A Fine Grained Architecture for Advanced Encryption Standard

[1]Anju Krishna, [2]Mr. P. Srinivasan M.E
[1]PG scholar, [2]Assistant Professor,
M.E- Embedded system technologies,
[1,2]Sun college of Engineering and Technology,
Erachakulam, Nagercoil, Tamilnadu, India

Abstract **-** A feasibility study for implementing the AES encryption algorithm in hardware achieving 500 Gbits/s is presented. The methodology followed in the process of obtaining the solution allowed us to reach a highly regular solution that is scalable. In recent years the internet has become one of the top communication medium used by the general public. More and more services are available through the internet. Managing sensitive information and the need for security has become a major concern for the users as well as the providers. Global security threats, cyber attacks to cripple a network connection or unauthorized intrusions to access restricted information are nowadays network security concerns all over the world. Encryption is a mean by which information can be safely exchanged.

Keyword: *AES; High Throughput; ASICs; High Speed Architectures.*

I. INTRODUCTION

Cryptography is the science of information and communication security. Cryptography is the science of secret codes, enabling the confidentiality of communication through an insecure channel. Cryptographic applications becoming increasingly more important in today's world of data exchange. Cryptography services are essential in order to provide the authentication, privacy, non-repudiation and integrity of private data being transmitted. It protects against unauthorized parties by preventing unauthorized alteration of use. In practice, cryptography is the task of transforming information into a form that is incomprehensible, but at the same time allows the intended recipient to retrieve the original information using the secret key, using most of the time a key. There exists certain cipher that doesn't need a key.

A. SECURITY MECHANISMS

A security mechanism is any process that is designed to detect,prevent,or recover from a security attack. The two main security mechanisms are:

- Encryption
- Decryption

An original message is known as the plain text, while the coded message are is called the cipher text. The two main encryption techniques are

i. Data encryption standard
ii. Advanced Encryption Standard

The National Institute of Standards and Technology (NIST) selected the Rijndael algorithm as the Advanced Encryption Standard (AES). The Rijndale was designed to have the following characteristics:

i. Resistance against all known attacks.
ii. Speed and code compactness on wide range of platforms.
iii. Design simplicity.

Because of the growing requirements for high speed secure communications, the application of AES algorithm in UART (Universal Asynchronous Receiver Transmitter) module which is a widely used in serial data communication to support full-duplex serial communication is proposed here.

B. STEPS IN AES ENCRYPTION

The following shows the AES encryption steps with the key expansion process. For encryption ,there are four basic transformations applied as follows:

1. SubBytes: The SubBytes operation is a nonlinear byte substitution. Each byte from the input state is

replaced by another byte according to the substitution box (called the S-box). The S-box is computed based on a multiplicative inverse in the finite field GF(28) and a bitwise affine transformation.

2. ShiftRows: In the Shift Rows transformation, the first row of the state array remains unchanged.The bytes in the second, third ,and fourth rows are cyclically shifted by one, two, and three bytes to the left, respectively.

3. MixColumns: During the MixColumns process, each column of the state array is considered as a polynomial over GF(2^8). After multiplying modulo x^4+1 with a fixed polynomial a(x), given by $a(x)=\{03\}x^3+\{01\}x^2+\{01\}x^1+\{02\}$ the result is the corresponding column of the output state.

4. AddRoundKey: A round key is added to the state array using a bitwise exclusive-or (XOR) operation. Round keys are calculated in the key expansion process.

II. TARGETED MANY-CORE ARCHITECTURE

According to Pollack's Rule, the performance increase of architecture is roughly proportional to the square root of its increase in complexity. Based on throughput requirement, number of pipelining stages can be restricted.

