ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The design and efficient hardware implementations for the Advanced Encryption Standard (AES) algorithm in order to attain better FPGA efficiency.

K.Chaitanya Kumar
M.Tech(VLSI & ES)
DRK Institute of Science And Technology
Bowrampet, Hyderabad.

Mr.M.Sivannarayana
Professor
DRK Institute of Science And Technology
Bowrampet, Hyderabad.

Abstract:
The Advanced Encryption Standard (AES) was endorsed by the National Institute of Standards and Technology in 2001. It was designed to replace the aging Data Encryption Standard (DES) and be useful for a wide range of applications with varying throughput, area, power dissipation and energy consumption requirements .Though they are highly flexible, FPGAs are often less efficient than Application Specific Integrated Circuits (ASICs); There have been many AES implementations that focus on obtaining high throughput or low area usage, but very little research done in the area of low power or energy efficient based AES; in fact, it is rare for estimates on power dissipation to be made at all.

This thesis introduces new efficient hardware implementations for the Advanced Encryption Standard (AES) algorithm. Two main contributions are presented in this thesis, the first one is a high speed 128 bits AES encryptor, and the second one is a new 32 bits AES design. In first contribution a 128 bits loop unrolled sub-pipelined AES encryptor is presented. In this encryptor an efficient merging for the encryption process sub-steps is implemented after relocating them. The second contribution presents a 32 bits AES design. In this design, the S-BOX is implemented with internal pipelining and it is shared between the main round and the key expansion units. Also, the key expansion unit is implemented to work on the fly and in parallel with the main round unit. These designs have achieved higher FPGA (Throughput/Area) efficiency comparing to previous AES designs.

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) = x8 + x4 + x3 + 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.

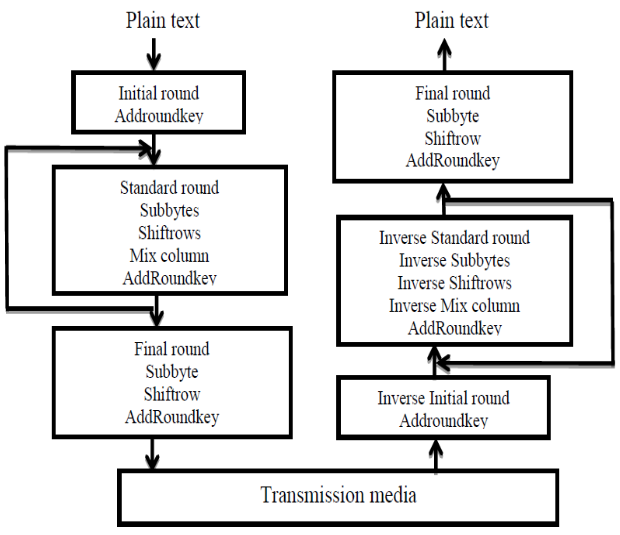


Fig.1 Encryption and Equivalent Decryption Structure

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.

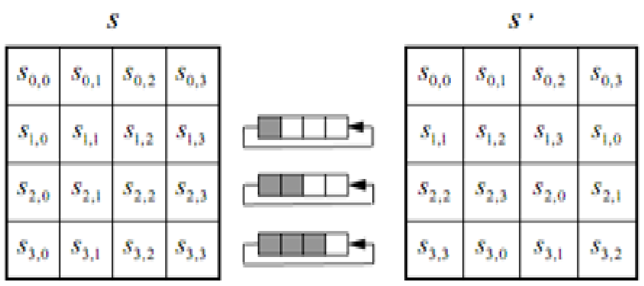

s
s'
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

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 $1 \leq$ round$\leq$ Nr, where Nr is the maximum number of rounds. The formula to perform the add round key is $S'(x) = S(x) \oplus R(x)$

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 $0 \leq i < Nb\,(Nr+1)$.

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.

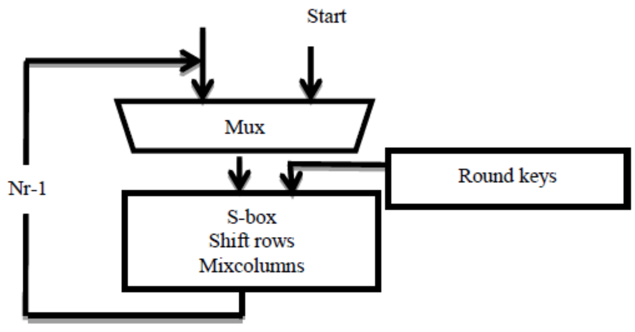


Fig. 5 Basic architecture of Design

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.

