Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2017, 4(7): 541-548

Research Article ISSN: 2394 - 658X

Implementation of Area Efficient 128-bit Based AES Algorithm in FPGA

N Sivasankari [1], K Rampriya[1] and A Muthukumar [2]

[1]Department of ECE, Mepco Schlenk Engineering College, Sivakasi, India
[2]Department of ECE, Kalasalingam University, Krishnankoil, India
sivani.sivasankari@gmail.com

ABSTRACT

Advanced Encryption Standard is the most widely used Symmetric cipher today. The algorithm uses a combination of Exclusive-OR operations (XOR), octet substitution with an S-box, row and column rotations, and a Mix Column. An adjusted engineering for AES with upgraded key development and for the Mix Column/Inverse Mix Column operations are fixed to reduce the chip region. Both encryption and decryption processes were implemented in the single chip. The throughput of 38.65 Gbps for encryption/decryption in a single chip (Virtex-5) FPGA has been accomplished.

Keywords: Sub pipelining, Encryption/Decryption, True random number generator

---

INTRODUCTION

Expanding utilization of web and remote correspondence depends the safety efforts to secure the information transmission through open channel by the client. To fulfil this AES (Advanced Encryption Algorithm) was picked by NIST (National Institute of Standards and Technology) [12] in December 2001, But the equipment usage of AES calculation requires extensive range. Due to the power consumption and power attack AES algorithms and standards come as a replacement for DES. The AES algorithm has numerous applications including storage cards, cell phones, PC, ATM (Automated Teller Machine), advanced video recorders and monetary exchanges. Programming and equipment executions are some potential problems in AES calculation. In VLSI based processor implementation, for doing each round operation separate processors are available. These processors accelerate the procedure speed also. For the AES application in installed frameworks and smart cards, the region is constrained and for the media transmission the speed in the scope of Gbps is fundamental.

Many methodologies for the equipment execution of AES has been proposed [1-3], which incorporates pipelining, subpipelining and loop unrolling, Tiling, parallel sub-pipelining [4], C-slow retiming [5], partial rolling etc are used for throughput and frequency enhancement. The high speed architecture for hardware accomplishment for AES with the slice 11022 and throughput of 21.56 Gbps is presented in [6]. Stevens and Mohamed [6] proposed a fully synchronous, memory-based single-chip FPGA execution of AES with throughput of 0.1 Gbps. Mestiri *et al* [7] proposed a structural design which is robust to fault infusion attack for AES implementation with slice 532 and 2.3Gbps throughput. The customary pipelined approach is proposed in [3] and looked at the aftereffect of the same for different FPGA gadgets.

From [1], that the subpipelining design for the usage of AES will accomplish the speed and advanced region utilization. Henceforth, to evade the asset essentials for the single chip execution we go for the subpipelined engineering. In any case, the SubByte and Inverse SubByte changes are executed by LUT (Look Up Table) approach. Since, it is straightforward and quick [8], but, the LUT approach is not reasonable for every one of the changes in AES because of its unavoidable postponement. Subsequently, the composite field arithmetic [1] is utilized which maps the GF (2^8) to some isomorphic composite fields. Because of this the composite field arithmetic is connected for MixColumn (MC) and Inverse MixColumn (IMC)[2] changes. The design for the KeyExpansion is additionally proposed in our work. With these proposed structures for the execution of AES-128, we can accomplish the speed in Gbps go which possesses less region.

THE AES ALGORITHM

The AES calculation is a symmetric piece figure, which scrambles and decodes the information square of 128-bits from both the sender and the collector utilizing a solitary key. The information square length is settled to 128 bits and the key length can be shifted as 128, 192, or 256 bits. They are alluded as AES-128, AES-192 and AES-256. Notwithstanding that AES calculation is an iterative calculation. Every emphasis can be called as round and the aggregate number of rounds is 10, 12, or 14, when the key length is 128, 192, or 256 bits, separately. The 128-piece input information is partitioned into 16 bytes. These bytes are then taken as a 4X4 exhibit called the State S, and all the inward operations of the AES calculation are performed on the State. Every component of a state is referred to as $B_{r, c}$, since it is a byte, where r (0 to 3) characterizes the line and c (0 to 3) characterizes the section. Every component of the state is in 8-bit hexadecimal shape the 4X4 state grid is given as takes after

$$S = \begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{bmatrix} \quad (1)$$

The architecture of pipelined AES calculation for both encryption and unscrambling is given in Fig. 1. The Sub-pipelined stage for the pipelined AES architecture is appeared in Fig. 2. For this scenario, the key length is taken as 128 bit. Thus it comprises of 10 rounds. Each round expects the last round utilizations four changes and the last round has just three changes. The change in the AES calculation is clarified quickly as takes after.

