corresponding polynomial is done by XOR operations. In matrix form, the MixColumns transformation can be expressed as

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

Fig.3 Mixcolumns transformation

The InvMixColumns multiplies the polynomial formed by each column of the State with $a^{-1}(x)$ modulo x^4+1, where $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$.

In matrix form, the InvMixColumns transformation can be expressed by

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Fig.4 Inverse Mixcolumns transformation

D. Addround key

In the add round key step the 128 bit data is XORed with the sub key of the current round using the key expansion operation. The add round key is used in two different places one during the start that is when round r=0 and then during the other rounds that is when 1 ≤ round≤ Nr, where Nr is the maximum number of rounds. The formula to perform the add round key is S'(x) = S(x) ⊕ R(x)

S'(x) – state after adding round key
S(x) – state before adding round key
R(x) – round key

III. KEY EXPANSION

In the AES algorithm, the key expansion module is used for generating round keys for every round. There are two approaches to provide round keys. One is to pre-compute and store all the round keys, and the other one is to produce them on-the-fly. In this paper it has been implemented with first approach. The key expansion has three steps*:*

- Byte Substitution *subword()*
- Rotation *rotword()*
- XOR with RCON (round constant)

The input to key schedule is the cipher key K. Key expansion generates a total of Nb*(Nr + 1) words. The algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [wi], with i in the range 0 ≤ i < Nb (Nr + 1).

The subword() function takes a four byte input and applies the byte substitution operation and produces an output word. The rotword() takes a word [a0, a1, a2, a3] as input and performs a cyclic permutation to produce [a1, a2, a3, a0] as output word. The round constant word array rcon[i] is calculated using the below formula in finite field.

$$Rcon[i] = x^{(254+i)} \bmod x^8 + x^4 + x^3 + x + 1$$

The first Nk words of the expanded key are filled with the cipher key. Every following word w[i] is equal to the xor of previous word w[i-1] and the word Nk positions earlier w[i-Nk]. For words in positions that are a multiple of Nk, a transformation is applied to w[i-1] prior to the XOR, followed by an XOR with a round constant Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word rotword () and byte substitution subword (). But in key expansion of 256-bit cipher if Nk=8 and i-4 is a multiple of Nk then subword() function is applied to w[i-1] prior to the xor.

IV. PIPELINING

Rijndeal is a block cipher with a basic looping architecture whereby data is iteratively passed through a round function. The architecture used in this implementation is shown in figure 5.In this architecture, Data to be encrypted/Decrypted and key is passed to the 2:1 multiplexer In the starting the start signal is high mux performs xor between data and key, handover its output to further operation of AES.

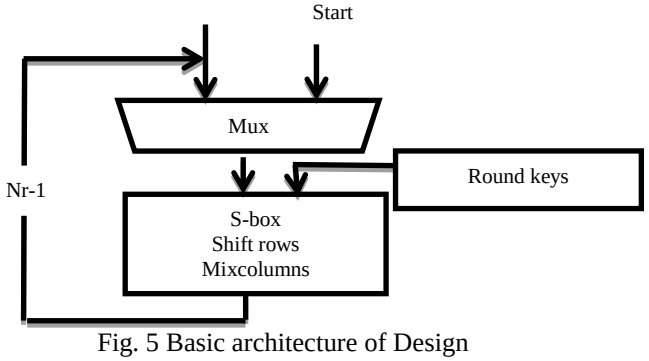

Fig. 5 Basic architecture of Design

In most of the application Speed is very important factor, In order to speed up the AES algorithm we can use pipelining architecture. If we do the pipelining, Hardware gets doubled for each pipelining stage. Here we can implement the pipelining in each round of AES. For every pipelining stage throughput will increase, simultaneously power and area also increases. Increase in Throughput at the cost of increased area and power becomes a major drawback in nanometer technology So we need to make a trade off while selecting the number of pipelining stages in the design. In this paper AES has been implemented with the different stages of pipelining, Throughput, Area and Power has been tabulated for all stages .Based on throughput requirement, number of pipelining stages can be restricted.

