Implementation of the AES Realization Method On Reconfigurable Hardware

[1]Pramod Tidke, [2]Prof. Nilesh Mohota
[1] M.Tech Student, [2]Assistant Professor
[1]Electronics Department
[1]JD College of Engineering & Management, Nagpur, India

Abstract—This paper presents a VLSI implementation of the Advanced Encryption Standard (AES) algorithm. The AES is a Federal Information Processing Standard (FIPS), which is a cryptographic algorithm that is used to protect digital data. AES encryption and decryption requires a 128 bit wide input block and a 128 bit wide input key. Under the influence of a key schedule the input block is encrypted by transforming it in a unique way into a new block of same size. The major emphasis is on presenting a power and moreover an area optimized AES. For implementing AES Rijndael algorithm on FPGA we will choose VHDL as the design entry technique. Xilinx ISE Design Suite version 14.7 will be used for the Synthesis and Simulation of the code.

Keywords—AES, FIPS, Rijndael, FPGA, FSM, plaintext, ciphertext,VHDL,XILINX

I. INTRODUCTION

CRYPTOGRAPHY is the science and study of creating and using systems for communicating in secret via communication channels that are not secure. With the rapid development and wide application of computer and communication networks, the information security has aroused high attention. Continuous development is seen in Cryptographic Techniques. For a long time, the Data Encryption Standard (DES) was considered as a standard for the symmetric key encryption [2].DES has a key length of 56 bits. But due to the successful breaks on this small key length the National Institute of Standards and Technology (NIST) opened a formal call for algorithms in September 1997.A group of fifteen AES candidate algorithms were announced in August 1998. Next, all algorithms were subject to assessment process performed by various groups of cryptographic researchers all over the world. Five algorithms were selected and subjected to further analysis. Finally, on October 2, 2000, NIST declared that the Rijndael algorithm, invented by Joan Daemen and Vincent Rijmen, was the winner [2]. Rijndael can be specified with key and block sizes in any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits [1]. Therefore, the problem of breaking the key becomes more arduous.

The input and output for the AES algorithm consists of sequences of 128 bits. These sequences are referred to as blocks and the numbers of bits they contain are referred to as their length. The Cipher Key for the AES algorithm is a sequence of 128,192 or 256 bits. The basic unit of processing in the AES algorithm is a byte**,** which is a sequence of eight bits treated as a single entity [1]. Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes. Each row of a state contains N_b numbers of bytes, where N_b is the block length divided by 32.

The AES algorithm can be implemented by hardware as well as software. Software implementations cost the least resource but offer slowest process. Besides, growing requirements for high speed, high volume secure communications combined with physical security, hardware implementation of cryptography takes place.

II. OVERVIEW OF AES

The principle design of AES is based on substitution permutation network, which can take a block of the plaintext and the key as inputs. It is a round-based encryption algorithm [4]. The number of rounds, N_r, is 10, 12, or 14, when the key length is 128, 192 or 256 bits, respectively as shown in Table I.

There are four transformations provided in this algorithm:

Sub Bytes, Shift Rows, Mix Columns and Add Round Key. The first and the last rounds differ from other rounds in that there is an initial Add Round key transformation at the beginning of the initial round and Mix Columns transformation is not performed in the last round. Using the initial input key a key schedule module generates iterative keys which are different for every round.

Table I: Number of Rounds (a word is 32 bits)

	Key length (words)	Number of rounds (Nr)
AES - 128	4	10
AES - 192	6	12
AES - 256	8	14

A. Rjindael Mathematics

The computations required in this algorithm are mainly done in the finite field GF (2^8). A field is a commutative ring in which all non-zero elements have multiplicative inverses.GF stands for Galois Field []. The field elements are represented as polynomials. Here, the coefficients of the field elements are modulo 2 and an irreducible polynomial f(x) is chosen of degree 8. The irreducible polynomial that is used is:

$$F(x) = x^8 + x^4 + x^3 + x + 1.$$

The notations used to describe Rijndael are hexadecimal numbers. These numbers must be converted first to binary numbers such that the coefficients of the polynomial can be determined. For example the hexadecimal number 'D2' can binary be represented as '11010010'.This represents the polynomial:

$$x^7 + x^6 + x^4 + x.$$

B. Encryption Transformations

Sub Bytes is a nonlinear transformation, which computes the multiplicative inverse of each byte of the State in GF (2^8) followed by an affine transformation. Computation for each byte is stored in a lookup table which is called S-box [1].

Shift Rows is a transposition step where the last three rows of the state are shifted left cyclically a certain number of steps. For Nb = 4 Row 1 is shifted over 1 byte, Row 2 over 2 bytes, and Row 3 over 3 bytes.

Mix Columns is a mixing operation which operates on the columns of the state, combining the four bytes in each column. It is basically a matrix multiplication but in Galois field. Fig. 1 shows the matrix multiplication representation.

Add Round Key operation is a simple EXOR operation between the State and the Round Key. The Round Key is derived from the Cipher key by means of the key schedule module.

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Figure1: Mix Column Multiplication on column of a state

C. AES Encryption Process

Fig.2 gives the flowchart for the AES encryption Process.

