CFA BASED PIPELINED S-BOX FOR AES ALGORITHM

[1]ARUNA V. PANICKER, [2]ANSIYA ESHACK

[1,2]Dept. of Electronics & Communication, KMEA Engineering College,
Edathala, Ernakulam, India
E-mail: [1]arunagopal.v@gmail.com, [2]aek.ec@kmeacollege.ac.in

Abstract - The increasing need for protecting data communication in computer networks has led to development of several cryptography algorithms. The Advanced Encryption Standard (AES) is a computer security standard issued by the National Institute of Standards and Technology (NIST) of US government intended for protecting electronic data. The AES cryptography algorithm is used to encrypt/decrypt blocks of 128 bits and is capable of using cipher keys of 128, 196 or 256 bits wide (AES128, AES196, and AES256).

In this paper, CFA based S-Box operation with the help of combinational logic circuits is presented. In addition to this, pipelining concept is introduced and 2 & 4-stage pipelining with CFA method is done. The design has been coded using Verilog HDL. All the results are synthesized and simulated using Xilinx and ModelSim respectively. The results indicate that when number of pipelined stages increases, system performance improves. Further there is a reduction in the gate delay as pipeline stages increases. It is also seen that the area of the device increases with the pipeline stages increases. Latency would be higher for the addition of pipelined registers.

Keywords — AES; CFA; pipelining;LUTs; FPGAs; HDL.

I. INTRODUCTION

Cryptography is a tool that has been widely used for many years. It is the research of mathematical procedure for secured communication in the presence of challenger and also it deals with the characteristics of information security such as confidentiality, data integrity, entity authentication and data source authentication.

The innovation of the Field Programmable Gate Array (FPGA) made by Xilinx in 1984 has provoked the engineers to make innovations in design, architecture and hardware implementation in many diverse fields such as Communication Devices, Digital Signal Processing (DSP), Cryptography and the System-on-Chip (SoC).

The hardware structure in FPGA is not fixed and is defined by the user itself. The functions performed by the logic cells are fixed and interconnections between them are determined by the user. It has high performance, minimizes the reliability, long term maintenance and increased time to market concerns compared to its software counterparts.

AES is more secure than other standards. It supports larger key sizes and is faster in both hardware and software.

The rest of this paper is organized as follows: Section 2 is about the overview of AES. Section 3 describes the CFA method for S-box transformations. Section 4 describes the proposed concepts of pipelining in CFA. Section 5 gives the simulation results of the system and the last session deals with the conclusion of the work.

II. OVERVIEW OF AES

AES is an iterated symmetric key block cipher which is based on the Substitution Permutation Network (SPN) structure, and it processes data blocks of 128 bits using a cipher key 128, 192, or 256 bits long. The cipher starts by initial key addition followed by 10, 12, or 14 repeated rounds of transformation for key lengths of 128, 192, or 256 bits, respectively, as shown in Fig 1. All operations are performed on a 4×4 array of bytes, called state. The round transformation consists of a sequence of four different transformations, called steps, which are SubBytes, ShiftRows, MixColumns, and AddRoundKey. These transformations are applied to the data block in the encryption procedure and in reverse order with inverse transformations in the decryption procedure. The round transformation in the last round of encryption or first round of decryption is applied without the Mix/Inverse Mix column transformation.

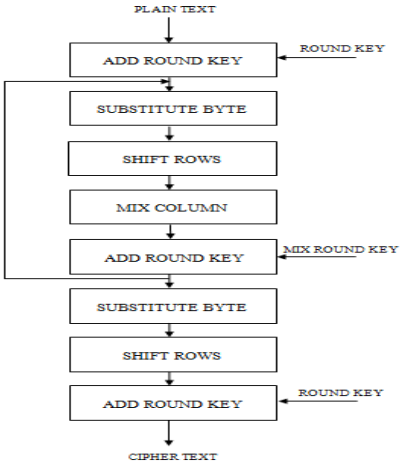


Fig.1 AES Encryption

In the encryption process in AES algorithm, following an initial addroundkey step, each round except the final round consists of four transformations: the SubByte, the ShiftRow, the

MixColumn and the AddRoundKey. The final round has only the SubByte, ShiftRow and addroundkey transformations Fig 1. The SubByte and InvSubByte are implemented as S-box and Inverse S-box respectively.

