# HARDWARE IMPLEMENTATION OF AES ENCRYPTION AND DECRYPTION FOR LOW AREA & POWER CONSUMPTION

**Pritamkumar N. Khose[1], Vrushali G. Raut[2]**

[1]*Dept. of Electronics and Telecommunications, Sinhgad College of Engineering, Pune, India*
[2]*Dept. of Electronics and Telecommunications, Sinhgad College of Engineering, Pune, India*

## Abstract

*An AES algorithm is implemented on FPGA platform to improve the safety of data in transmission. AES algorithms can be implemented on FPGA in order to speed data processing and reduce time for key generating. We achieve higher performance by maintaining standard speed and reliability with low area and power. The 128 bit AES algorithm is implements on a FPGA using VHDL language with help of Xilinx tool.*

***Keywords***— *Advanced Encryption Standard (AES), Field-Programmable Gate Array (FPGA), VHSIC Hardware Description Language (VHDL)*

----------***----------

## 1. INTRODUCTION

There is increasing need of information data in Computer Network and Communication Technology. This data is handled by public networks and it is vulnerable. So cryptography becomes important for such sensitive data which should be kept secure and safe against automated spying or hacking.

AES can be implemented in software or hardware but, hardware implementation is more suitable for high speed applications in real time. Main goal AES hardware implementation is high throughput design and low-area design work at highest operating frequency. The latter devotes most efforts to minimize size of the design and lower the power consumption.

The rest of paper is organized as follows. Section II will present a brief overview of AES and previously proposed existing work done. In section III, a proposed work contains Sbox architecture, Key Expansion module, and AES encryption & decryption crypto core. Section IV will provide experimental results of AES encryption and decryption is compare with previously technique. Finally, section V will provide the conclusion of our proposed design.

## 2. LITERATURE REVIEW

AES was standardized by National Institute of Standards and Technology (NIST) in 2001 became Federal Information Processing Standard FIPS-197. Where Rijndael algorithm by Joan Daeman and Vicent Rijimen was selected as standard AES algorithm. The AES is private or symmetric block cipher which uses the same key for encryption and decryption is more suitable for faster implementation. The AES is a symmetric key for both encryption and decryption. AES cryptography algorithm is capable of encrypting and decrypting block size 128 bit data using cipher keys of 128, 196 or 256 bits (AES128, AES196 and AES256) [6].

The proposed design has ability to defend against Fault and glitch attacks with some large area than conventional design. Also proposed S-box is capable to reduce hardware resources and defend against glitch attacks [1].AES algorithm can resist any kinds of password attacks with a strong practicability in information security and reliability. So AES is widely adopted for various applications from high-end computers to low power portable devices. Numerous AES hardware architecture use in computer processor, SAN & Wi-Fi network, ATM, cellular phones and digital video recorders.

## 3. PROPOSED WORK

AES algorithm is symmetric block cipher that processes the state arraying from 128 bits data block using a key of 128, 192 or 256 bits length repeatedly. In encryption process round function consists of four different transformations- SubBytes, ShiftRows, MixColumns and AddRoundKey but last round function without MixColumns transformations. A ShiftRows transformations are truly dependent on state-wise operation of cyclic row shifting and MixColumns select 4 byte column operation done at simultaneously. Similarly inverse chipper decryption process round function consists of four different transformations- InvSubBytes, InvShiftRows, InvMixColumns and AddRoundKey, but last round function without InvMixColumns transformations.

Key Expansion or Schedule generation module is a common unit in AES encryption and decryption core. Key Expansion used to generate a series of Round Keys from the Cipher Key. A State is two-dimensional array of 4 x 4 size having 8 bytes consists of four rows contain block length divided by 32 is presented in hexadecimal format.