SubBytes

A non-straight byte substitution that works freely on every byte of the State utilizing a substitution table (called the S-box).

ShiftRows

Plays out a roundabout moving operation on the columns of the State with various quantities of bytes (counterbalances). The principal push stays same and the second column is correct moved by onebyte. the third and fourth column is correct moved by two and three bytes separately. The InverseShiftRow is only a converse operation of the ShiftRows.It is finished by left moving the columns of the state in a similar way of the ShiftRows for decryption. Fig. 3 shows the idea of the ShiftRow and InverseShiftRow

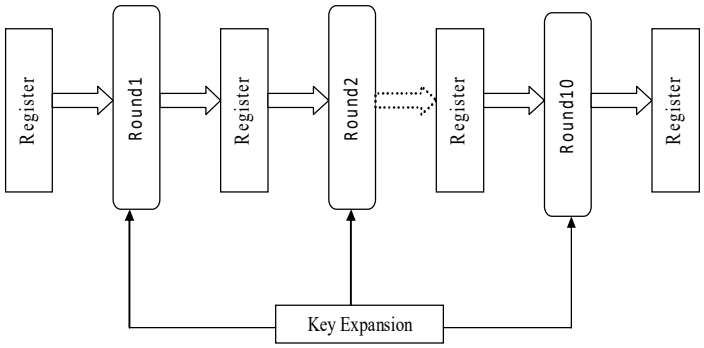

Fig.1 Fully-pipelined AES architecture

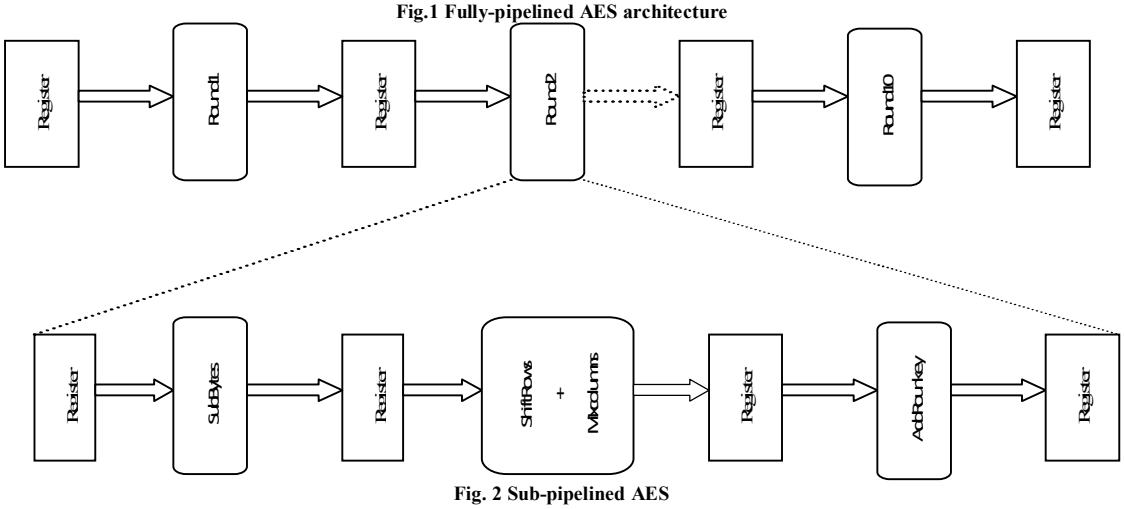

Fig. 2 Sub-pipelined AES

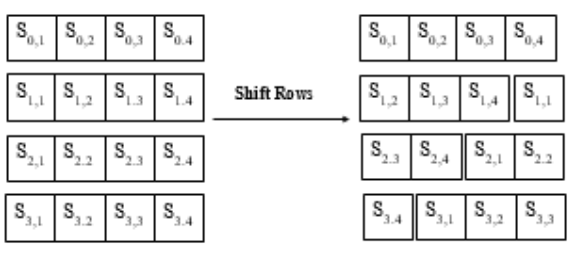

Fig. 3 (a) ShiftRows

Fig. 3 (b) Inverse ShiftRows

MixColumns(MC)