A. AES Encryption

The algorithm operates on a 128-bit data block and is executed in Nr-1 loop times. The key length is 128, 192 or 256 bits in length respectively. In this process there is an additional AddRoundKey transformation at the beginning of the first round and no MixCoulmns transformation is performed in the last round.

High-level description of the algorithm

1. KeyExpansions—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. InitialRound
    1. AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.
3. Rounds
    1. SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
    2. ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    3. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. AddRoundKey
4. Final Round (no MixColumns)
    1. SubBytes
    2. ShiftRows
    3. AddRoundKey.

B. AES Decryption

It is the reverse of encryption which inverse round transformations inorder to compute the original plaintext of an encrypted cipher text in reverse order. The round transformation of decryption uses the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes successively.

III. CFA METHOD FOR S-BOX OPERATION

Substitute byte is a non-linear byte substitution from the S-Box. It is done by using two methods. One is by using Look Up Table and the other with the help of Combinational Logic circuits using Composite Field Arithmetic (CFA) method. The values of the S-Box are obtained by taking multiplicative inverse over $GF(2^8)$ and Affine Transformation (AT). Composite field is an object of study in field theory. In this project combinational logic based Rijndael S-Box implementation for the SubByte transformation in the AES algorithm for FPGAs is considered. It is mainly used for small area occupancy and high throughput and hence pipelining can be accomplished with the help of combinational logic circuits compared to the ROM based LUT implementation.

While dealing with the LUT based method, they have greater delay which is unbreakable than that of other logic. An conventional method is to use combinational logic, which is faster than that of LUT. In this method sub bytes can be implemented by finding multiplicative inverse along with the affine transform. Similarly inverse sub bytes implemented by using inverse affine transform followed by multiplicative inverse can also be considered. Here multiplicative inverse is common for both the cases. By taking this as an advantage one can implement a single structure for both SubByte and its inverse. In this method it does not require hardware for the implementation of square operation.

In AES-128 algorithm, a block of data which is of l6 bytes is sent where each byte is applied for byte substitution. Here the calculation of multiplicative inverse in $GF(2^8)$ is very complex. Hence, the computation of multiplicative inverse can be done by decomposing the more complex $GF(2^8)$ into the lower order fields such as $GF(2^4)$, $GF(2^2)$, $GF(2^1)$ and these were achieved using the following irreducible polynomials.

$GF(2^2) \longrightarrow GF(2) : X^2+X+1$

$GF((2^2)^2) \longrightarrow GF(2^2) : X^2+X+\varphi$

$GF(((2^2)^2)^2) \longrightarrow GF((2^2)^2) : X^2+X+\lambda$

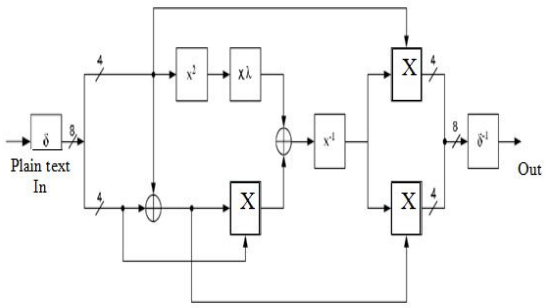


Fig 2. CFA based S-box using Combinational Logic Circuits

The CFA method for the S-box calculation is shown in Fig 2. The operations includes isomorphic mapping, squarer in $GF(2^4)$, multiplication with a constant λ in $GF(2^4)$, multiplicative inverse, XOR operation in $GF(2^4)$, multiplication operation, inverse isomorphic mapping to $GF(2^4)$ respectively.