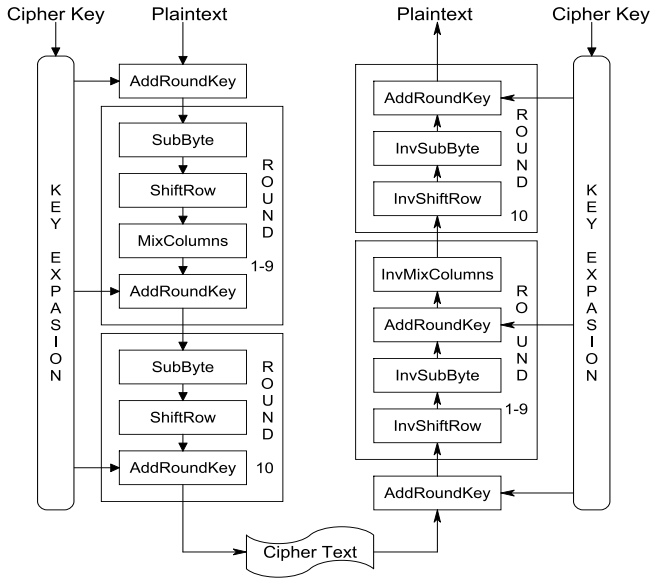


**Fig. 1:** AES standard encryption & decryption Algorithm

### 3.1 S-box Architecture

S-Box for SubByte and InvSubByte operation is implement using two method - Conventional BRAM implementation or combinational logic. Conventional BRAM has all pre-computed 256 values are stored in a ROM based lookup table and input byte wired to ROM's address bus. But BRAM method suffers from unbreakable delay as fixed access time for read and write operation and low latency due to ROM access time To increase throughput parallel ROMs are leading to large size of chip area requires high amount of memory. Therefore S-box transformation through composite field arithmetic is more suitable for low latency with reduction in area against. A more suitable second method to implementing S-Box is using combinational logic. It has advantage like small area occupancy and pipelined for increased performance in clock frequency. In this paper S-Box architecture based on combinational logic is present.

It is computed by multiplicative inverse in GF($2^8$) followed by an affine transformation. Where InvSubByte transformation compute using inverse affine transformation is applied first then multiplicative inverse as shown in figure 2.

The Affine Transformation (AT) and inverse Affine Transformation ($AT^{-1}$) are computed using following equation

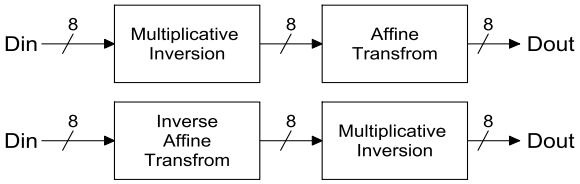


**Fig. 2:** SubByte and InvSubByte transformation in Sbox

I $$AT(a) = \begin{pmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} a_7\\ a_6\\ a_5\\ a_4\\ a_3\\ a_2\\ a_1\\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0\\1\\1\\0\\0\\0\\1\\1 \end{pmatrix}$$

II $$AT^{-1}(a) = \begin{pmatrix} 0&1&0&1&0&0&1&0\\ 0&0&1&0&1&0&0&1\\ 1&0&0&1&0&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&0&1&0&1\\ 1&0&0&1&0&0&1&0\\ 0&1&0&0&1&0&0&1\\ 1&0&1&0&0&1&0&0 \end{pmatrix} \times \begin{pmatrix} a_7\\ a_6\\ a_5\\ a_4\\ a_3\\ a_2\\ a_1\\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0\\0\\0\\0\\1\\0\\1 \end{pmatrix}$$

Both Sbox and InvSbox transformations have same multiplicative inversion module. GF multiplication is compute by decomposing complex GF($2^8$) to lower order fields as GF($2^1$) and GF($2^2$) also GF(($2^2$)$^2$). In figure 3 computation of multiplicative inverse in composite fields cannot be directly applied to GF($2^8$) multiplication.

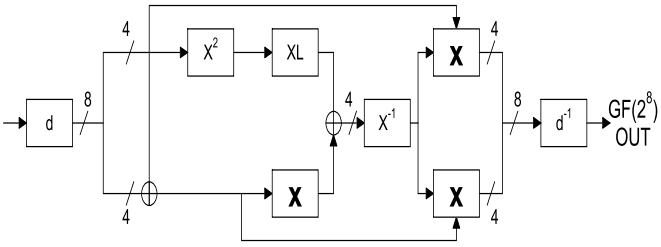


**Fig. 3:** Multiplicative inversion module for S-Box [4]

Addition of GF($2^4$) elements is compute using simply bitwise XOR operation. Multiplicative Inversion ($X^{-1}$) is inverse of individual bits compute from larger equation so that pre-computed value can be used [5]. The Isomorphic Mapping (d) and Inverse Isomorphic Mapping ($d^{-1}$) composite field are compute using following equation

