An Implementation of AES Algorithm in FPGA

Isaac Nattan da Silva Palmeira, Alcir Cledson de S. Góis, Wanderson Roger Azevedo Dias, Edward David Moreno
Department of Computing - DComp
Federal University of Sergipe - UFS
Aracaju, Sergipe, Brazil
{isaacnattan2, wradias, edwdavid}@gmail.com, alcirgois@yahoo.com

ABSTRACT
This article aims to present an alternative implementation of the Rijndael algorithm, the AES (Advanced Encription Standart). The algorithm described above is able to encrypt pieces of 16-byte text using a key of the same size. The basic operations of the AES operation will be described: AddRoundKey, SubBytes, ShiftRows, MixColumns, and their respective inverses still a key generator algorithm (KeyExpansion).

Keywords
cryptography; AES algorithm; encryption; decryption.

1. INTRODUCTION
Encryption originates from the greek word which means cryptos secret or hidden. The objective of cryptography are encryption methods or data messages so that only the legitimate receiver of certain information may have access to it. In the treatment of problems related to information security is added to it the existence of cryptanalysis, responsible for studying the means of deciphering an encrypted message without all the information needed to decode the message correctly.

Encryption electronics began to be used after the Second World War. But not until 1974 that the first cryptographic algorithm has been used in a commercial manner. Lucifer was developed by IBM and, after several changes made by the NSA (National Security Agency), came to be called DES (Data Encryption Standart) then being used as a U.S. cryptographic standard.

For 20 years, DES was the default algorithm used by the Yankee government to protect confidential information. The emergence of the AES (Advanced Encryption Standard) was due to the great need to replace DES, which has become outdated because of the small key size (56 bits) used. For this, the NIST (National Institute of Standards and Technology) has launched a competition in 1997 to adopt the new symmetric algorithm, which would be called AES.

The algorithm should meet some requirements such as: copyright free, public disclosure, faster compared to DES, block cipher with 128-bit keys of 128, 192 and 256 bits, possibility of implementation in software and hardware.

The algorithm was created by Belgian Vincent Rijmen and Joan Daemen. As both the first in the second stage of the competition all the algorithms described meeting requirements of the tender, the decision was made based on other qualities such as security, flexibility, good performance in software and hardware etc.

2. ALGORITHMIC STRUCTURE OF AES
Some concepts are of vital importance for the understanding of the proposed algorithm, one is understanding what comes to the state. The state is an array of bytes that will be handled during the various rounds or rounds, and therefore will change every step.

The size of the array will depend on the block size used, consisting of four rows and columns Nb, where Nb is the number of bits divided by the block 32. The proposed algorithm has the ability to encrypt text of 128 bits (16 bytes) , thus the state will have 4x4 size. The main key is grouped in the same way that the state with Nk columns.

As the number of rounds AES or number of rounds varies depending on the key length, and Nr (number of rounds) of 10, 12 and 14 for Nk (number of columns) equal to 4, 6 and 8 respectively. In each round of the encryption algorithm, five steps are performed: AddRoundKey, SubBytes, ShiftRows, MixColumns and KeyExpansion algorithm. In the last round, however, the MixColumns operation is suppressed.

2.1 Mathematical Considerations
Various operations are defined in the AES bytes, with one byte represented in the finite field GF (2^8), which mathematically defines a Galois field, also known as Galois Field (GF).

1.1.1. GF(2^8)
The GF (2^8) symbolizes a field, algebraic structure which has two operations {+, ·} addition and multiplication respectively, closed on elements of the field.

Definition 1.1: a field is a set F with two laws of composition C and such that.

- (F +) / C is a commutative group;
- (F ·) /, where F \ {0}, is a commutative group;
- the distributive law holds.

A field contains at least two distinct elements '0 'and '1'.