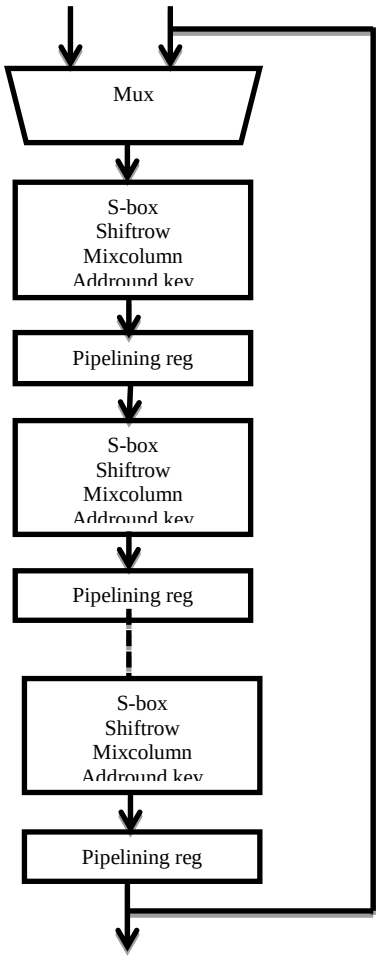

Fig. 6 pipelining architecture

In the above architecture, extra registers and hardware is repeated depending upon the number of pipelining stages so that, several blocks of data can be processed at a time as shown in figure 6.

V. IMPLEMENTATION RESULTS

The AES algorithm is implemented using Verilog hardware descriptive language and simulated using a Xilinx ISE 9.2 simulator. The algorithm is tested by encrypting and decrypting a single 128 bit block. Encryption simulation was successfully completed by the use of key expansion and transformations of shift Rows, sub bytes, mix columns, add round keys without pipelining stages shown in fig 7**.**

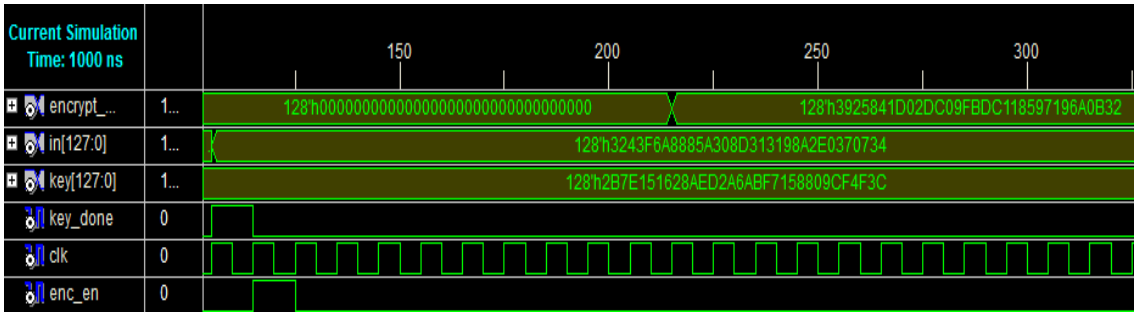

Fig.7 Encryption result

Decryption simulation was successfully completed by the use of key expansion and transformations of inverse shift Rows, inverse sub bytes, inverse mix columns, inverse add round keys shown in fig 8.

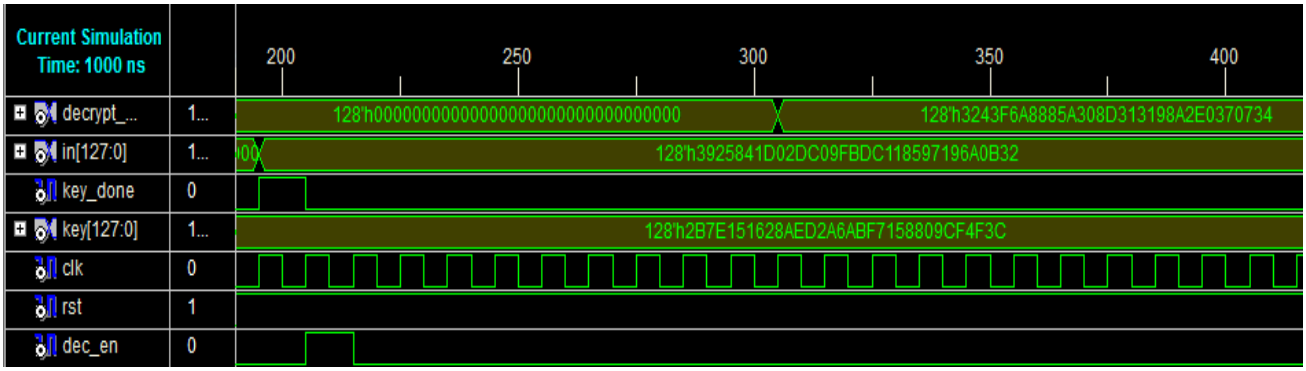

Fig.8 Decryption result

The Different pipelining stage implementation of AES algorithm has been synthesized in RTL compiler using TSMC's 180nm standard cells and corresponding variations in Throughput, Area, and Power for both Encryption and Decryption is tabulated .The synthesis results of an Encryption block shown in the TABLE II.