$$\delta \times q = \begin{pmatrix} 1&0&1&0&0&0&0&0\\ 1&1&0&1&1&1&1&0\\ 1&0&1&0&1&1&0&0\\ 1&0&1&0&1&1&1&0\\ 1&1&0&0&0&1&1&0\\ 1&0&0&1&1&1&1&0\\ 0&1&0&1&0&0&1&0\\ 0&1&0&0&0&0&1&1 \end{pmatrix} \times \begin{pmatrix} q_7\\ q_6\\ q_5\\ q_4\\ q_3\\ q_2\\ q_1\\ q_0 \end{pmatrix} \quad \delta^{-1} \times q = \begin{pmatrix} 1&1&1&0&0&0&1&0\\ 0&1&0&0&0&1&0&0\\ 0&1&1&0&0&0&1&0\\ 0&1&1&1&0&1&1&0\\ 0&0&1&1&1&1&1&0\\ 1&0&0&1&1&1&1&0\\ 0&0&1&1&0&0&0&0\\ 0&1&1&1&0&1&0&1 \end{pmatrix} \times \begin{pmatrix} q_7\\ q_6\\ q_5\\ q_4\\ q_3\\ q_2\\ q_1\\ q_0 \end{pmatrix}$$

A GF($2^4$) Squaring operation compute using below equation

S2 = d3 XOR d2; S1 = d2 XOR d1;

S0 = d3 XOR d1 XOR d0; S3 = d3;

Multiplication with constant (λ) is generate by substitute irreducible polynomial as shown in below expression.

λ3 = S2 XOR S0; λ1 = S3; λ0 = S2;

λ2 = S3 XOR S2 XOR S1 XOR S0;

Multiplication of GF($2^4$) contain addition and decomposition multiplication operations in GF($2^2$) as shown in figure 4. Where

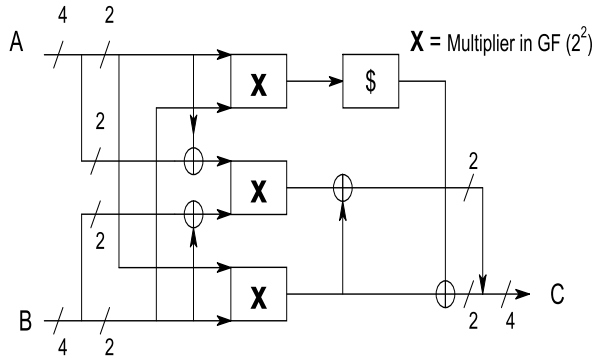


**Fig. 4:** Hardware implementation of GF($2^4$) multiplication

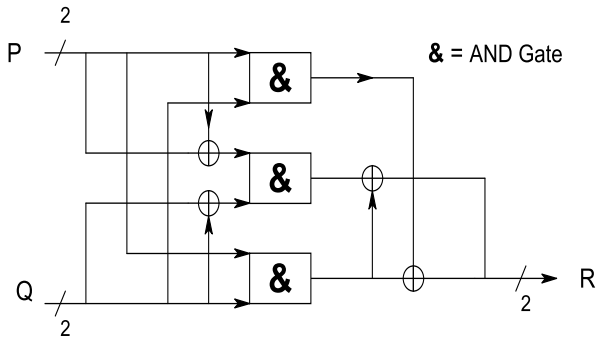


**Fig. 5:** Hardware implementation of GF($2^2$) multiplication

Multiplication with constant ($) is derived from equation

$1 = q1 XOR q0; $0 = q1;

A figure 5 describe Multiplication of GF($2^2$) is made only using through logical AND & XOR gate.

### 3.2 Key Expansion

Key Expansion routine to perform key scheduling which generate a series of Round Keys from the cipher key as shown figure 6. SubWord present in Key Expansion routine that takes a 4-byte input word gives 4-bytes output word using Sbox. The RotWord function performs a cyclic permutation on input word gives cyclic right shifted 4 bytes output word. Rcon is array of bytes in a word having fixed logical value having size of 128 bit [2]. A 128 bit Key register is fixed signal used to temporal storage of cipher key computed for each round of operation. Here key expansion module generate 10 number of 128 bit size Partial key for each round of operation.