The blends the bytes in every segment by the duplication of the State with a settled polynomial modulo x^4+1. MC and IMC are actualized by performing framework duplication over Galois field i.e. GF (2^8) utilizing the irreducible polynomial $p^8+ p^4+ p^3+p+1$ [4]. The network utilized for blending of sections and reverse blending of segments are one of a kind chosen by FIPS [2]. GF (2^8) framework augmentation alongside the one of a kind lattices for both MC and IMC are delineated as

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} X \begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{bmatrix} = \begin{bmatrix} B'_{0,0} & B'_{0,1} & B'_{0,2} & B'_{0,3} \\ B'_{1,0} & B'_{1,1} & B'_{1,2} & B'_{1,3} \\ B'_{2,0} & B'_{2,1} & B'_{2,2} & B'_{2,3} \\ B'_{3,0} & B'_{3,1} & B'_{3,2} & B'_{3,3} \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} X \begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{bmatrix} = \begin{bmatrix} B'_{0,0} & B'_{0,1} & B'_{0,2} & B'_{0,3} \\ B'_{1,0} & B'_{1,1} & B'_{1,2} & B'_{1,3} \\ B'_{2,0} & B'_{2,1} & B'_{2,2} & B'_{2,3} \\ B'_{3,0} & B'_{3,1} & B'_{3,2} & B'_{3,3} \end{bmatrix} \quad (3)$$

AddRoundKey

It is a XOR operation that adds a round key to the State in every emphasis, where the round keys are produced amid the key extension stage.

Key Expansion(KE)

Routine used to create a progression of Round Keys from the Cipher Key for 10 rounds. Key created for each round is alluded as Roundkey. This Roundkey can be gotten by the accompanying calculation given in Table -1. Where the initial four words (h0, h1, h2, h3) are acquired from the key. Since the key is taken as 16 bytes (KE0 to KE15). Subsequently the initial four byte (KE0 to KE3) turns into a word h0 and the following four bytes are h1 and so on.

- RotWord—routine is like the ShiftRows change however it is connected to just a single column.
- SubWord—this routine is like Substitute Bytes change yet it is additionally connected to just a single line.
- RCon—is a 4-byte esteem in which the furthest right three bytes are constantly zero.

KeyExpansion ([KE_0 to KE_{15}], [h_0 to h_{43}])
{
for (i = 0 to 3)
$h_i = (KE_{4i} + KE_{4i+1} + KE_{4i+2} + KE_{4i+3})$
for (i = 4 to 43)
{
if (i mod 4≠0) $h_i = h_{i-1} + h_{i-4}$
else
{ t = SubWord (RotWord (h_{i-1})) XorRCon$_i$/4
$h_j = t + h_{i-4}$
}
}
}

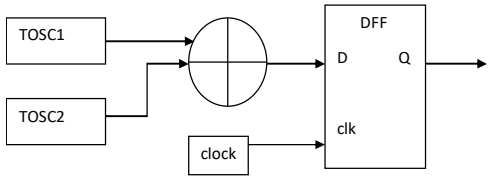

Fig. 4 The structure of proposed TRNG [9]

TRUE RANDOM NUMBER GENERATOR (TRNG)

To secure a processor it needs a mystery key for memory encryption and decoding. What's more, the mystery key ought to be unusual for aggressors. A genuine arbitrary number generator (TRNG) is used to create a mystery key for the AES engine.

Fig.4 demonstrates the structure of TRNG proposed in [8], which comprises of indistinguishable oscillators. The oscillators proposed by [3] alluded as tetrahedral oscillator. The design of proposed TRNG depends on examining two oscillators' frequencies utilizing D-flip flop which is appeared in Fig. 4. It contains two oscillators named (Tetrahedral Oscillator) TOSC1 and TOSC2, a XOR entryway and a D flip-flounder (DFF). The frequencies of the TOSC1 and

TOSC2 are XORed to give seed. At that point, the seed is given as contribution to the DFF and 20MHz clock recurrence is given. Presently the DFF go about as a sampler i.e., a low-recurrence oscillator (framework clock) tests the yield of high recurrence oscillator (seed).

