Journal of Theoretical and Applied Information Technology
20th July 2013. Vol. 53 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPLEMENTATION OF STRONGER AES BY USING DYNAMIC S-BOX DEPENDENT OF MASTER KEY

[1]SLIMAN ARRAG, [2]ABDELLATIF HAMDOUN, [3]ABDERRAHIM TRAGHA, AND [4]SALAH EDDINE KHAMLICH

[1,2 and 4] Laboratory of treatment of information Universite Hassan II Mohammedia, Casablanca, Morocco

[3] Laboratory of Information Technology and Modeling, Universite Hassan II Mohammedia, Casablanca, Morocco

E-mail: [1]arragsliman@yahoo.fr, [2]alhamdoun@yahoo.fr, [3]a.tragha@univh2m.ac.ma, [4]khamlich.salah@gmail.com

ABSTRACT

In this paper we propose and we show a new approach of nonlinear transformation algorithm for AES SBox to enhance the complexity of the SBox structure,we making AES stronger by using Dynamic Sbox,with look up table Sbox and Key expansion as modified when we change the initial key, that effectively providing a high resistance against differential cryptanalysis and especially the linear cryptanalysis. The structure of the AES S-box has been expanded and modified to be accordance with the proposed algorithm and to obtain good nonlinearity of the Sbox. This has been done without changing the basic operations of AES.The proposed modifications of the Advanced Encryption and a modified Sbox are implemented in Cyclone II Dvice by using VHDL language.

Keywords: *AES, Dynamic S-box, Master key, FPGA, Vhdl.*

1. INTRODUCTION

The evolution of information technology and in particular the increase in the speed of processing devices has necessitated the need to reconsider the cryptographic algorithms used. The National Institute of Standards and Technology of the United States (NIST) in cooperation with industry and cryptographic communities [1] have worked together to create a new cryptographic standard. The primary objective was a federal standard (Federal Information Processing Standard FIPS) to establish that specifies a cryptographic algorithm with the ability to improved protect sensitive information from the government. It was expected that the algorithm can be used both in the governmental structures of the United States and in the analysis of companies and private full sectors.After (mathematics, cryptography, statistics, engineering, etc.) of the algorithms, NIST [8] announced that the new standard uses the Rijndael algorithm and since 2001, it is the foundation of the new encryption standard AES.

In this paper we presente the design and hardware implementation of the AES (Advanced Encryption Security) in Cyclonne II, as using Xilinx ISE synthesis tool and Qaurtus II v9 for great simulated and synthesized. The architectural design has been described code in VHDL language. The AES has for the moment not been broken but the cryptanalysis of Rijndael (AES) has not stopped, for that we used a new approach allows us to obtain competitive performance. This approach will be studied its effect on the AES focusing on the S-Boxes wich look up table Sbox and Key expansion as modified when we change the initial key. Literatures [4], [10], [11], [12], [13], [18], [20], [21], [22], [25] describe design and implementation of AES and AES modified treatment in the FPGA.

2. AES RIJNDAEL ALGORITHM

The AES algorithm (or rijndael) [2], [3], [5] takes as input a block of 128 bits (16 bytes), the key is 128, 192 or 256 bits. The 16 bytes are swapped input according to a predefined table. These bytes are then placed in a matrix of 4x4 elements and lines are rotated to the right. The increment for the rotation varies with the number of the line. A linear transformation is then applied to the matrix, it consists of the binary multiplication of each matrix element with polynomials from a auxiliary matrix, this multiplication is subjected to

Journal of Theoretical and Applied Information Technology
20th July 2013. Vol. 53 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

special rules according GF (28) (or Galois field finite field). Linear transformation ensures better diffusion (spread of bits in the structure) over several turns.Finally, an XOR between the matrix and other matrix allows to obtain an intermediate matrix. These operations are repeated several times and define a "turn." For a 128, 192 or 256, AES requires respectively 10, 12 or 14 rounds.