C. Isomorphic and Inverse Isomorphic Mapping

Working out of the multiplicative inverse in composite fields cannot be directly applied to an element in $GF(2^8)$. That element has to be mapped to its composite field representation via an isomorphic

function, δ. Similarly, after performing the multiplicative inversion, the result will also have to be mapped back from its composite field representation to its equivalent in GF(2^8). it is done via the inverse isomorphic function, δ^{-1}. Both δ and δ^{-1} can be represented as an 8x8 matrix.

Let q be an element in GF(2^8), then the isomorphic mappings and its inverse can be written as δ*q and δ^{-1}*q, which is a case of matrix multiplication as shown below in (1) and (2), where q7 is the most significant bit and q0 is the least significant bit. Here the matrix multiplication can be translated to logical XOR operation.

$$\delta * q = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} q7 \\ q6 \\ q5 \\ q4 \\ q3 \\ q2 \\ q1 \\ q0 \end{bmatrix} \quad (1)$$

$$\delta^{-1} * q = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} q7 \\ q6 \\ q5 \\ q4 \\ q3 \\ q2 \\ q1 \\ q0 \end{bmatrix} \quad (2)$$

$$AT(a) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a7 \\ a6 \\ a5 \\ a4 \\ a3 \\ a2 \\ a1 \\ a0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad (3)$$

Affine and Inverse Affine Transformation is shown in matrix form above, where AT and AT^{-1} are the Affine Transformation were shown in (3) and (4) and its inverse while the vector a is the multiplicative inverse of the input byte from the state array. From this, it is cleared that both the SubByte and the Inverse SubByte transformation involve a multiplicative inversion operation. Thus, both transformations may actually share the same multiplicative inversion module in a combined architecture.

$$AT^{-1}(a) = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} a7 \\ a6 \\ a5 \\ a4 \\ a3 \\ a2 \\ a1 \\ a0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad (4)$$

D. Multiplicative Inverse in GF(2^8)

A formula which is used to compute the multiplicative inverse of q is shown below where q is an element of GF(2^4) such that $q^{-1} = \{q_3^{-1}, q_2^{-1}, q_1^{-1}, q_0^{-1}\}$. The inverses of the individual elements of q can be computed from (5) below.

$$q_3^{-1} = q3 \oplus q3q2q1 \oplus q3q0 \oplus q2$$
$$q_2^{-1} = q3q2q1 \oplus q3q2q0 \oplus q3q0 \oplus q2 \oplus q2q1$$
$$q_1^{-1} = q3 \oplus q3q2q1 \oplus q3q1q0 \oplus q2 \oplus q2q0 \oplus q1$$
$$q_0^{-1} = q3q2q1 \oplus q3q2q0 \oplus q3q1 \oplus q3q1q0 \oplus q3q0 \oplus q2 \oplus q2q1 \oplus q2q1q0 \oplus q1 \oplus q0 \quad (5)$$

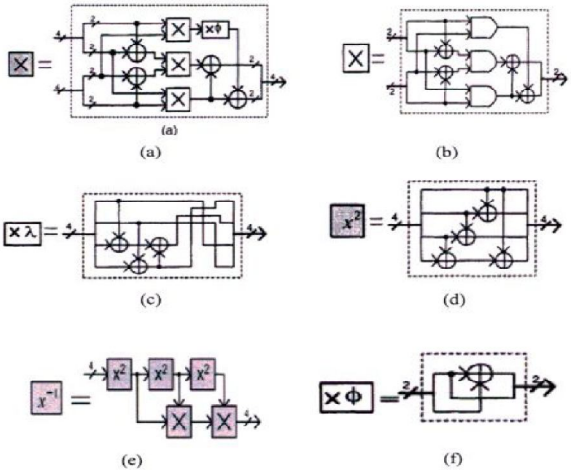


Fig 3. Sub blocks in CFA method (a)Multiplier in GF(2^4),(b) Multiplier in GF(2^2), (c) Constant Multiplier (xλ.), (d) Squarer in GF(2^2), (e) Inverse in GF(2^2), (f) Constant Multiplier (Xϕ)