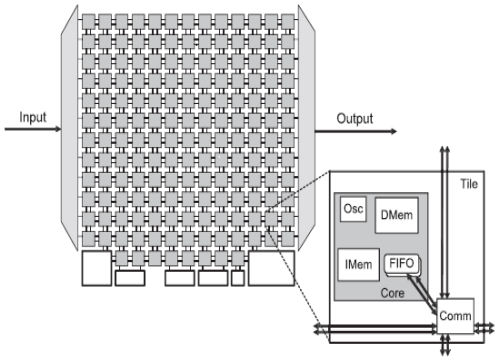

Figure 2.1. Computational platform of ASAP processor

A. MATHS PRELIMINARIES

The basic unit of processing in the AES algorithm is a byte. All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between the braces in the order (b7, b6, b5, b4, b3, b2, b1, b0). Several operations in AES are defined at byte level, with bytes representing elements in the finite field GF(2^8)[12]. Other operations are defined in terms of 4-byte words. Finite field elements can be added and multiplied , but these operations are different from those used for numbers. The following subsections introduce the basic mathematical concepts needed.

B. The Field GF(2^8)

The elements of a finite field can be represented in several different ways. For any prime power there is a single finite field, hence all representations of GF(2^8) are isomorphic.. A byte b, consisting of bits $b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$, is considered as a polynomial with coefficient in {0,1}:

$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$

C. Finite Field Addition

The addition of two elements in a finite field is achieved by "adding" the coefficients for the corresponding powers in the polynomials for the two elements. The addition is performed with the XOR operation (denoted by $\oplus$) - i.e., modulo 2 - so that $1\oplus1 = 0$, $1\oplus 0 = 1$, and $0 \oplus 0 = 0$. Consequently, subtraction of polynomials is identical to addition of polynomials. Alternatively, addition of finite field elements can be described as the modulo 2 addition of corresponding bits in the byte. For two bytes $\{a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0\}$ and $\{b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0\}$, the sum is $\{c_7\ c_6\ c_5\ c_4\ c_3\ c_2\ c_1\ c_0\}$, where each $c_i = a_i \oplus b_i$ (i.e., $c_7 = a_7 \oplus b_7$, $c_6 = a_6 \oplus b_6$, ...$c_0 = a_0 \oplus b_0$).

For example, the following expressions are equivalent to one another: (Polynomial notation) $(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$

(Binary notation) $\{01010111\} \oplus \{10000011\} = \{11010100\}$

(Hexadecimal notation) $\{57\} \oplus \{8E\} = \{D4\}$

D. Finite Field Multiplication

Finite field multiplication is more difficult than addition and is achieved by multiplying the polynomials for the two elements concerned and collecting like powers of x in the result. This situation is handled by replacing the result with the remainder polynomial after division by a special eight order irreducible polynomial, which for AES is $m(x) = x^8 + x^4 + x^3 + x + 1$. Since this polynomial has powers of x up to 8, it cannot be represented by a single byte and

will be written as either 1{00011011} or 1{1B} as indicated earlier

For example to find the result of $(x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1)$ $(x^6 + x^4 + x^2 + x + 1) \bullet x^7$

$= x^{13} + x^{11} + x^9 + x^8 + x^7$ $(x^6 + x^4 + x^2 + x + 1) \bullet x$

$= x^7 + x^5 + x^3 + x^2 + x(x^6 + x^4 + x^2 + x + 1) \bullet 1$

$= x^6 + x^4 + x^2 + x + 1$

$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

This intermediate result is now divided by $m(x)$ above. The final result can be the following

$x^7 + x^6 + 1 = \{C1\}$.