The following diagram describes briefly the progress of encryption Fig 1:

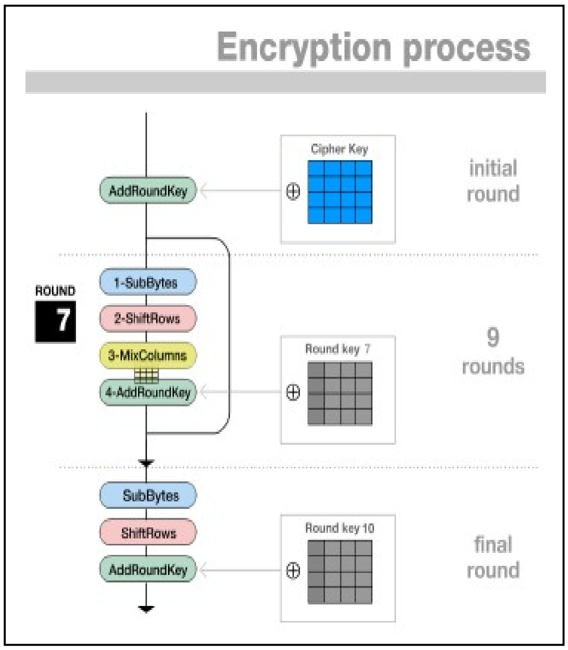

Figure. 1: Structure Of The AES Algorithm.

Three main criteria have been met in its design:
-Resistance against all known attacks.
-Speed code on the widest variety of platforms possible.
-Simplicity in design.
Rijndael (1998) was strongly influenced by his predecessor, Square algorithm (1997).Crypton and Twofish algorithms also use transactions Square. Rijndael is pronounced "Raindal".

• BYTE_SUB (Byte Substitution) is a nonlinear function operating independently on each block from a table called substitution.

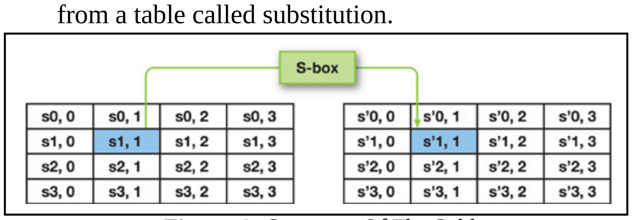

Figure 2: Structure Of The Subbyte

Table 1: Look Up Table Of S-BOX.

63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

• SHIFT_ROW is a function operating lags (typically it takes into 4 pieces of 4 bytes and operates left shifts of 0, 1, 2 and 3 bytes for the tracks 1, 2, 3 and 4 respectively).

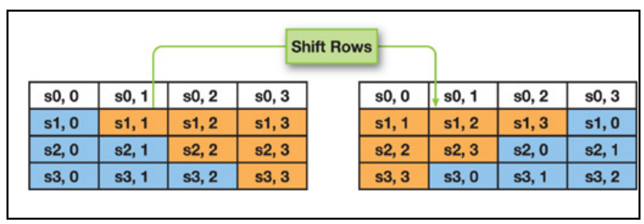

Figure 3: Structure Of The Shiftrows

• MIX_COL is a function which converts each input byte into a linear combination of input bytes and can be mathematically expressed by a product marticiel on the Galois field (28) [15].

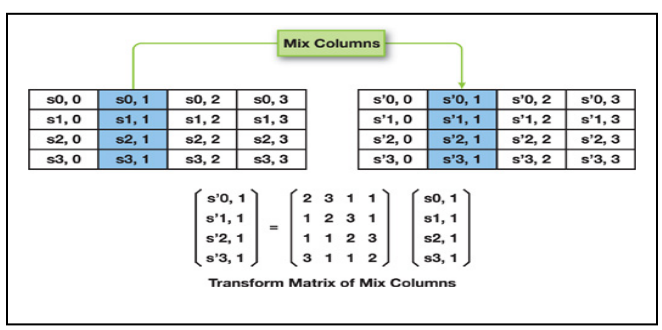

Figure 4: Structure Of The Mixcolumn

Journal of Theoretical and Applied Information Technology
20th July 2013. Vol. 53 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

• The Add RoundKey operation is a simple bit by bit XOR operation between the data and the roundkey (by using initial key or Key expansion).
The circled ⊕ means exclusive OR operation (XOR).
K_i is the ith subkey calculated by an algorithm from the master key K.