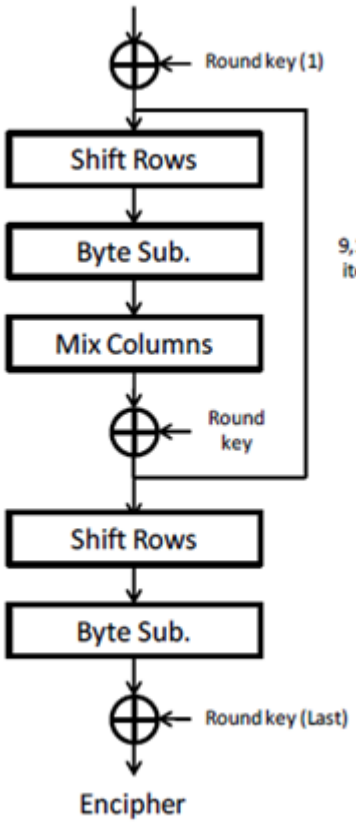

Figure 2: AES Encryption Flow

D. Decryption Transformations

Add Round Key is its own inverse function because the XOR function is its own inverse. The round keys have to be selected in reverse order.

Inv Shift Rows functions similar to Shift Rows, only the last three rows of the state are shifted right cyclically instead of left.

Inv Sub Bytes is done using a pre-calculated substitution table called Inv S-box. This table contains 256 numbers (from 0 to 255) and their corresponding values.

Inv Mix Columns is a mixing operation which operates on the columns of the state, combining the four bytes in each column. It is basically a matrix multiplication but in Galois field. Fig. 3 shows the matrix multiplication representation.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Figure 3:InvMix Column Multiplication on a column of a state

E. AES Decryption Process

Fig.4 gives the flowchart for the AES decryption Process.

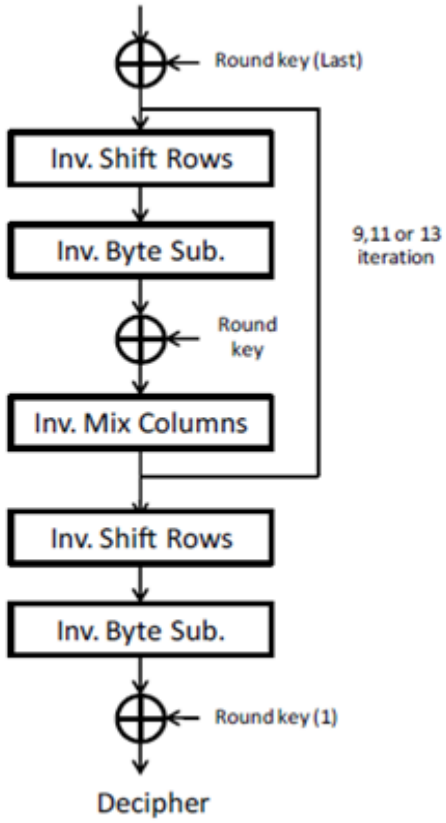

Figure 4: AES Decryption Flow

F. Key Schedule

Key scheduling is a process aimed at generating (Nr +1) round keys based on a single input key. This process consists of two phases called Key Expansion and Round Key Selection. The pseudo code for Key Expansion is shown below.

```
For i =0 to N k - 1
        W i = key i
end
for i = N k to 4(Nr + 1) - 1
        temp = wi-1
        if (i mod Nk = 0)
        temp = SubWord(RotWord(wi-1)) XOR Rcon(i / Nk)
        else if (Nk> 6 and i mod Nk = 4)
        temp = SubWord(wi-1)
        end if
```

wi = wi-Nk XOR temp

end

In the key expansion, the Sub Word applies Sub Bytes transformation to each of the four bytes in a word, while the Rot Word cyclically shifts each byte in a word one byte to the left. The Rcon is a constant word array, and only the leftmost byte in each word is nonzero [8].

III. DESIGN METHODOLOGY AND FPGA IMPLEMENTATION

Our goal is to implement AES on FPGA and its power performance. We will present the novel architecture for AES. The novel architecture is optimized for power using RTL clock gating technique. To create a novel architecture for AES we used FSM to execute the encryption and decryption using a common block. This resource sharing reduced the area effectively. A finite-state machine (FSM) is a mathematical model of computation used to design computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states defined for that machine. The machine is in only one state at a time. The state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition which is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition. Finite-state machines can model a large number of problems, among which are electronic design automation, language parsing, communication protocol design and other engineering applications.

IV. POWER OPTIMIZAION OF AES

As consumers continue to demand more functionality in smaller, more efficient devices, power optimization rules a hardware designer's life. For all applications, the total power consumption of complex SOCs presents a challenge. Low power methodology describes the most effective new techniques for managing dynamic and static power in SOC design.

Dynamic power is the power consumed when the device is active i.e. when signals are changing values. Static power is the power when device is powered up but no signals are changing values. Low power decision involves what techniques to use, when and where and on what sections of the chip.