III. POLYNOMIALS & COEFFICIENTS GF(2^8)

Four term polynomials can be defined with coefficients that are finite field elements as: $a(x) = a_3x^3 + a_2 x^2 + a_1 x + a0$ where the four coefficients, each represented by a byte, will be denoted as a 32-bit word in the form [a3, a2, a1, a0]. With a second polynomial: $b(x) = b_3x^3 + b_2x^2 + b_1x + b0$ addition can be performed by adding the finite field coefficients of like powers of x, which corresponds to an XOR operation between the corresponding bytes in each of the words or an XOR of the complete 32-bit word values (note that the variable x here is different to that used in the definition of individual finite field elements). Multiplication is achieved by algebraically expanding the polynomial product and collecting like powers of x to give:

$c(x) = c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x + c0$

Where,

$c_0 = a_0 \cdot b_0$

$c_4 = a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3$

$c_1 = a_1 \cdot b_0 \oplus a_0 \cdot b_1$

$c_5 = a_3 \cdot b_2 \oplus a_2 \cdot b_3$

$c_2 = a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2$

$c_6 = a_3 \cdot b_3$

$c_3 = a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3$

With $\bullet$ and $\oplus$ representing finite field multiplication and addition (XOR) respectively. This result requires six bytes to represent its coefficients but it can be reduced modulo a degree 4 polynomial to produce a result that is of degree less than 4. In Rijndael the polynomial used is $(x^4 + 1)$ and reduction produces the following polynomial coefficients:

$d_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$

$d_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3$

$d_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus a_3 \bullet b_2 \oplus a_2 \bullet b_3$

$d_0 = a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$

If one of the polynomials is fixed, this can conveniently be written in matrix form as:

$$\begin{bmatrix} d3 \\ d2 \\ d1 \\ d0 \end{bmatrix} = \begin{bmatrix} a0 & a1 & a2 & a3 \\ a3 & a0 & a1 & a2 \\ a2 & a3 & a0 & a1 \\ a1 & a2 & a3 & a0 \end{bmatrix} \begin{bmatrix} b3 \\ b2 \\ b1 \\ b0 \end{bmatrix}$$

Because $(x^4 + 1)$ is not an irreducible polynomial, not all polynomial multiplications are invertible. For Rijndael, however, a polynomial that has an inverse has been chosen:

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

$a^{-1}(x) = \{0b\} x^3 + \{0d\} x^2 + \{09\} x + \{0e\}$

This transformation is used in MixColumn and InvMixColumn. Another polynomial that Rijndael uses has $a_0 = a_2 = a_3 = \{00\}$ and $a1 = \{01\}$, which is the polynomial x. Inspection of above will show that its effect is to form the output word by rotating the bytes in the input word so that $[b_3, b_2, b_1, b_0]$ is transformed into $[b_2, b_1, b_0, b_3]$, with bytes moving to higher index positions and the top byte wrapping round to the lowest position. Higher powers of x correspond to the other cyclic permutations of the four bytes within a 32-bit word. The Rot function that is used in the key expander corresponds to x_3.

A. SubBytes( ) Transformation

This step is carried out byte non-linear substitution through the AES's S-box, in order to achieve resistance to differential and linear attacks purposes[2]. The method is: every byte recorded as abcdefgh, abcd and efgh is the representative of the line and column respectively. The transformed byte will be replaced as the line abcd, column efgh correspond in the S-box though the look-up table, as follows (in decryption of the inverse S-box for the same treatment). The SubBytes( ) 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(2^8), the element {00} is mapped to itself.
2. Apply the following affine transformation (over GF(2)): $b_i' = b_i \oplus b_{(i+4) mod\ 8} \oplus b_{(i+5) mod\ 8} \oplus b_{(i+6) mod\ 8} \oplus b_{(i+7) mod\ 8} \oplus c_i$

for, $0 \le i < 8$ where bi is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c with the value {63} or {01100011}. Here and elsewhere, a prime on a variable (e.g., b') indicates that the variable is to be updated with the value on the right. In matrix form, the affine transformation element of the S-box can be expressed as:

$$\begin{bmatrix} b'0 \\ b'1 \\ b'2 \\ b'3 \\ b'4 \\ b'5 \\ b'6 \\ b'7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