• Key Expansion operation: The algorithm for generating the 10 turns of the round key is as follows: The fourth column of the i-1 key is rotated such that each element is moved one row.

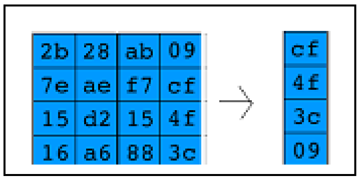

Figure 5:Rotated The Last Row

It then places the result by way of an algorithm forwards Sub-Box that replaces all eight bits of the matrix with a corresponding 8-bit value of S-Box. (See figure 7 below for SubByte inverse).

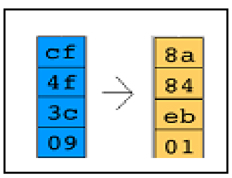

Figure 6:Subbyte The Row By Using Sbox Look Up Table

To generate the first column of the i-th key, this result is XOR-ed with the first column of the first i-key and a constant (row constant or Rcon) which depends on i.

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Figure 7:Look Up Table Of The Rcon

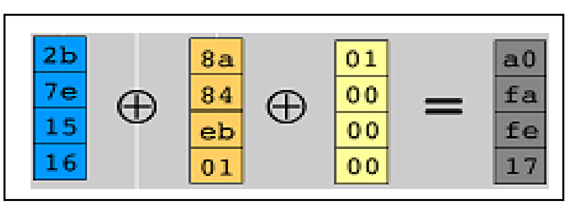

Figure 8:Operation Xor Between First Row Of Key And Last Row After The Modefication And First Row Of Rcon

The second column is generated by XOR-ing the first column of the i-th key in the second column of the first key-i.

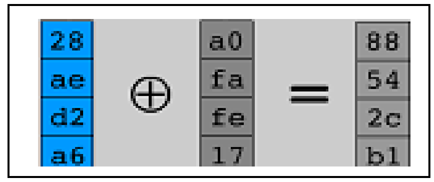

Figure 9:XOR-Ing Between Modified Row And Second Row Of Master Key

This iteration continues for the other two columns to generate the key to ith.

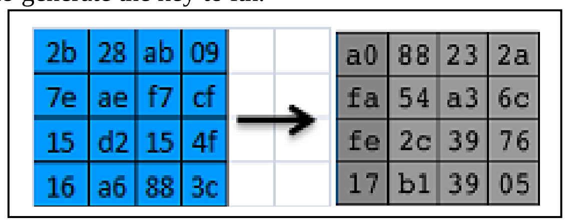

Figure 10: Example Key Extension

In addition, this process continues iteratively to generate all 10 keys As a final note, all these keys are stored in a static manner once they have been calculated first as th key generated is necessary for the tower (10-i) th decryption.

• Decryption is the process of applying the reverse operations in the reverse order and with subkeys also in reverse order.

2.1 Attacks

The AES has for the moment not been broken and the exhaustive search ("brute force") is the only solution. Rijndael was designed so as to make conventional methods such as linear and differential analysis very difficult[17], [19].

3. DEFINING THE PROBLEM

One of the main functions used in AES is the "Substitution of bytes (SubBytes)". This function performs a non-linear substitution, which is performed independently on each input byte. The matrix which gives the relationship between the input and output bytes, called (S-BOX) in the AES algorithm is invertible.