Clock Gating is one of the effective Low power Techniques. A significant fraction of the dynamic power in a chip is in the distribution network of the clock. Clock nets have very high capacitances which results in 50% or more dynamic power dissipation. The flip flops receiving clock dissipate power even if input equals the output. So to reduce such dissipation, clocks should be turned off when they are not required. This is achieved without changing the function of the logic using modern design tools.

In the traditional synchronous design style used for most HDL and synthesis-based designs, the system clock is connected to the clock pin on every flip-flop in the design. This results in three major components of power consumption: 1) power consumed by combinatorial logic whose values are changing on each clock edge; 2) power consumed by flip-flops (this has a non-zero value even if the inputs to the flip-flops, and therefore, the internal state of the flip-flops, is not changing); and 3) the power consumed by the clock buffer tree in the design. RTL clock gating has the potential of reducing both the power consumed by flip flops and the power consumed by the clock distribution network [9].

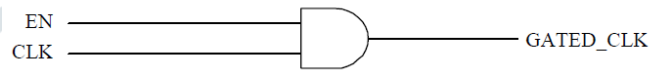

Figure 5: Latch free Clock Gating

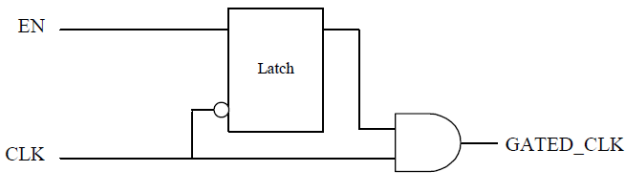

Figure 6: Latch based Clock Gating

Fig 6 represents simple latch free gating. Latch free gating makes the design sensitive to glitches. This makes it as the

least preferred clock gating technique. Another popular clock gating technique is the latch based clock gating. In latch based clock gating, latch is used as control element, it controls the Enable pin. In negative clock cycle, latch is allowed to reflect the change of Enable pin. In positive clock cycle, output of latch remains fixed. The high output of latch allows the clock to reach sequential logic.

The period in which, we can sense the change in Enable signal is called active period, and other period in which we cannot sense the change in Enable signal is called sleep period.

Fig. 8 represents a novel clock gating method i.e. the FF based Clock Gating. In FF based clock gating, FF is used as control element. When the negative edge of clock arrives, change of Enable will be reflected on FF output. If output of FF is high, clock is applied on sequential circuit. The sleep period is longer in FF based clock gating compared to Latch based clock gating. It means there is a greater chance to miss the change that happens on Enable signal [10].

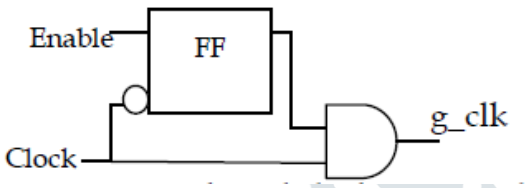

Figure 7: FF based Clock Gating

V. CONCLUSION

We will implement an efficient architecture of AES. The synthesis and simulation results will achieved using Xilinx ISE 14.7. For hardware implementation we will use ALtera FPGA.

REFERENCES

[1] J. Daemen and V. Rijmen, "AES Proposal: Rijndael. NIST AES Proposal," June 1998. Available at http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf.

[2] National Institute of Standards and Technology (U.S.), "Advanced Encryption Standard (AES)," Available at http://csrc.nist.gov/publications/drafts/dfips-AES.pdf.

[3] A. Rudra, P.K. Dubey, C.S. Jutla, V. Kumar, J.R. Rao, P. Rohatgi, "Efficient Rijndael encryption implementation with composite field arithmetic," *Lecture Notes in Computer Science* 2162 (2001) 171–184.

[4] J. Daemen, V. Rijmen, " AES proposal: The Rijndael Block Cipher, " Version 2 (Sept. 1999) pp. 1–45.

[5] Mr. Atul M. Borkar, Dr. R. V. Kshirsagar and Mrs. M. V. Vyawahare, "FPGA Implementation of AES Algorithm", International Conference on Electronics Computer Technology (ICECT), pp. 401-405, 2011 3rd.

[6] Leelavathi.G, Prakasha S, Shaila K, Venugopal K R, L. M. Patnaik "Design and Implementation of Advanced Encryption Algorithm with FPGA and ASIC", International Journal of Research in Engineering & Advanced Technology (IJREAT), Volume 1, Issue 3, June-July, 2013.

[7] A. Menezes, P. Van Oorschot, and S. Vanstone, "Handbook of applied cryptography", CRC press, New York, 1997, pp. 81-83.

[8] Xinmiao Zhang, *Student Member, IEEE,* and Keshab K. Parhi, *Fellow, IEEE* "High-Speed VLSI Architectures for the AES Algorithm", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 12, NO. 9, SEPTEMBER 2004.

[9] Frank Emnett and Mark Biegel ,"Power Reduction Through RTL Clock Gating", SNUG San Jose 2000.

[10] Dushyant Kumar Sharma ,"Effects of Different Clock gating Techniques on Design", International Journal of Scientific & Engineering Research Volume 3, Issue 5, May-2012.