Design and Implementation of AES algorithm with Fault Detection and Correction in FPGA

Virupakshgoud Bistanagouda
PG student , M.Tech(VLSI & ES),Dept. of ECE.
AMCEC, Bangalore
Email: goudavirupaksh@gmail.com

Prof. Vinay kumar T N
Assistant Professor, Dept. of ECE.
AMCEC, Bangalore
Email: vinaykumartn29@gmail.com

Abstract : The proposed design will work in error detection and correction using crypto system. The Advanced Encryption Standard (AES) based on the Rijndael algorithm is an efficient cryptographic technique that includes generation of ciphers for encryption and inverse ciphers for decryption. Higher security and speed of encryption /decryption is ensured by operations like Sub Bytes(S-box) and Inverse Sub Byte operation, Mix Column and Inverse Mix Column operations are designed as Look Up Tables (LUTs). In encryption side error detection and correction will be done. Its gives more security mainly in wireless communication system, Hardware implementation of cryptographic algorithms are physically secure than software implementations since outside attackers cannot modify them. In order to achieve higher performance in today's heavily loaded communication networks hardware implementation is a wise choice in terms of better speed and security. The AES algorithm is designed in Verilog language and hardware implementation on Xilinx–Spartan3 Field Programmable Gate Array (FPGA).

Keywords

AES, Rijndael; Cryptography; FPGA; Verilog; Encryption; Decryption.

1. INTRODUCTION

The transfers of data in the present days invariably necessitate the use of encryption. Besides its uses in Military and Government's secret communication, Encryption is also used for protecting many kinds of civilian systems such as Internet e-commerce, Mobile networks, automatic teller machine transactions, copy protection (especially protection against reverse Engineering and Software piracy), and many more. Data encryption is achieved by following a systematic algorithm called encryption algorithm. An encryption algorithm provides Confidentiality, Authentication, Integrity and Non-repudiation. Confidentiality is the requirement that the information is kept secret from people who are not authorized to access it. Authentication is the certainty that the message indeed originates from the supposed sender. Integrity is the requirement that the information is unaltered and complete, or, that information "is modified only by those users who have the right to do so." Non-repudiation means that the sender or receiver of a message cannot deny having sent or received the message

.

Encryption is usually done just before sending data. To utilize the channel resources completely encryption algorithm must have a speed at least equivalent to data transmission speed. Achieving high throughput for encryption algorithm for a communication channel of high data rate is a challenging task. The hardware (FPGAs and Application Specific Integrated Circuits-ASICs) implementation of such algorithm which meets these requirements is done in the present work. FPGAs are chosen considering several advantages over the other counterpart.

The AES was published by National Institute of Standards and Technology (NIST) in 2001. Later Rijndael algorithm was selected as AES algorithm. Rijndael algorithm can have key length of 128, 192 and 256 bits while block size must be 128 bit.

There are many architecture proposals for AES Rijndael algorithm but many of them are poor in terms of security and speed. This paper proposes a different approach to increase speed by utilizing lesser resources available in FPGA and error detection and correction is done for secure data transmission.

2.ADVANCED ENCRYPTION STANDARD (AES)

The AES is a cryptographic algorithm that is used to encrypt (encipher), and decrypt, (decipher), information. Key Expansion generates a Key Schedulethat is used in Cipher and Inverse Cipher procedures. Cipher and Inverse Cipher are composed of specific number of rounds (Table 1). For the AES algorithm, the number of rounds to be performedduring the execution of the algorithm is dependent on the key length.

Table 1:Comparison of block size, key length and number of rounds of AES keys.