Each matrix must meet the following criteria: balance, non-linearity, completeness, strict avalanche criterion, low Table XOR ,order diffusion, Invertability, the static criteria (independence between the input and data output, the independence between the output and the input data, the independence between the output and the

Journal of Theoretical and Applied Information Technology
20th July 2013. Vol. 53 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

output data), the dynamic criteria (Dynamic independence between the data input and output, the independence between dynamic the input and output of data, dynamic independence between the output and output data), specific criteria (Entire S-BOX and non-contradiction). These criteria are defined and described in detail in [9], [14], [23]. Requirements defined so that each S-BOX must meet are determined by the need to be stable algorithm for linear cryptanalysis and differential at a time.

Therefore, to meet the requirements set forth above, new substitution matrices must be found, which should be applied in the algorithm parameters or function key values and at the same time, these S-boxes must have characteristics same or higher than those used in the AES standard.

4. PROPOSITION FOR GENERATING NEW S-BOX DEPENDING OF MASTER KEY

Having the substitution matrix used in AES core as a basis, it is necessary to identify other same matrices with the same or better characteristic. The principal SBOX (showing in figure), used in the cryptographic operation is regarded as a base..

Based on the S-BOX suggested in AES (SBOX AES) and depending on key used to encrypt the new plaintext substitution matrices are compted (SBOXxor) by the following process:

- First of all select one byte from master key (initial key) Key[i];
- Computting new SBOXxor, where each cell is equal to XOR with selected byte, SBOXxor[x,y]=SBOXAES[x,y] ⊕ Key[i];
- A substitution matrix newly calculated is used for plaintext encryption.

Decryption process will the following approach is used:

- Selecte same byte from key - Key[i];
- Computing new SBOXxor, where each cell is equal to XOR with selecte byte, SBOXxor[x,y]=SBOXAES[x,y] ⊕ Key[i];
- Computing inverse matrices by using SBOXxor, SBOXxor INV=INV (SBOXxor) (structure of the SBOXxor inv showing in the figure);
- A recalculation inverse S-BOX is used to decrypt the plaintext.

By using the above described method 256 substitution matrices have been obtained. One of them SBOXAES ⊕ 0hex is equal to original SBOX suggered in AES core. Some of them are depicted in Table 1 and Table 2 , respectively, SBOX computed by XOR with byte equal to 24hex and 6F hex.

Table 2: $SBOX_{24}$ (S-BOX $\oplus 24_{hex}$).

47	58	53	5F	D6	4F	4B	E1	14	25	43	0F	DA	F3	8F	52
EE	A6	ED	59	DE	7D	63	D4	89	F0	86	8B	B8	80	56	E4
93	D9	B7	02	12	1B	D3	E8	10	81	C1	D5	55	FC	15	31
20	E3	07	E7	3C	B2	21	BE	23	36	A4	C6	CF	03	96	51
2D	A7	08	3E	3F	4A	7E	84	76	1F	F2	97	0D	C7	0B	A0
77	F5	24	C9	04	D8	95	7F	4E	EF	9A	1D	6E	68	7C	EB
F4	CB	8E	DF	67	69	17	A1	61	DD	26	5B	74	18	BB	8C
75	87	64	AB	B6	B9	1C	D1	98	92	FE	05	34	DB	D7	F6
E9	28	37	C8	7B	B3	60	33	E0	83	5A	19	40	79	3D	57
44	A5	6B	F8	06	0E	B4	AC	62	CA	9C	30	FA	7A	2F	FF
C4	16	1E	2E	6D	22	00	78	E6	F7	88	46	B5	B1	C0	5D
C3	EC	13	49	A9	F1	6A	8D	48	72	D0	CE	41	5E	8A	2C
9E	5C	01	0A	38	82	90	E2	CC	F9	50	3B	6F	99	AF	AE
54	1A	91	42	6C	27	D2	2A	45	11	73	9D	A2	E5	39	BA
C5	DC	BC	35	4D	FD	AA	B0	BF	3A	A3	CD	EA	71	0C	FB
A8	85	AD	29	9B	C2	66	4C	65	BD	09	2B	94	70	9F	32

Table 3: $SBOX_{6F}$ (SBOX $\oplus 6F_{hex}$).