TABLE II.Synthesis results of encryption Block

Pipelining stages	Throughput In Mbytes/sec	Area		Power dissipation In nw		No of clock cycle
		No of Cells	Total area in um^2	Dynamic power in nw	Leakage power in nw	
1	267	14916	337214	86465435.266	4860.211	12
2	369	23541	473064	108021840.228	7650.091	13
3	457	32252	611795	151485105.881	10412.942	14
4	533	40642	748726	180421887.898	13095.544	15
5	600	49349	883871	212606469.901	15875.431	16
6	659	57869	1025875	256464500.954	18473.436	17
7	711	66849	1163162	282845003.502	21325.012	18
8	758	75195	1302502	322218174.381	24131.641	19
9	800	84268	1436433	341897824.509	27043.799	20

The Encryption block operating at an average frequency of 195 MHz for all pipelining stages and Decryption block operating at an average frequency of 226 MHz .The frequency has been calculated by synthesizing the design on virtex family. So, for throughput calculation 100 MHz clock has been considered as a reference. By looking at the TABLE II we can make out increase in number of pipelining stages leads to increase in area, power and Throughput. The synthesis results for Decryption block shown in the TABLE III.

TABLE III. Synthesis results of Decryption Block

Pipelining stages	Throughput	Area		Power dissipation In nw		No of clock cycle
		No of cells	Total area in um^2	Dynamic power in nw	Leakage power in nw	
1	267	18795	467339	165766605.201	5810.555	12
2	369	29432	643891	206566376.905	8932.516	13
3	457	40116	822253	241280230.729	12075.520	14
4	533	50629	1005641	281209992.645	15169.160	15
5	600	61279	1183127	307362796.144	18285.152	16
6	659	71949	1361745	344157094.231	21424.810	17
7	711	82534	1536218	410289205.593	24536.315	18
8	758	93478	1717743	447218740.465	27652.542	19
9	800	103926	1893739	497091453.253	30748.012	20

VI. CONCLUSION

In this paper, we presented a hardware implementation of pipeline AES architecture which includes both encryption and decryption and also gives an idea of restricting the number of pipelining stages in the design. The design is modelled

using Verilog HDL and simulated with the help of Xilinx ISE 9.2. Synthesis is done by using RTL Compiler v11.2. The encrypted cipher text and the decrypted text are analysed and proved to be correct for all the stages of pipelining.

ACKNOWLEDGMENT

I would like to thank my guide R.K Karunavathi and Co-ordinator Dr.Vijaya Prakash A.M for their encouragement, guidance and support by providing full time lab facility round the clock. I would like to thank my classmates for their support. Also I would like to thank Department of Electronics and Communication, Bangalore institute of technology for support.

REFERENCES

[1] Sumanth Kumar Reddy S, R.Sakthivel and P praneeth "*VLSI Implementation of AES Crypto Processor for High Throughput*" International journal of advanced engineering science and technologies, Vol No. 6, Issue No. 1, 022 – 026.

[2] M.Vanitha, R.Sakthivel and Subha, "*Highly Secured High Throughput VLSI Architecture for AES Algorithm*".

[3] L.Thulasimani and M.Madheswaran "*A Single Chip Design and Implementation of AES -128/192/256 Encryption Algorithms*," International Journal of Engineering Science and Technology, Vol. 2(5), 2010, 1052-1059.

[4] National Inst. Of Standards and technology, "Federal Information Processing Standard Publication 197,the advanced Encryption Standard (AES)," Nov. 2001

[5] Song J. Park, "*Analysis of AES Hardware Implementations*," .

[6] C.-P. Su, T.-F. Lin, C.-T. Huang, and C.-W. Wu, "*A high-throughput low-cost AES processor*," IEEE Commun. Mag., vol. 41, no. 12, pp. 86– 91, Dec. 2003.

[7] N. Sklavos and O. Koufopavlou, "*Architectures and VLSI Implementations of the AES-Proposal Rijndael*," IEEE Trans. on Computers, vol. 51, Issue 12, pp. 1454-1459, 2002.

[8] Kaur, Swinder,Vig and Renu , "*Efficient Implementation of AES Algorithm in FPGA Device*," in Conference on Computational Intelligence and Multimedia Applications, Nov 2007,pp. 179-187.

[9] Atul M. Borkar, Dr. R. V. Kshirsagar and Mrs. M. V. Vyawahare, " *Design of AES Algorithm using FPGA*," in Universal Association of Computer and Electronics Engineers.

[10] Mg Suresh and Dr.Nataraj.K.R,*"Area Optimized and Pipelined FPGA Implementation of AES Encryption and Decryption*," International Journal Of Computational Engineering Research Vol. 2 Issue. 7.