2.1.2. Addition in GF(2^8)
The addition of polynomials in GF (2^8), corresponds to the XOR (exclusive OR) bitwise.

2.1.3. Multiplication in GF(2^8)
In the polynomial representation, this procedure corresponds to polynomial multiplication already known, ie, applying the distributive property, whereas the degree of the resulting polynomial can never be greater than or equal to eight, hence the name GF (2^8). For the degree of the polynomial higher than or equal to eight ($8 \geq g$) is added to the irreducible polynomial $m(x)= x^g + x^{g-4} + x^{g-5} + x^{g-7} + x^{g-8}$.

2.2 AES Transformations
Possession of some mathematical considerations and a brief introduction of algorithmic structure of the AES, the basic functions that operate behind the encryption/decryption of a block process will be presented.

2.2.1. AddRoundKey

The AddRoundKey operation is nothing more than a bitwise xor operation between the state and the key round. This is the one that has no reverse, in the deciphering process is used with the following keys generated backwards. The Figure 1 shows schematically how this operation works.

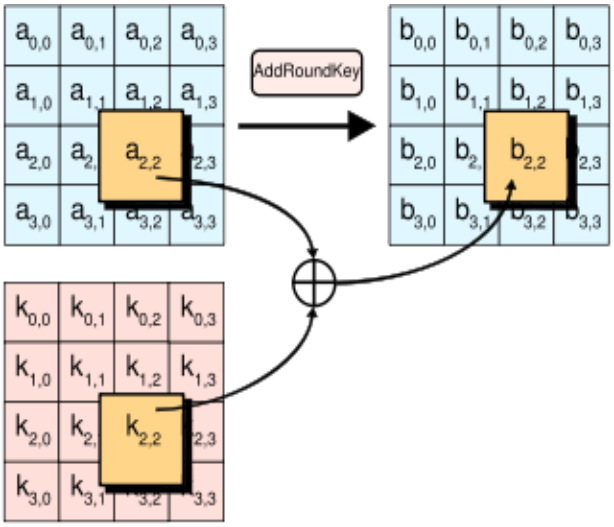

Figure 1. Operation schematic AddRoundKey.

2.2.2. SubBytes and its Inverse

SubBytes transformation modifies the values of the state based on a substitution box (S-Box), using the bytes from the current state as indices to the values contained in the S-Box used. The S-Box is an array of size 16x16 with different hex values. The replacement is performed as follows: the first and second current number hexadecimal value respectively represent the row and column of the value contained in the S-Box. In this case the inverse transform process applies, however considering the inverse S-box.

2.2.3. ShiftRows and its Inverse

This transformation consists in a rotation left the state lines so: (i) the first line is not amended; (ii) the second one suffers a rotation line; (iii) the third line suffers two rotations; (iv) the fourth line suffers three rotations. The Figure 2 shows how this process works.

2.2.4. MixColumns and its Inverse

MixColumns transformation is performed in a matrix multiplication in GF (28). The state is being modified multiplied by a constant matrix formed by variations of the polynomial a (x) = {02}+{03}+{01}+{01}, the matrix formed by the variations of the polynomial is shown the Figure 3.

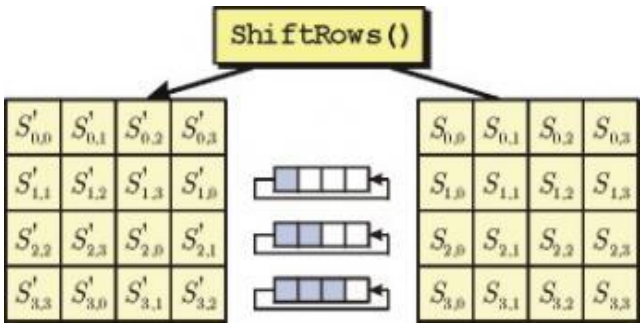

Figure 2. Operation schematic ShiftRows.

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Figure 3. MixColumns constant Matrix.

In its reverse process happens analogously differing only in the constant matrix used (see Figure 4).

e	b	d	9
9	e	b	d
d	9	e	b
b	d	9	e

Figure 4. InvMixColumns constant Matrix.

2. KEYEXPANSION ALGORITHM

For each round in the implementation of the algorithm is applied to a key, specifically the implementation of the transformation AddRoundKey. These keys are generated by KeyExpansion algorithm.