Type	Block Size Nbwords	Key Length Nkwords	Number of Rounds Nr
AES -128 bits key	4	4	10
AES -192 bits key	4	6	12
AES -256 bits key	4	8	14

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte oriented transformations: SubBytes, ShiftRows, MixColumns and Add Round Key. Fig. 1 shows the basic schematic of the structure of AES**.**

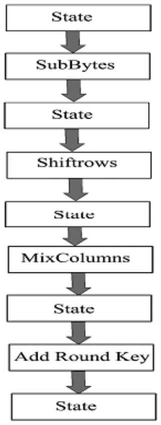


Fig.1:Basic structure of AES

2.1 AES OPERATION

Fig2 indicates the transformation in AES algorithm based on the structure in Fig 1.The brief intrudction listed below:

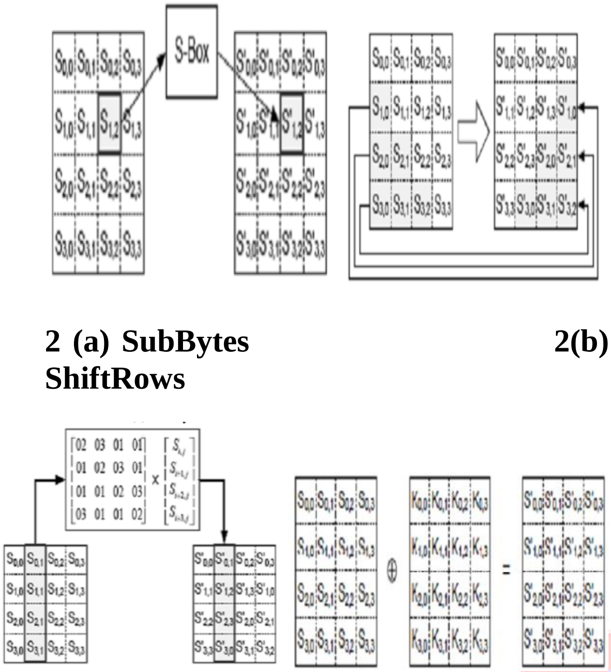
1)Sub Bytes: The sub byte operation is a non- linear byte substitution that operates on each byte of the State using a substitution table.

2) ShiftRows: In the ShiftRows operation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes.

3) MixColumns: Mixing operation which operates on the columns of the State using a linear transformation.

4) Add Round Key: A Round Key is added to the State by a simple bitwise XOR operation.

Key: A Round Key is added to the State by a simple bitwise XOR operation.



2 (a) SubBytes 2(b) ShiftRows

2(c)Mix colums 2(d) AddRoundKey

Fig.2: Transformations in AES Algorithm

2.2 Inputs and Outputs

The input and output for the AES algorithm consistsof sequences of 128 bits. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits. The basic unit for processing in the AES algorithm is a byte (a sequence of eight bits), so the input bit sequence is first transformed into byte sequence. In the next step a two-dimenstion array of bytes(called the state) is built. The State array consists of four rows of bytes, each containing Nb bytes, where Nb is the block size divided by 32 (number of words). All internal operations (Cipher and Inverse Cipher) of the AES algorithms are then performed on the State array, after which its final value is copied to the output (State array is transformed back to the bit sequence).

2.3 Cipher

Using round function, which is composed of four different byte-oriented transformations, the Cipher converts input data (the input data is first copiedto the State array) to an unintelligible from called cipertext. After an initial Round Key addition, the State array is transformed by implementing a round function with the final round differing slightly from the firs Nr-1(Table1) rounds. The round function is parameterized using a key schedule that consists of a onedimensional array of four-byte words (Round Key) derived using the Key Expansion routine. All Nr rounds (see Table 1) are identical with the exception of the final round, which does not include the MixColumns transformation.

2.4 Key Schedule

Key scheduling is a critical process in AES that generates (Nr+1) round keys based on a external single key. The Key expansion process of AES algorithm uses a Cipher Key K to generate a key schedule. This generates Nb(Nr+1) words, of which the algorithm requires initial Nb words and each of the Nr rounds require Nb words of Key Data. Thisresults in a key schedule of a linear array of 4-bytes words denoted by [wi], where $0 \leq i< Nb(Nr + 1)$. The SubWord( ) function accepts a 4-byte input word and on each of the four bytes S-box transformation is applied. The RotWord( ) performs cyclic permutations on the input word[a0,a1,a2,a3] and returns [a1,a2,a3,a0].The round constant word array or Rcon[i] holds [x i-1,{00},{00},{00}], where in x^ i-1 being powers of x (x is denoted as {02}) in the field GF(2^8). The key expansion routine for 128-bits using 10 rounds has been shown in figure 3.