0C	13	18	14	9D	04	00	AA	5F	6E	08	44	91	B8	C4	19
A5	ED	A6	12	95	36	28	9F	C2	BB	CD	C0	F3	CB	1D	AF
D8	92	FC	49	59	50	98	A3	5B	CA	8A	9E	1E	B7	5E	7A
6B	A8	4C	AC	77	F9	6A	F5	68	7D	EF	8D	84	48	DD	1A
66	EC	43	75	74	01	35	CF	3D	54	B9	DC	46	8C	40	EB
3C	BE	6F	82	4F	93	DE	34	05	A4	D1	56	25	23	37	A0
BF	80	C5	94	2C	22	5C	EA	2A	96	6D	10	3F	53	F0	C7
3E	CC	2F	E0	FD	F2	57	9A	D3	D9	B5	4E	7F	90	9C	BD
A2	63	7C	83	30	F8	2B	78	AB	C8	11	52	0B	32	76	1C
0F	EE	20	B3	4D	45	FF	E7	29	81	D7	7B	B1	31	64	B4
8F	5D	55	65	26	69	4B	33	AD	BC	C3	0D	FE	FA	8B	16
88	A7	58	02	E2	BA	21	C6	03	39	9B	85	0A	15	C1	67
D5	17	4A	41	73	C9	DB	A9	87	B2	1B	70	24	D2	E4	E5
1F	51	DA	09	27	6C	99	61	0E	5A	38	D6	E9	AE	72	F1
8E	97	F7	7E	06	B6	E1	FB	F4	71	E8	86	A1	3A	47	B0
E3	CE	E6	62	D0	89	2D	07	2E	F6	42	60	DF	3B	D4	79

Journal of Theoretical and Applied Information Technology
20th July 2013. Vol. 53 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Note: in the table 1 we considered 24 $_{hex}$ first one byte on master key, and same idea for 6F $_{hex}$.

4.1 New algorithm using the SBOX depending on the AES key

Now, with Sbox dependent initial key, AES will be much stronger. [24] We now present how the property above of Sbox can be used to master the function key by using one of the two cases (we can also use other cases) depending on the level of the security requirement. In case the demand moderate level of security Case 1 can be used. For more high security requirements case 2 may be adopted.

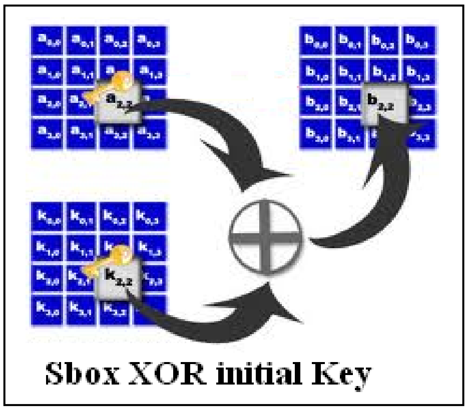

Figure 11:Our Suggestion For Dynamic Sbox Depending Of Initial Key

Case 1:

Here the various round keys are produceded using a expansion key algorithm which is similar to that in AES core key extension algorithm. The round keys thus produceded will used for find a value that is used to change the static Sbox.

The similar round keys are used for Add RoundKey step as well. Supposed for a particular round j, if the round key value is :

X"6172726167736C696D616E5F6D697469", eqyivalent “arragsliman_miti” in code ascii.

The first byte 61(Hex) is used to XOR-ing the Sbox (change the sbox by using XOR operation between each byte of fixe Sbox(showing in Table 1) and the first byte of the master key). The resulting SBOXxor [eache byte,61 hex] is used during the Subbyte operation.

Case 2:

Here different round keys are generated using a key expansion algorithm which is similar to that of AES key expansion algorithm. The round keys thus generated will used for finding a value that is used to rotate the S-box.

The same round keys are used for AddRoundKey stage as well. Suppose for a particular round j, if the round key value is :

X"6172726167736C696D616E5F6D697469".

Here XOR operation of all the bytes is taken.