CRYPTOGRAPHIC KEY GENERATION

The random bit can acquire from Fig.4 is in single bit. Be that as it may, the encryption and decryption of the AES engine requires 128 bit for mystery key. Subsequently to produce 128-bit key from single bit TRNG is applied from 128-bit shift register named as square as key generator. The 128-bit shift register is connected which comprises of 128 single bit flip-flops. Since the TRNG creates only one random bit for each cycle. Along these lines, it takes the shift register 128 clock cycles to store 128 arbitrary bits These 128 arbitrary bits are then associated with a 128-bit.

SIPO (Serial in Parallel Output) enlist, which yields them as a key. Notwithstanding the registers, there is a counter in the key generator. The counter checks from 0 to 127. At the point when the tally achieves 127, it sets a banner flag to 1, demonstrating that the key is prepared. After the key is perused, the banner flag is reset to 0 until 128 new bits are accessible. The design for the cryptographic key generator is appeared in Fig. 5

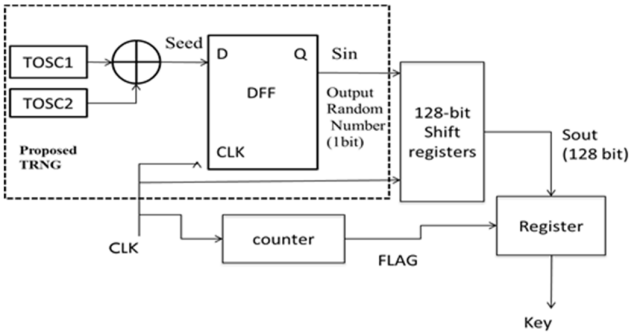

Fig. 5 Architecture for Cryptographic Key Generator

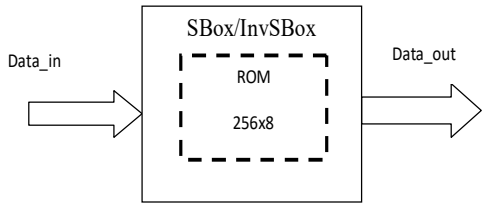

Fig.6 Implementation of SubByte/InvSubByte

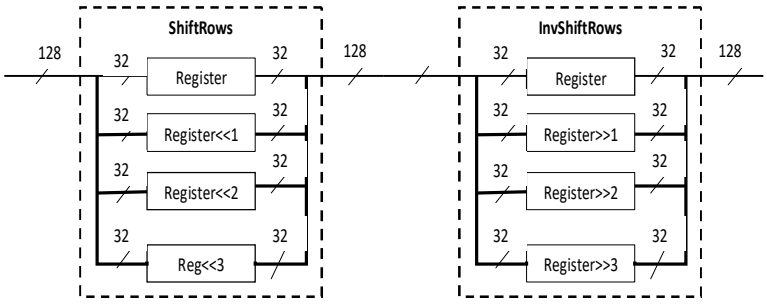

Fig.7 Implementation of ShiftRows/Inverse ShiftRows

HARDWARE IMPLEMENTATION ARCHITECTURE

The itemized structures for each of the changes in the AES calculation are exhibited in this area. To lessen zone and increment speed every change is upgraded for the execution. With respect to the key extension an effective design for sub-pipelined round units is introduced.

Implementation of SubByte/InverseSubByte

From [8], to stay away from usage of a lot of FPGA assets, S-Box can be actualized utilizing a look into table (LUT). Accordingly, the S-Box would require 8-bit wide 256 sections of ROM and is appeared in Fig. 6

Implementation of ShiftRows/InverseShiftRows

The 128-bit/state is separated into 32 bits from MSB and each 32 bit is given to the shift registers which do the left shifting by one byte, two byte and three byte. The Inverse ShiftRow block does the opposite operation of the ShiftRow block. The architecture of ShiftRows/Inverse ShiftRows is given in Fig.7.