Fig 3 shows the sub-blocks in CFA method for computing S-box. Fig 3(a) indicates the implementation of multiplication in GF(2^4). Here two 4 bit input is feed to produce a 4 bit output. It contains 4 XOR operations, multiplication with a constantφ and multiplication in GF(2^2). Multiplication in GF(2^2) is shown in Fig 3(b) Here two 2 bit input is provided to get 2 bit output and contains 3 AND gates and 4 XOR gates. Fig.3(c) shows the implementation of multiplication with a constantλ. Here also 4 bit input is feed to produce a 4 bit output and contains 3 XOR operations. Fig 3(d) shows the implementation of squarer in GF(2^4). 4 bit is feed as the input to squarer and it contains 4 XOR operations. Fig 3(e) shows the inversion module in GF(2^2). Fig 3(f) indicates the implementation of constantφ. Here 10 is obtained as 01 after this operation.

IV. PIPELINING CONCEPTS IN S-BOX

Pipeline is an important technique to increase the performance of a system. The basic idea is to overlap the processing of several tasks so that more tasks can be completed in the same amount of time. If a combinational digital circuit can be divided into stages, we can insert buffers (registers) at proper places and convert the circuit into a pipelined design.

The proposed structure of pipelined S-box is shown in Fig 4. Here GF(2^8) elements were feed as input to isomorphic mapping to the composite fields. Then the

8-bit divides equally and is driven to the multiplicative inversion module and then to inverse isomorphic mapping, finally to the Affine Transformations. Thus the output which is same as that of the LUT based S-box is obtained from this method. The only difference is that, in LUT based S-box method correct location of the values were directly obtained from the 16*16 table itself, whereas in CFA method, several operations were undertaken to obtain those values

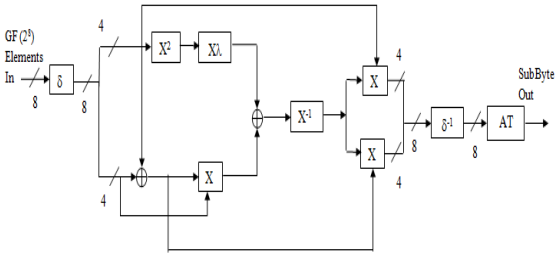


Fig 4. Proposed CFA Structure

The Fig 5 implies the proposed 2-stage pipelined structure for the CFA method. The double line in the figure shows the pipeline registers.

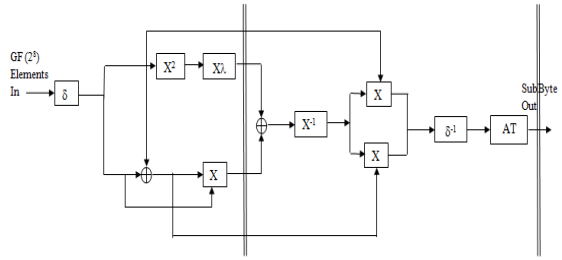


Fig 5. Proposed 2-stage pipelined CFA structure

For 2-stage structure, there were two pipelined registers. Here one register is used to store the output of the inversion module and other is used for the sub-byte output. 4-stage pipelined CFA encryption of AES is shown in Fig 6. Here there are four pipeline registers to accommodate the required functions.

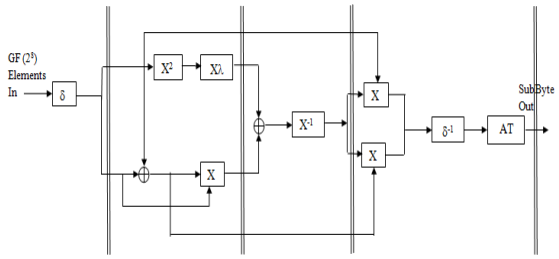


Fig 6. Proposed 4-stage pipelined CFA structure

V. IMPLEMENTATION RESULTS AND DISCUSSIONS

The AES algorithm with CFA method for the S-Box calculation is implemented in FPGA devices. Xilinx ISE 8.2*i* is used to synthesis the required design. The design has been coded using Verilog HDL.