The key setup phase and then read them from this memory whenever required by the encryption/decryption unit. The critical path of Key Expansion is shorter than that of any round, speed of the system can't be enhanced by reducing the critical path of Key Exapansion. Genarating round keys on the fly eliminates the requirement for key storage, but brings overhead for decryption since decryption begins after the last roundkey is generated.

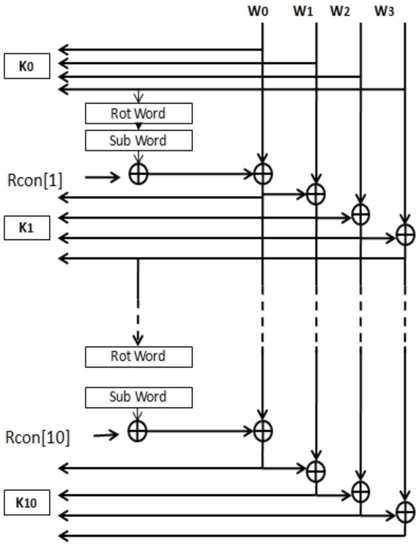


Fig.3:AES 128 bit key expansion operations

3. BLOCK DIAGRAM

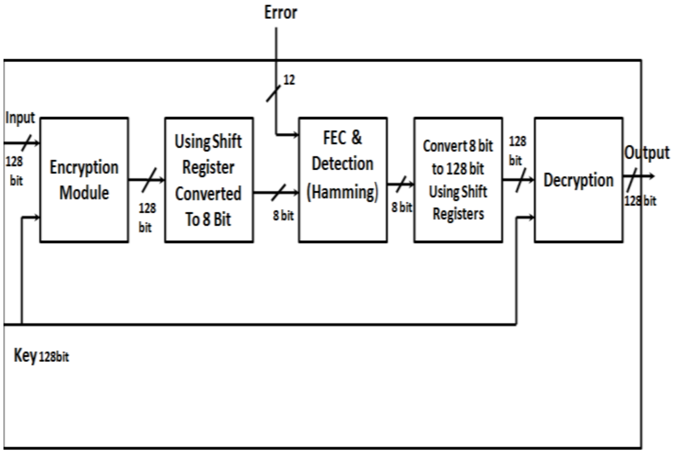

Error
12
Input
128 bit
Encryption Module
128 bit
Using Shift Register Converted To 8 Bit
8 bit
FEC & Detection (Hamming)
8 bit
Convert 8 bit to 128 bit Using Shift Registers
128 bit
Decryption
Output
128 bit
Key 128bit

Fig.4: Error Detction and Correction using crypto-system block diagram

The above block diagram explains all about encryption and decryption. The 128 bit input is given to the encryption module where 128 bit cipher text is produced. This cipher text is converted to the 8 bit data. The 8bit shifted data along with the error is made to pass through forward error correction and detection designed by hamming code, where the introduced error is detected and it is corrected. This corrected data is passed through shift register which converts to 128 bits. The final cipher text is decrypted using appropriate key.