Implementation of MixColumn/InverseMixColumn

Diverse structures have been proposed for the execution of the MixColumns/Inverse MixColumns in [2, 6 & 8]. From [1] a productive MixColumns execution architecture has been inferred which shares the substructure to the calculation of a byte and between the calculation of the four bytes in a segment of the State. From (2) the steady augmentation of

02 in Galois Field can be gotten by Fig. 8 appeared in (4) and it is named as 2 Multiplicative Galois Field (2MGF) which can be executed by 3 XOR gates and one XOR entryway at basic way.

$$B = b7p^7 + b6p^6 + b5p^5 + b4p^4 + b3p^3 + b2p^2 + b1p^1 + b0 \quad (4)$$

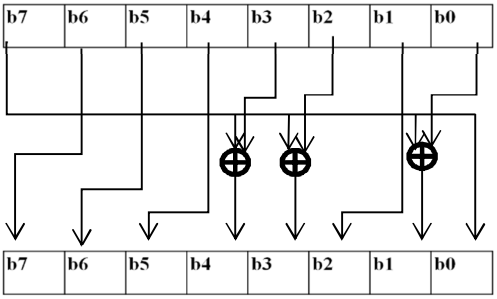

Fig. 8 Architecture of 2MGF

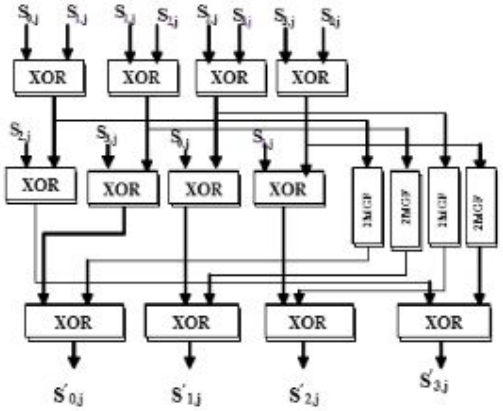

Fig.9 Architecture for MixColumn

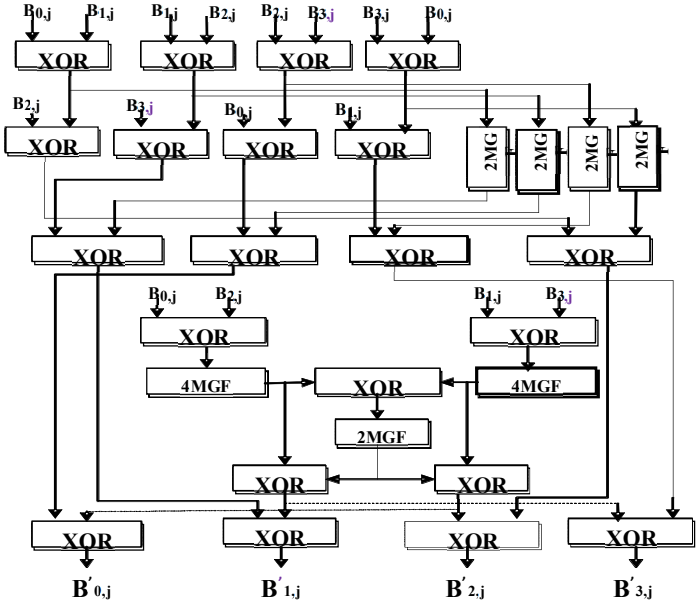

Fig. 10 Architecture of InvMixColumn

$$B\times 02 = \left(b6p^7 + b5p^6 + b4p^5 + (b3 \oplus b7)p^4 + (b2 \oplus b7)p^3 + b1p^2 + (b0 \oplus b7)p^1 + b7\right) \quad (5)$$

From (5) b0=b7. Hence the consistent increase of 03 can be gotten by 03= 02+01 which contains just 8 XOR gates. This is not as much as the quantity of XOR gates in

$$B \times 03 = (B \times 02) + 01 \quad (6)$$

Consequently, the design for mix column is given in Fig. 9 which processes one segment of the state for one clock cycle. Thus, the entire framework in the wake of blending the segment can be accomplished by 5 clock cycles. The engineering contains 108 XOR gates and 3XOR entryway for basic way. This engineering is accomplished by the improving (2) & (3) by substituting (5) & (6) appeared in (7).

$$\begin{cases} B'_{0,0} = [02.(B_{0,0} \oplus B_{1,0})] \oplus [B_{1,0} \oplus B_{2,0} \oplus B_{3,0}] \\ B'_{1,0} = [02.(B_{1,0} \oplus B_{2,0})] \oplus [B_{0,0} \oplus B_{2,0} \oplus B_{3,0}] \\ B'_{2,0} = [02.(B_{2,0} \oplus B_{3,0})] \oplus [B_{0,0} \oplus B_{1,0} \oplus B_{3,0}] \\ B'_{3,0} = [02.(B_{0,0} \oplus B_{3,0})] \oplus [B_{0,0} \oplus B_{1,0} \oplus B_{12,0}] \end{cases} \quad (7)$$