A. Simulation Results

Here the simulation is carried out in ModelSim and the simulation results were shown in Fig 7 for the AES Encryption with CFA method. The results are shown in Hexadecimal format. This method is applied mainly for evaluating the performance of S-box operations. Instead of S-box with ROM based LUT method, here S-box is realized using combinational logic circuits.

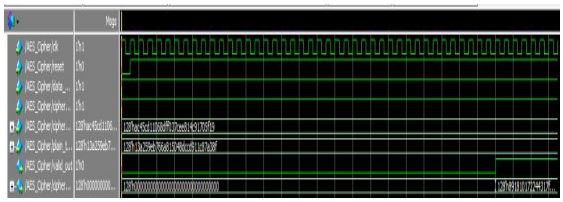


Fig 7 AES Encryption using CFA

In this method, the output is obtained after 44 clock cycles which is same as that of the conventional process. The only difference between the two processes is in the operations of the S-box.

The output of pipelining for the 2-stage CFA based encryption is shown in Fig 8. The plain text is: 128'h13a259eb766a815048dccd911c87a38f and the corresponding cipher key is 128'h ac45cd11068dff037cee814c91705f19. The valid out is set as 1 only after 44 clock cycles and it takes another 11 clock cycles to obtain the output which is 128'h891810172244317f705f24665a1c1de8.

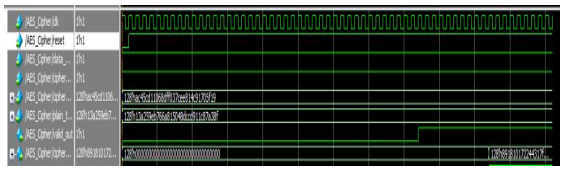


Fig 8 2-stage pipelining

The Fig 9 shows the encrypted output of 4-stage pipelining. The same procedure is followed for the 2-stage and 4-stage except an increase in the number of pipeline stages. The number of pipelined registers is increased from 2 to 4. The valid out is set after 44 clock cycle and it waits for an additional 31 clock cycles to get the required cipher text. Hence the difference between 2-stage and 4-stage pipelining lies mainly in the time required for generation of the output.

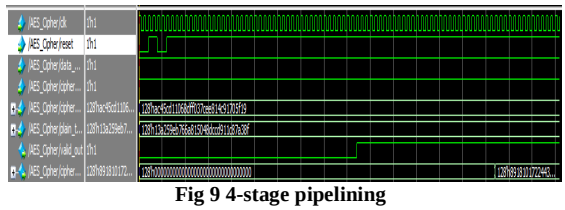


Fig 9 4-stage pipelining

The comparison of the parameters like number of registers used, delay, latency, etc when using the CFA method alone, 2-stage pipelined CFA & 4-stage pipelined CFA for the device XC5VLX50 with a speed of -3 is shown in Table 1.

TABLE 1 Comparison of stages using CFA method for the device XC5VLX50

Para meters	CFA	2-stage Pipeline	4-stage Pipeline	Avail able
Slice Register	9329	13777	16177	28800
Slice LUT	16529	17815	18015	28800
LUT FF pairs	18634	20898	21128	28800
Latency (cycles)	44	55	75	
Delay (ns)	5.128	4.119	2.929	

TABLE 2 Comparison of stages using CFA method for the device XC5VLX85

Para meters	CFA	2-stage Pipeline	4-stage Pipeline	Avail able
Slice Register	9329	13777	16177	51840
Slice LUT	16529	17815	18015	51840
LUT FF pairs	18634	20898	21128	51840
Latency (cycles)	44	55	75	
Delay (ns)	5.128	4.119	2.929	

For the two devices the total availability of the resources is different. From the tables it is seen that as pipeline stage increases, there is only an increase of 4% in the area used. It is also observed that gate delays decreases from with increase in pipeline stages. However, latency increases with the addition of pipelined registers.