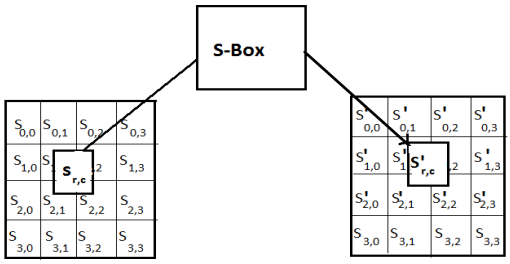

Figure 3.1. Sub bytes ( ) Transformation

For Affine Transformation the constant c is {63} or {01100011} and for Inverse Affine Transformation the constant c is {05} or {00000101}. The S-box used in the SubBytes( ) transformation is presented in hexadecimal form. For example, if $S_{1,1}$ = {53}, then the substitution value would be determined by the intersection of the row with index '5' and the column with index '3'.

The sub bytes transformation is performed on a state of 256 permutation of bytes.It is a non-linear substitution step where each byte is replaced with another according to the look up table. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF (2^8), known to have good non-linearity properties.Specifically, the ShiftRows( ) transformation proceeds as follows:

$$s'_{r,c} = s_{r,(c+shift(r,Nb)) mod\ Nb}$$

where the shift value shift(r,Nb) depends on the row number, r, as follows:

shift(1,4) = 1; shift(2,4) = 2; shift(3,4) = 3

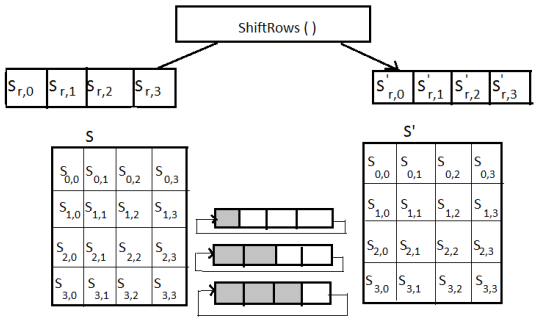

Figure 3.2 Shift rows ( ) Transformation

B. 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 $a(x)=\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.

Let $s(x) = a(x)$ XOR $s(x)$:

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

As a result of this multiplication, the four bytes in a column are replaced by the following:

$$s'_{0,c} = (\{02\}.s_{0,c}) \oplus (\{03\}.s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\}.s_{1,c}) \oplus (\{03\}.s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\}.s_{2,c}) \oplus (\{03\}.s_{3,c})$$
$$s'_{3,c} = (\{03\}.s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\}.s_{3,c})$$

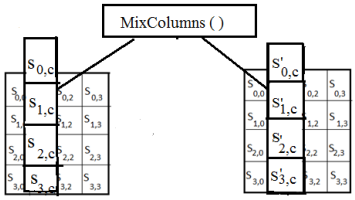

Figure 3.3 MixColumns( ) Transformation

In the Mix columns step, the four bytes of each column of the state are combined using an invertible linear transformation.Mix column function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with shift row, mix columns provide diffusion in the cipher.

C. AddRoundKey ( ) Transformation

In the AddRoundKey ( ) transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of *Nb* words from the key schedule. Those *Nb* words are each added into the columns of the State, such that

$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}]=[s_{0,c}, s_{1,c}, s_{2,c} s_{3,c}] \oplus [w_{round *Nb+1}]$

where [wi] are the key schedule words, and round is a value in the range 0 ≤ round ≤ Nr. The action of this transformation is illustrated in the figure given below, where l = round * Nb.

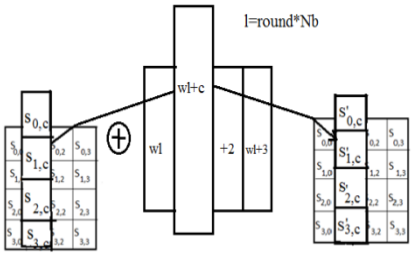

Figure 3.6. AddRoundKey( ) Transformation

IV. SIMULATION TOOL

Our work-steps Writing VHDL Code (Very high speed integrated circuit Hardware Descriptive Language), Simulating the code on "ModelSim, Synthesizing & Implementing (i.e. Translate, Map & Place and Route) the code on "Xilinx - Project Navigator. Tools used as,

- Xilinx ISE 8.1i
- ModelSim SE PLUS 5.7f

VHDL (VHSIC Hardware Description Language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits. VHDL can also be used as a general purpose parallel programming language.VHDL is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. A simulation program is used to test the logic design using simulation models to represent the logic circuits that interface to the design.