With respect to the converse mixcolumn from (3) the consistent augmentation of 0E, 09, 0D, 0B can be determined by part the Galois Field technique [6].

$$\left.\begin{aligned} 0D &= 04 + 04 + 02 + 02 + 01 \\ 0E &= 04 + 02 + 04 + 02 + 04 \\ 09 &= 04 + 02 + 04 \\ 0B &= 02 + 04 + 01 + 04 \end{aligned}\right\} \tag{8}$$

$$04 \times B = p^2 B = \left(b7p^9 + b6p^8 + b5p^7 + b4p^6 + b3p^5 + b2p^4 + b1p^3 + b0p^2\right)$$

$$= b5p^7 + b4p^6 + (b3+b7)p^5 + \left((b7+b6)+b2\right)p^4 + (b1+b6)p^3 + (b0+b7)p^2 + (b6+b7)p + b6 \tag{9}$$

The consistent augmentation of 04 can be gotten by actualizing two serially connected 2MGF squarean it is allured by 4MGF. Subsequently it comprises of 5 XOR gates with 2 XOR gates in basic way. Thus, by substituting (8) & 9 in (3) and after improvement the auxiliary outline for IM is acquired in (10) and it showed up in Fig. 10where the top module is like Fig. 9. Consequently Fig. 10 is sufficient to execute the MC/IM for encryption/decryption joint design.

$$\left\{\begin{aligned} B'_{0,c} &= \left[\begin{matrix}02\times(B_{0,c}\oplus B_{1,c})\oplus\\(B_{2,c}\oplus B_{3,c}\oplus B_{1,c}\end{matrix}\right]\oplus\left[\begin{matrix}02\times(04\times(B_{0,c}\oplus B_{2,c})\\\oplus 04\times(B_{1,c}\oplus B_{3,c}))\\\oplus 04\times(B_{0,c}\oplus B_{2,c})\end{matrix}\right] \\ B'_{1,c} &= \left[\begin{matrix}02\times(B_{1,c}\oplus B_{2,c})\oplus\\(B_{2,c}\oplus B_{3,c}\oplus B_{0,c}\end{matrix}\right]\oplus\left[\begin{matrix}02\times(04\times(B_{0,c}\oplus B_{2,c})\\\oplus 04\times(B_{1,c}\oplus B_{3,c}))\\\oplus 04\times(B_{1,c}\oplus B_{2,c})\end{matrix}\right] \\ B'_{2,c} &= \left[\begin{matrix}02\times(B_{2,c}\oplus B_{3,c})\oplus\\(B_{0,c}\oplus B_{3,c}\oplus B_{1,c}\end{matrix}\right]\oplus\left[\begin{matrix}02\times(04\times(B_{0,c}\oplus B_{2,c})\\\oplus 04\times(B_{1,c}\oplus B_{3,c}))\\\oplus 04\times(B_{0,c}\oplus B_{2,c})\end{matrix}\right] \\ B'_{3,c} &= \left[\begin{matrix}02\times(B_{0,c}\oplus B_{3,c})\oplus\\(B_{2,c}\oplus B_{0,c}\oplus B_{1,c}\end{matrix}\right]\oplus\left[\begin{matrix}02\times(04\times(B_{0,c}\oplus B_{2,c})\\\oplus 04\times(B_{1,c}\oplus B_{3,c}))\\\oplus 04\times(B_{1,c}\oplus B_{3,c})\end{matrix}\right] \end{aligned}\right. \tag{10}$$

Implementation of Key Expansion

In the past works the round keys are produced on the fly. Here the round keys are created previously and put away in memory. Thus there is no deferral for decryption since the round keys are perused out from the memory. The structural design for Key Expansion is appeared in Fig. 11. The reconfigurability [10] in our design is included by control input. Both encryption and decryption logics can be done by the same units, but in opposite manner.

Implementation of Encryption/Decryption in Single-Chip

The architecture for both encryption and decryption which proficiently plays out the encryption, decryption for 128-bit and 128-bit key encryption and it is appeared in Fig. 12. This design [11] can go about as either encryption unit or decryption unit relying on the control input. On the off chance that the control information is low it plays out the encryption and the other way around. The extended keys are put away in the memory which can be gotten to by the control contribution concurring the comparing control unit. The control signal of'1' indicates the operation selected for the processor is encryption.'0' indicates the process of decryption.