Static timing report for both the devices with different speeds and at 2-stage pipelining is shown in Table 3. The minimum clock period that can be applied to the circuit is 4.026ns and 4.027 ns for XC5VLX50 and XC5VLX85 respectively. This gives a maximum clock frequency of 248MHz approximately for both the devices.

TABLE 3 Static timing report for 2-stage pipelining

Device	Speed (ns)		
	-3	-2	-1
XC5VLX50	4.026	4.587	5.350
XC5VLX85	4.027	4.549	5.334

Table 4 shows the static timing report for both devices under the different speeds and at 4-stage pipelining. The minimum clock period here is 2.725ns for the device XC5VLX50 and 2.691ns for the device XC5VLX85. This gives a maximum clock frequency of 366.97MHz and 371.60MHz respectively.

TABLE 4 Static timing report for 4-stage pipelining

Device	Speed (ns)		
	-3	-2	-1
XC5VLX50	2.725	3.073	3.629
XC5VLX85	2.691	3.017	3.567

It can be thus seen that higher clock frequencies can be accomplished by separating the S-box by placing more number of registers suited for pipelining. Thus pipelining increases the performance of the system. However, as more registers are added, additional clock cycles are required for the propagation of data from one register to the other. This explains the increase in latency.

CONCLUSION

The main focus here is to implement pipelining in S-box using CFA method. Here two stage and four stage pipelining is also performed. A comparison is done with the CFA method implemented and the 2 & 4 stage pipelined CFA. The devices used in the work are XC5VLX50 and XC5VLX85, which are from the Virtex-5 series of the Xilinx family.

It is observed that the system performance increases with an increase in the pipelining stages. This is due to reduction in minimum clock period that can be used for the operation. From the static timing report, the minimum clock period that can be applied to the circuit in a 2-stage pipeline is 4.026ns and 4.027 ns & in a 4-stage is 2.721ns and 2.691ns for XC5VLX50 and XC5VLX85 respectively. This gives a maximum clock frequency of approximately 248MHz in 2-stage & 366.97MHz and 371.60MHz in 4-stage.

Further there is a reduction in the gate delay as pipeline stages increases**.** It is also seen that the area of the device increases with the pipeline stages. This is due to increase in the number of registers used for the different stages. The increase in area between 2-stage and 4-stage however is only 4%. Also latency of the device increases with the addition of pipelined registers. This is because as pipeline registers are added, it takes additional clock cycles for the propagation of data from one register to the other.

It may be thus concluded that use of pipelining in CFA method for AES Encryption leads to increased system performance and a slight increase in the area of the device. This characteristic of high system performance allows it to be used in applications demanding quick and fast output. Further increase in number of pipeline stages can be employed for future works.

REFERENECES

[1] N.Anitha Christy and P.Karthigaikumar, 'FPGA Implementation of AES Algorithm using Composite Field Arithmetic', 2012 International Conference on Devices, Circuits and Systems (ICDCS).

[2] Veerendra Babu Dara, P. Sankara Rao,' Design and Implementation of Galios Field Based AES-256 Algorithm for Optimized Cryptosystem', International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Volume 3 Issue 11, November 2014.

[3] Chih-Peng Fan and Jun-Kui Hwang. 'FPGA implementations of high throughput sequential and fully pipelined AES algorithm', International Journal of Electrical Engineering, VOL.15, NO.6 PP 447-455 (2008).

[4] N. Radhika, Obili Ramesh, Priyadarshini, 'Design and Verification of Area-Optimized AES Based on FPGA Using Verilog HDL', International Journal of Engineering Trends and Technology (IJETT) – Volume 4 Issue 9- Sep 2013.

[5] Stallings, W. (2010) Network Security Essentials: Applications and Standards. 4th Edition, Prentice Hall, Englewood Cliff.

★★★