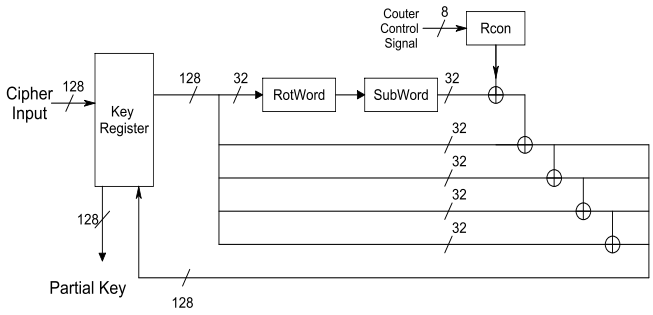


**Fig. 6:** Block diagram of Key Expansion

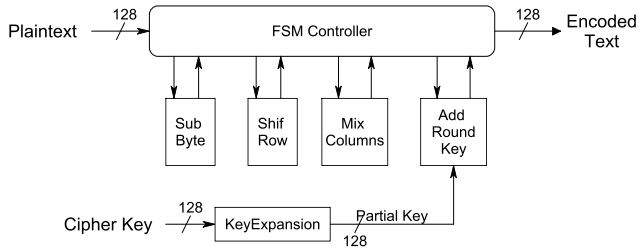


**Fig. 7:** Block diagram of AES Encryption

### 3.3 AES Encryption

AES Encryption has following subsequent steps are: SubByte, ShiftRows, MixColumns and AddRoundKey as shown figure 7. In SubBytes transformation is cipher undergo process of nonlinear byte substitution table (S-box) that operates on each of the State bytes independently. ShiftRows transformation is cipher that processes the State by cyclically right shifting of last three rows in State. MixColumns is transformation where cipher takes a columns of State and mixes their data independently gives one another to produce new columns using GF ($2^8$) polynomial. AddRoundKey is transformation cipher and Inverse Cipher is XOR operation with Round Key added to State.

A FSM controller is used for synchronization purpose where clock and reset are input of system. A block 128 bit plaintext input is XOR with partial key repeated for 10 times uses SubByte, ShiftRows, MixColumns, and AddRoundKey which generate encoded text. The main purpose of saving computation period of key expansion operation unnecessary repeated for same cipher key which enhance throughput of system by maintain standard frequency.

### 3.4 AES Decryption

It is inverted operation of encryption is implemented using reverse order Inverse Cipher in AES algorithm. AES decryption contain following subsequent steps are: InvShiftRows, InvSubBytes, InvMixColumns and AddRoundKey as shown figure 7. InvShiftRows is transformation is inverse of ShiftRows processes the State by cyclically left shifting of last three rows in State.

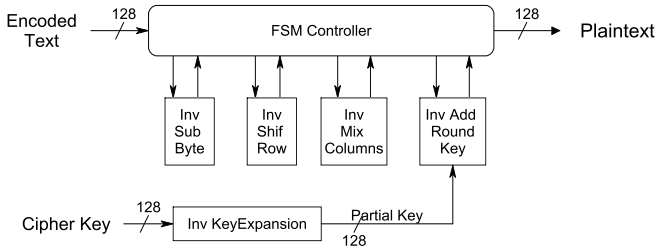


**Fig. 8:** Block diagram of AES Decryption

InvSubBytes is the inverse of byte substitution transformation. An inverse S-box obtained by applying the inverse affine transformation followed by taking multiplicative inverse in $GF(2^8)$ polynomial. InvMixColumns is the inverse of MixColumns transformation operates on column-wise operation four term polynomial and multiplied modulo $(X^4+1)$ with a fixed polynomial. InvAddRoundKey Transformation is own inverse AddRoundKey which involves 128 bitwise XOR operation. InvKeyExpansion module is 128 bit cipher generate 10 number of 128 bit size Partial key for each round of operation similar as KeyExpansion. But major difference is sequence of generated partial key is inverted.

### 3.5 AES Attack with Analysis

Fault injection attacks based on exploit computational errors to find cryptographic keys. Attacker requires use of a "fault model" to device require detailed knowledge of design of system and a mean to reliably induce faults without permanently damaging a unit. In AES-base smart card to induce setup time violations is presented which allowed predictable fault injection analysis to retrieve a full AES-128 key [6].

Glitch attack cause due to glitch which define as undesired transition is occurs before a signal settles to its intended value. A glitch is create a transient fault that difficult to troubleshoot in hardwired crypto core. A clock glitches & power supply glitches are helpful corruption data. Using proper cascade structure of flip flop within synchronized system glitch attack reduce dynamical reduced.