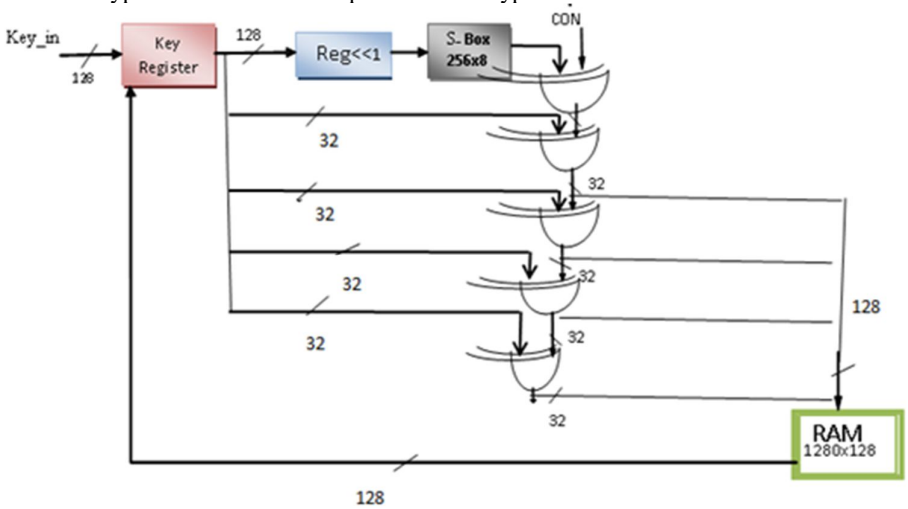

Fig. 11 Architecture for Key Expansion

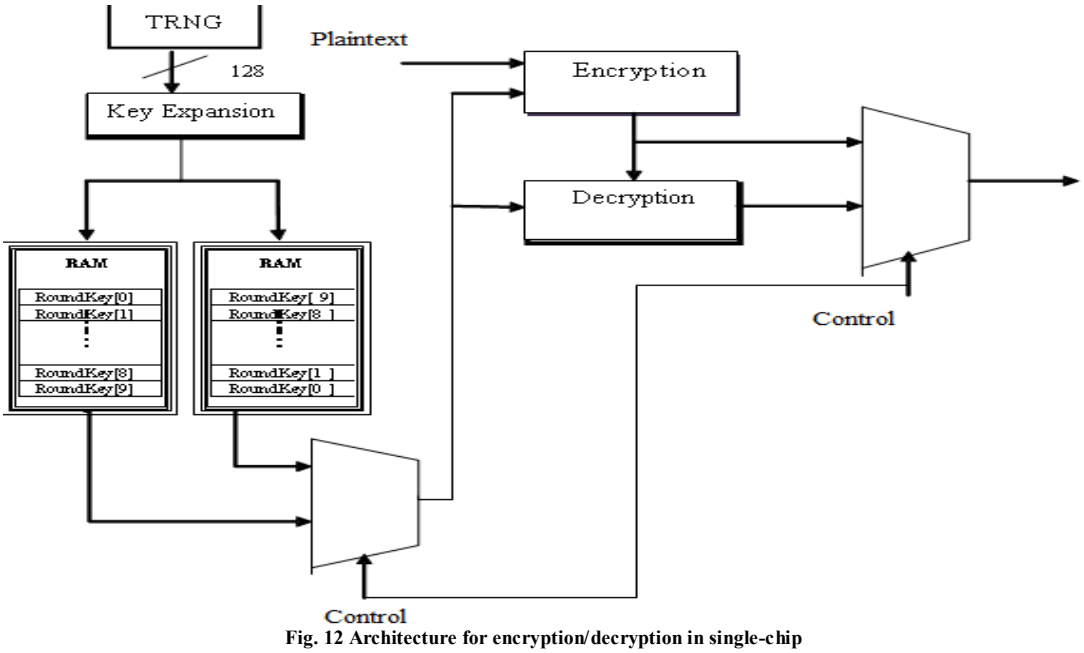

Fig. 12 Architecture for encryption/decryption in single-chip

RESULTS AND DISCUSSION

The proposed design for territory effective AES-128 is executed in XC5VLX50T. The simulations are done by Modelsim-Altera 6.4a (Quards-II 9.0) utilizing Verilog HDL and it is synthesized in Xilinx ISE Design Suite 14.6. The proposed strategy can perform encryption and decryption at a most extreme clock recurrence of 175.452 MHz accomplishing a throughput of 38.65 Gbps. Because of the asset complexities, the equipment uses and also the throughput of this proposed technique has dropped to half of what has been accomplished with encryption or decryption as an individual unit. These outcomes are appeared in Table -2. The throughput can be accomplished by the critical path delay for our proposed method is 5.700ns. Our method based on memory achieves high throughput. However, the latency is low. Table -3 shows the details of encrypted data and decrypted data. The device utilization summary for the proposed model is appeared in Table-4. The comparison of the proposed method and the previous method is shown in Table -5.