The key generation operation is to be applied to each word as they are known the columns of the state, one of the approaches to two distinct situations:

If the first byte of the word is not operated multiple of Nk, the new word will receive the result of the xor operation between the word and the word immediately preceding the previous key corresponding position.

If the identifier word op is a multiple of Nk, one xor should be applied between the replacement using the S-Box, the word immediately preceding rotated to the left, and the constant round of RC [j]. Then a new xor between the result of the previous operation and the word of the previous key corresponding position will be applied. The following table shows all the constants used in the rounds of the algorithm (see Table 1).

Table 1. Constants of Rounds

RC[1]	01 (0000 0001)
RC[2]	02 (0000 0010)
RC[3]	04 (0000 0100)
RC[4]	08 (0000 1000)
RC[5]	10 (0001 0000)
RC[6]	20 (0010 0000)
RC[7]	40 (0100 0000)
RC[8]	80 (1000 0000)
RC[9]	1b (0001 1011)
RC[10]	36 (0011 0110)

4. COMPLETE PROCEDURE OF THE AES ALGORITHM

Transformations in the encryption process can be followed in Figure 5. Deciphers In the sequence is exactly the reverse process to that shown.

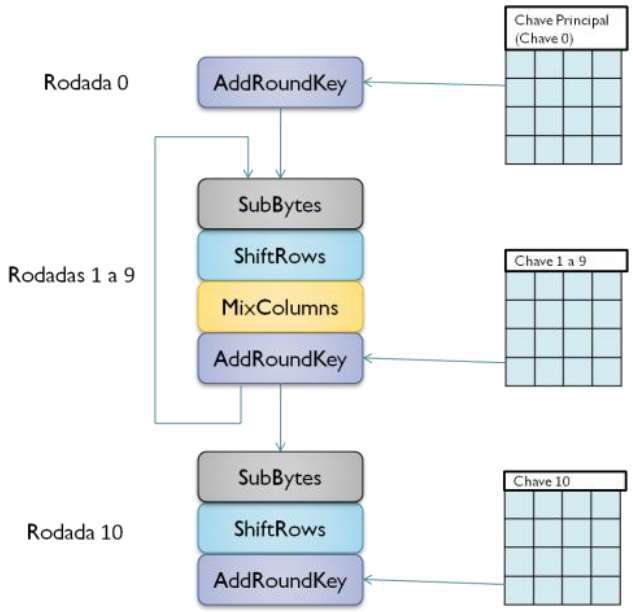

Figure 5. Complete procedure of cryptography.

The AES algorithm has been implemented with a view to saving space. Since the goal was to develop a method encrypts and decrypts, we studied the possibility of merging the transformations with their inverse and found, from this, the union between these factors. SubBytes operation for this factor was not found, but this was united with its inverse in order to reduce complexity.

ShiftRows transformation was realized that the same rotations that its inverse could be applied in reverse order. Thus the rotation applied to the third row would be the same for both cases.

The uniting factor found between the MixColumns operation and its inverse is the generalization of the constant matrix (see Figure 6).

Where the variable d assume '0 'and '1 in encryption process' in the decrypts. Once the changes merged, the complete algorithm should meet two methods of transformation of the state. Viewing this way the key generator would have to be completely executed before this, since in the latter decrypts key generated would be first processed.

*dd*10	*d*011	*dd*01	*d*001
*d*001	*dd*10	*d*011	*dd*01
*dd*01	*d*001	*dd*10	*d*011
*d*011	*dd*01	*d*001	*dd*10

Figure 6. Generalization of constant matrixes.

The transformation of state always start with the AddRoundKey operation. This would follow in the next transformations, based on the method and the round in question.

5. SOME RESULTS

All functions were simulated and analyzed in isolation and together, with the aid of the tool Amendment Quartus II version 9.0 and targeted for implementation in FPGA (DE-2 Amendment) capable of supporting up to 33.216 logic elements. The results of Table 2 can be obtained.

Having the data, the algorithm KeyExpansion was the most costly, but the complexity of the algorithm analyzing your time will be considered constant compared with the transformation of the state.