## 4. RESULTS AND COMPARISONS

The experimental results use the Xilinx Spartan 6 FPGA target device XC6SLX16-3-CSG324. A Xilinx ISE 14.7 tools used for synthesis & implementation of logic also XPower Analyzer for power estimation. A Xilinx Isim P.2013 and ModelSim SE 6.3c for testing & verification of simulation result. A proposed designs of AES encryptor and decryptor has area utilization as summarized in tables 1 & table 2 respectively. A table 3 describe Sbox design compared with previous Sbox designs. Our proposed design having very low dynamic power consumption with less number of 4 LUT's count. It's seen from table 4 that proposed design also achieves much higher throughput than previous AES encryption designs is summarized in detail.

**Table 1** Proposed work AES Encryption design utilization

| **Parameter** | **Encrypt core** | |
|---|---|---|
| Slice Registers | 554 | 3 % |
| Slice LUTs | 3531 | 38 % |
| LUT FF | 407 | 11 % |
| Block RAM | 8 | 25 % |
| Maximum Frequency | 277.369 MHz | - |

**Table 2** Proposed work AES Decryption design utilization

| **Parameter** | **Decrypt core** | |
|---|---|---|
| Slice Registers | 607 | 3 % |
| Slice LUTs | 3529 | 38 % |
| Logic gate | 426 | 11 % |
| Block RAM | 20 | 62 % |
| Maximum Frequency | 223.157 MHz | - |

**Table 3** Comparisons of Sbox Architecture

| **Parameter** | **Proposed Design** | **[4]** | **[5]** |
|---|---|---|---|
| Device | XC6SLX16 | XC2VP30 | XC2VP30 |
| # of Slice | 69 | 40 | 37 |
| # of 4 LUTs | 7 | 71 | 66 |
| Max. Delay (ns) | 15.45 | 15.00 | 15.6 |
| Total Dynamic Power (W) | 0.020 | 7.271 | 9.74 |

**Table 4** Comparisons of proposed Encryptor with existing design

| **Parameter** | **Proposed Design** | **[1]** | **[2]** | **[3]** |
|---|---|---|---|---|
| Platforms | Xilinx Sparton-6 XC6SLX16 | Xilinx Virtex-5 XC5VL50 | Xilinx Virtex-2 XC2VP20 | Altera APEX20K-C |
| Data path (bit) | 128 | 32 | 128 | 128 |
| Area | 554 slices/ 3531 LUT | 769 slices/ 2350 LUT | 9028 slices | 40960 slices/ 895 LUT |
| Frequency (MHz) | 277.369 | 100.8 | 220.7 | - |
| Throughput (Mbps) | 200 | 73.3 | 28250 | 1188 |

## 5. CONCLUSIONS

This paper gives a design of AES algorithm using pipeline structure and parallel processing. The proposed design has ability to defend against glitch attacks without very extra larger area compare to conventional design. It shows improvement in area and power consumption compared to conventional architecture. Whereas using pipeline structure

for repeated computation it become lower down speed as well as data rate. But its capable follows as per standard of AES.

## REFERENCES

[1] Z. Yuan, Y. Wang, J. Li, R. Li and W. Zhao, "FPGA based optimization for masked AES implementation", Proc. IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS), pp.1-4 2011.

[2] Issam Hammad, Kamal El-Sankary, Ezz El-Masry, "High-Speed AES Encryptor with Efficient Merging Techniques" Proc. IEEE Embedded Systems Letters, vol. 2, no. 3, Sept 2010.

[3] Hoang Trang, Nguyen Van Loi, "An efficient FPGA implementation of the Advanced Encryption Standard algorithm", Proc. Computing and Communication Technologies RIVF International Conference, pp. 1-4 2012.

[4] Saurabh Kumar, V.K. Sharma, K. K. Mahapatra, "Low Latency VLSI Architecture of S-box for AES Encryption", Proc. International Conference on Circuits, Power and Computing Technologies, pp. 694-698 2013.

[5] Saurabh Kumar, V.K. Sharma, K. K. Mahapatra, "An Improved VLSI Architecture of S-box for AES Encryption", International Conference on Communication Systems & Network Technologies, pp. 2013.

[6] FIPS 197, Advanced Encryption Standard http://csrc.nist.gov/publications/fips/ fips197/fips-197.pdf