Table- 2 Throughput Comparison

Method	Throughput(Gbps)
Encryption	41.59
Decryption	41.05
Encryption/ Decryption	38.65

Table -3 Encryption/Decryption Result

I/O	Value
Input data	00041214120412000C00131108231919
Encrypted data	BC028BD3E0E3B195550D6DFBE6F18241
Decrypted data	00041214120412000C00131108231919

Table -4 Utilization Summary

Method	Slice	LUT	CPU Memory Usage
Traditional Pipelined approach [4]	21,969	23,704	326044
Proposed Encryption/ Decryption	16754	17,251	247844

CONCLUSION

The proposed territory efficient AES-128 for both encryption and decryption has been actualized in a solitary chip (FPGA-XC5VLX50T). It performs both (encrypt/decrypt) operation in a solitary chip with low asset use and the high throughput of 38.65Gbps. By taking after the proposed strategy AES-192 and AES 256 can likewise be executed, this is assigned for future work. This proposed strategy is strong to physical attacks.

REFERENCES

[1] Xinmiao Zhang and KK Parhi, High-Speed VLSI Architectures for the AES Algorithm, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **2004,** 12(9), 957-967.

[2] SR Huddar, SR Rupanagudi, R Ravi, S Yadav and S Jain, Novel Architecture for Inverse Mix Columns for AES using Ancient Vedic Mathematics on FPGA, *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Mysore, **2013**,1924-1929.

[3] N Weaver, The Effects of Datapath Placement and C-slow Retiming on Three Computational Benchmarks, *IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, USA, **2002.**

[4] K Rahimunnisa and P Karthigaikumar, PSP: Parallel Sub-Pipelined Architecture for High Throughput AES on FPGA and ASIC, *Central European Journal of Computer Science*, **2013**, 3 (1), 173-186.

[5] JM Szefer, Wei Zhang and Yu-Yuan Chen, Rapid Single-Chip Secure Processor Prototyping on the Open SPARC FPGA Platform, *22nd IEEE International Symposium on Rapid System Prototyping*, Karlsruhe, **2011**, 38-44.

[6] K Stevens and OA Mohamed, Single-Chip FPGA Implementation of a Pipelined, Memory-Based AES Rijndael Encryption Design, *Canadian Conference on Electrical and Computer Engineering,* **2005**, 1296-1299.

[7] H Mestiri, N Benhadjyoussef, M Machhout and R Tourki, An FPGA Implementation of the AES with Fault Detection Countermeasure, *International Conference on Control, Decision and Information Technologies (CoDIT)*, Hammamet, **2013**, 264-270.

[8] Z Liu, L Li and X Zou, A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards, *IEEE Transactions on Circuits and Systems II*, **2016,** 63 (6), 608- 612.

[9] D Kotturi, Seong-Moo Yoo and J Blizzard, AES Crypto Chip Utilizing High-Speed Parallel Pipelined Architecture, *IEEE International Symposium on Circuits and Systems*, **2005**, 5, 4653-4656.

[10] F Rodriguez-Henriquez, NA Saqib, A Díaz Pérez and CK Koc, *Cryptographic Algorithms on Reconfigurable Hardware*, first edition, *Springer Series on Signals and Communication Technology*, *US*, **2007,**

[11] Philippe Bulens, François-Xavier Standaert, Jean-Jacques Quisquater, Pascal Pellegrin and Gaël Rouvro, Implementation of the AES-128 on Virtex-5 FPGAs, *International Conference on Cryptology in Africa*, **2008**.

[12] Morris J Dworkin, Elaine B Barker, James R Nechvatal, James Foti, Lawrence E Bassham and James F Dray, *Advanced Encryption Standard*, Federal Information Processing Standards Publication 197 web. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf, **2001.**