V. RESULT

The AES encryption and decryption is made to run on the Xilinx software and the waveform of the process is obtained. The clock and the reset value is given.In the Xilinx software input and output is given in the binary format.

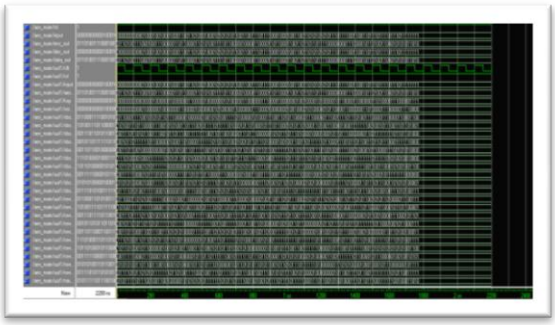

Figure 5.1 Output of AES encryption and decryption

The input is given in a binary sequence and the same sequence in the same format.One can design hardware in a VHDL IDE (for FPGA implementation such as Xilinx ISE, Altera Quartus, Synopsys Synplify or Mentor Graphics HDL Designer) to produce the RTLschematic of the desired circuit. After that, the generated schematic can be verified using simulation software which shows the waveforms of inputs and outputs of the circuit after generating the appropriate testbench. To generate an appropriate testbench for a

particular circuit or VHDL code, the inputs have to be defined correctly. For example, for clock input, a loop process or an iterative statement is required.

VI. FUTURE WORK

In future work, the parallel encryption process of AES encryption can be performed by using fine grained processors. Here the encryption process can be performed by using a number of processors. This will provide more efficiency and throughput.

REFERENCES

[1] J. Daemen and V. Rijmen, "*AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparison of 128, 192 and 256-bits Key"*, Springer, 2002.

[2] Johannes Wolkerstorfer, Elisabeth Oswald and Mario Lamberger, *"An Optimized S-Box for Advanced Encryption Standard (AES) Design "*, Institute for Applied Information Processing and Communication, Graz University of Technology, Springer-Verlag Berlin Heidelberg, 2002.

[3]Mozaffari-Kermani, M.; Reyhani-Masoleh, " *New Comparative Study Between DES, 3DES and AES within Nine Factors,"* Electro/Information Technology, 2009. eit '09. IEEE International Conference on , vol., no., pp.52-55, 7-9 June 2009.

[4] NIST, *"Data Integrity and Security in Cloud Environment Using AES Algorithm"*, FIPS PUBS 197, National Institute of Standards and Technology, November 2001.

[5] O P Verma, Ritu Agarwal, Dhiraj Dafouti, Shobha Tyagi in "*Image Encryption and Decryption using AES"*2011

[6] P. Rudra, K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, & P. Rahatgi. *" Chaotic variation of AES algorithm"* Proceedings of the Cryptographic Hardware in Embedded Systems (CHES). pp. 171-184, 2001.

[7] P. Rudra, K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, & P. Rahatgi *" Secured High throughput implementation of AES Algorithm."* Proceedings of the Cryptographic Hardware in Embedded Systems (CHES). pp. 171-184, 2001.

[8]Sai Praveen Venigalla, M. Nagesh Babu, Srinivas Boddu, G. Santhi Swaroop Vemana, in " *Implemenation of AES algorithm UART module for secured data transfer"*2012

[9]Sujatha Hiremath and M.S.Suma in "*Review paper on FPGA based implementation of advanced encryption standard(AES) algorithm"*2011

[10]Xiaona Lv, Liping Xu, *Enhanced AES Algorithm*, 2012