Table 2. Time and Space Statistics

	Time (ns)	Space the FPGA (logic elements)
AddRoundKey	12.124	128 <1% of the capacity maximum
SubBytes	19.278	6720 20% of the capacity maximum
ShiftRows	12.822	64 <1% of the capacity maximum
MixColumns	17.137	472 1% of the capacity aximum
KeyExpansion	90.828	8904 27% of the capacity maximum
COMPLETE	771.12	21421 64% of the capacity maximum

Thus to evaluate the method of encryption/decryption simply collect the transformation of worse weather (SubBytes), this time is the minimum time clock, and multiply it by the number of times that each transformation is applied. Thus we have:

$$Minimum\ Time = 19.278\ ns \cdot 40 = 771.12\ ns$$

With this result it is possible to estimate the throughput as:

$$Transfer\ Fee = 128\ bits/s\ 771.12 \approx 166\ Mbits/s$$

In the spatial aspect, the complete algorithm using totaled 21.421 logic elements, about 64% of the maximum capacity of the FPGA, among them we can highlight:

- Use of 1.540 bit registers (11 keys, current state and round in question);
- 4.186 bits of ROM (S-Box and Inverse, constant round and constant matrices).

The Figure 7 shows the state machine of the AES.

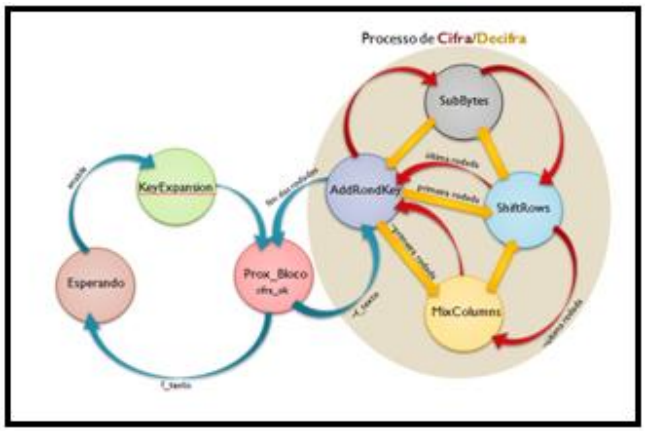

Figure 7. Complete procedure of cryptography.

6. CONCLUSIONS

Compared with other implementations, which separated their algorithms in cipher and decipher, one can notice the duplication of all transformations, mainly MixColumns which is of considerable size, so there's no doubt about space reduction of the proposed algorithm. With the separation of the methods, times cipher were evaluated separately from the times of decrypts, weighing more in the second, more precisely the inverse of the MixColumns operation, so the implementation

proposal should be compared to the decryption, since the union-inverse transformation does not compromise much run time.

This implementation also enables the use of pipeline - proposed future implementation. Could be a sharing of transformations for up to three blocks of the state. Thus, the throughput obtained for one block would be tripled, thereby increasing its speed of execution.

7. REFERENCES

[1] Daemen, J., and Rijmen, V. "AES submission document on Rijndael - version 2", September 1999.

[2] FIPS PUB 197, "Advanced Encryption Standard (AES)", National Institute of Standards and technology, U.S. Department of Commerce, November 2001. Link in: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[3] Daemen, J., and Rijmen, V. "The Rijndael Block Cipher, AES Proposal: Rijndael". Link in: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf.

[4] Mathias, L. A. P. "Algoritmo AES - Relatório técnico da disciplina Redes I", UFRJ, Julho 2011. Link in: http://www.gta.ufrj.br/grad/05_2/aes/.

[5] Ordoñez, E. D. M., Pereira, F. D., Penteado, C. G., and Pericine, R. de A., "Projeto, Desempenho e Aplicação de Sistemas Digitais em Circuitos Programáveis (FPGAs)". Marília – SP: Bless, 2004.

[6] Milene, J. S., "Fields and Galois Theory - Version 4.22", March 30, 2011. Link in: http://www.jmilne.org/math/CourseNotes/FT.pdf.