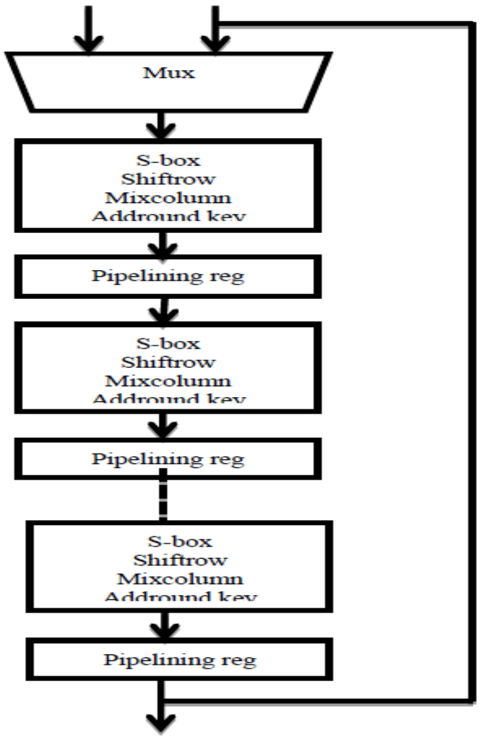
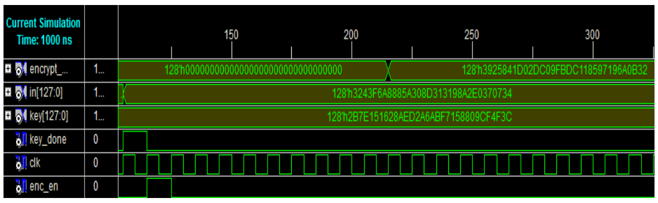


Fig. 6 pipelining architecture

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

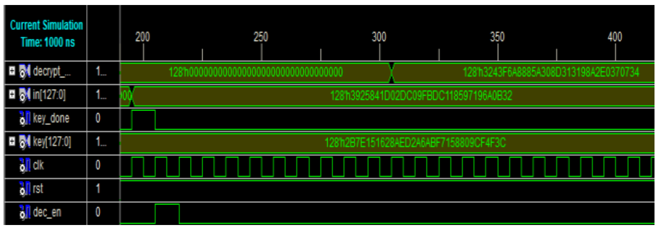
The AES algorithm is implemented using Verilog hardware descriptive language and simulated using a Xilinx ISE 9.2 simulator. The algorithm is tested by encrypting and decrypting a single 128 bit block. Encryption simulation was successfully completed by the use of key expansion and transformations of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



```
Current Simulation
Time: 1000 ns          150        200        250        300
encrypt_...   1...  128'h00000000000000000000000000000000 | 128'h3925841D02DC09FBDC118597196A0B32
in[127:0]     1...  128'h3243F6A8885A308D313198A2E0370734
key[127:0]    1...  128'h2B7E151628AED2A6ABF7158809CF4F3C
key_done      0
clk           0
enc_en        0
```

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.



```
Current Simulation
Time: 1000 ns          200        250        300        350        400
decrypt_...   1...  128'h00000000000000000000000000000000 | 128'h3243F6A8885A308D313198A2E0370734
in[127:0]     1...  00 | 128'h3925841D02DC09FBDC118597196A0B32
key_done      0
key[127:0]    1...  128'h2B7E151628AED2A6ABF7158809CF4F3C
clk           0
rst           1
dec_en        0
```

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 modeled 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.

REFERENCES

[1] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," 2001.

[2] S. K. Mathew, et al. "53 Gbps native GF(24)2 composite field AESencrypt/ decrypt accelerator for content-protection in 45nm highperformance microprocessors," IEEE Journal of Solid-State Circuits, vol. 46, no. 4, pp. 767-776, April 2011.

[3] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient high performance parallel hardware architectures for the AES-GCM," IEEE Transactions on Computers, vol. 61, no. 8, pp. 1165-1178, August 2012.

[4] S.-F. Hsiaso, M.-C. Chen and C.-S. Tu, "Memory-free low cost designs of Advanced Encryption Standard using common subexpression elemination for subfunctions in transformations," IEEE Transactions on Circuits and Systems, vol. 53, no. 3, pp. 615-626, March 2006.

[5] X. Zhang and K. K. Parhi, "High speed VLSI architectures for the AES algorithm," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 9, pp. 957-967, September 2004.

[6] S. K. Reddy S, R. Sakthivel and P. Praneeth, "VLSI implementation of AES crypto processor for high throughput," International Journal of Advanced Engineering Sciences and Technologies, vol. 6, no. 1, pp. 022-026, 2011.

[7] J. Chu and M. Benaissa, "Low area memory-free FPGA implementation of the AES algorithm," in 22nd International Conference on Field Programmable Logic and Applications, pp. 623-626, August 2012.

[8] T. A. Pham, S. H. Mohammad and H. Yu, "Area and power optimisation for AES encryption module implmentation on FPGA," in 18th International Conference on Automation and Computing, pp. 1-6, September 2012.

[9] J. M. Granado-Criado, M. A. Vega-Rodríguez, J. M. Sánchez-Pérez and J. A. Gómez-Pulido, "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," Integartion, the VLSI Journal, vol. 43, pp. 72-80, January 2010.

[10] K. Rahimunnisa, P. Karthigaikumar, S. Rasheed, J. Jayakumar and S. SureshKumar, "FPGA implementation of AES algorithm for high throughput using folded parallel architecture," Journal of Security and Communication Networks, vol. 5, no. 10, October 2012.

[11] M. El Maraghi, S. Hesham and M. A. Abd El Ghany, "Real-time efficient FPGA implementation of AES algorithm," in 26th Internation System on Chip Conference, pp. 203-208, September 2013.

[12] M. Fayed, M. W. El-Kharashi and F. Gebali, "A high speed fullypipelined VLSI architecture for real-time AES," in International Conference on Information and Communications Technology, December 2006.

[13] C. Paar, "Efficient VLSI architecture for bit-parallel computations in Galois field," Ph.D. dissertation, Essen, Germany, 1994.