4. SIMULATION RESULTS

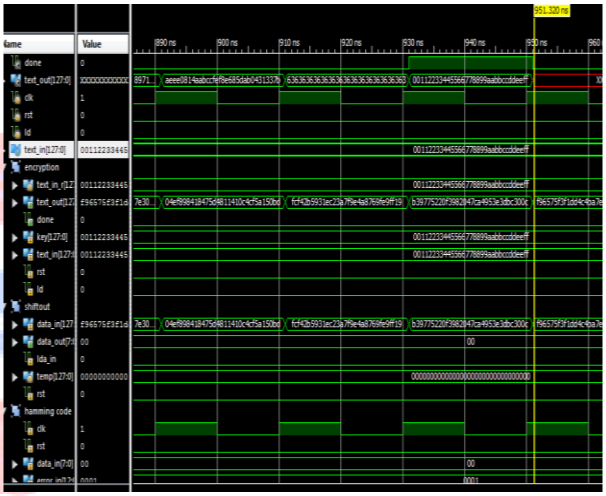

951.320 ns
Name
Value
890 ns
900 ns
910 ns
920 ns
930 ns
940 ns
950 ns
960
done 0
text_out[127:0] XXXXXXXXXXXX
8971...
aeee0814aabccfef8e685dab0431337b
63636363636363636363636363636363
00112233445566778899aabbccddeeff
clk 1
rst 0
ld 0
text_in[127:0] 00112233445
00112233445566778899aabbccddeeff
encryption
text_in_r[127 00112233445
00112233445566778899aabbccddeeff
text_out[127 f96575f3f1d
7e30...
04ef898418475d4811410c4cf5a150bd
fcf42b5931ec23a7f9e4a8769fe9ff19
b39775220f3982047ca4953e3dbc300c
f96575f3f1dd4c4ba7e
done 0
key[127:0] 00112233445
00112233445566778899aabbccddeeff
text_in[127:0] 00112233445
00112233445566778899aabbccddeeff
rst 0
ld 0
shiftout
data_in[127 f96575f3f1d
7e30...
04ef898418475d4811410c4cf5a150bd
fcf42b5931ec23a7f9e4a8769fe9ff19
b39775220f3982047ca4953e3dbc300c
f96575f3f1dd4c4ba7e
data_out[7:0] 00
00
lda_in 0
temp[127:0] 00000000000
00000000000000000000000000000000
rst 0
hamming code
clk 1
rst 0
data_in[7:0] 00
00
error_in[12:0] 0001
0001

Fig.5: Decryption Simulation Result

Input:
textin:128'H00112233445566778899AABBCCDDEEFF
Input
Key:128'H00112233445566778899AABBCCDDEEFF
Output text out:
128'H00112233445566778899AABBCCDDEEFF

5. CONCLUSION

The encryption standard alone is not so efficient to protect the information (data). Because of which we are facing lot of issues regarding the security of the data, the fault detection and correction computation overhead. By using the advanced encryption standard algorithm the problem can be entirely eliminated and the fault detection and correction can be achieved. The reliability and the integrity of the data can be ensured with high accuracy. The proposed fault detection and correction AES model targets wireless communication application like satellite application domain, internet e-commerce, mobile networks etc. The AES algorithm with fault detection and correction is also implemented in spartan3 FPGA.

.

6. REFERENCES

[1]Abhijith.P S ,mallika srivastava,aparnamishra, “high performance hardware implementation of aes using minimal resources” iee int. conf. indian institute of information technology, allahabad, india 2013.International Journal Advances in Engineering & Technology, May 2012.©IJAET ISSN: 2231-1963

[2] chih-peng fan* and jun-kui hwang “fpga implementations of high throughput sequential and fully pipelined aes algorithm” international juornal of electrical enginnering vol.15 n0.6 pp.445-447(2008)

[3] rourab paul, sangeet saha, suman sau,“ design and implementation of real time aes-128 on real time Operating System for Multiple FPGA Communication” Dept. of Electronic Science1 Kolkata, India

[4] Shylashree.N1, Nagarjun Bhat2 and V. Shridhar3 “FPGA Implementations of Advanced Encryption Standard: A Survey”

[5] Federal Information Processing Standards Publication 197 November 26, 2001 Announcing the Advanced Encryption Standard (AES)

[6] Samir El Adib and Naoufal Raissouni, “AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparison of 128, 192 and 256-bits Key” International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 1, No. 2, July 2012, pp. 67~74