35(Hex)=61⊕72⊕72⊕61⊕67⊕73⊕6C⊕69⊕6D⊕61⊕6E⊕5F⊕6D⊕69⊕74⊕69.

The resulting byte value 35(Hex) is used to XOR-ing the Sbox (change the sbox by using XOR operation between each byte of fixe Sbox(showing in Table 1) and the 35 (hex)). The resulting Sbox is used during the Sub Bytes operation.

On the basis of the surveyed substitution matrices and depending on the particular value of the encryption key used in Advanced encryption standard the following algorithm is offered:

Encryption:

- Chose a key initial for AES;
- The first byte of the Key master is selected Key[1];
- Computing new SBOXkey[1] = SBOXAES⊕Key[1] (showing in figure 11);
- Continue according to the algorithm set out in AES by using new calculated S-BOXkey[1].

Decryption:

- Chose a key initial for AES;
- The first byte of the Key is selected Key[1];
- Computing new SBOXkey[1] = SBOXAES⊕Key[1];
- Computing inverse SBOXkey[1]INV=INV (SBOXkey[1])=INV (SBOXAES⊕Key[1]);
- Continue as described in the AES algorithm with set out in AES by using new computed SBOXkey[1]INV.

Table 4 : Comparaison Between AES Core And Our Suggestion.

	AES	Dynamique AES (Our suggestion)
Block Length	128-bit	same
Key Lenght	128-192-256 bits	same

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Number of Rounds	For key length 128-bit. 10 Rounds	same
Rounf Function	Composed of 4 transformations, namely: -ByteSub using SBOX -Shift Row -Mix Column -AddRoundKey. For last round Mix Column is eliminated	Composed of 4 transformations, namely: -ByteSub using SBOXxor key -Shift Row -Mix Column -AddRoundKey. For last round Mix Column is eliminated
SBOX	Fixed	initial Key Dependent
Key Expansion	Use the master key and static SBOX	Use the master key and Dynamic SBOXxor key

5. IMPLEMENTATION OF MODIFIED PIPELINED AES IN FPGA

Implementation uses the VHDL programming language that nowadays is commonly a language used very established for FPGA [16]. The drawing & the software of the simulation is Quartus II v9.1.

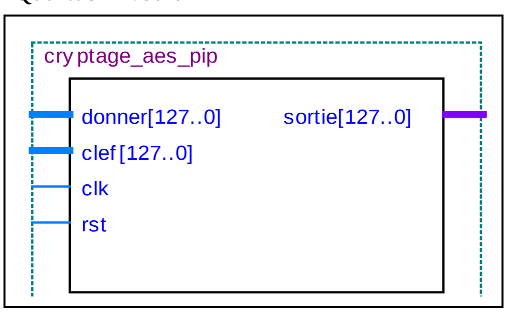

Figure 12: Cryptage AES 128 Pipeline

Figure 13: Decryptage AES 128 Pipeline

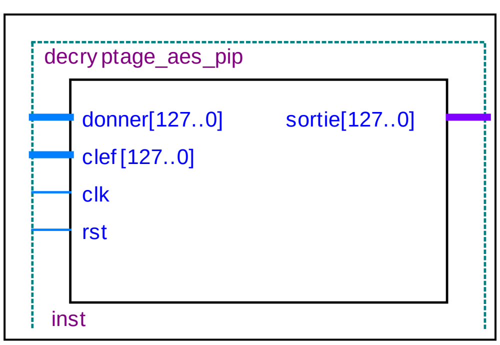

Table 5 : I/O functional descriptions of proposed and modified AES-128.

Pin name	I/O	Function description
CLK	I	System frequency
Rst	I	System reset
Donner	I	Plaintext bits (for Encryption) Encryption bits (for Decryption)
Clef	I	Key for Encryption or decryption
Sortie	O	Encryption bits (for Encryption) Plaintext bits (for Decryption)

Table 6: Comparative Table Between Different implementation constitute AES algorithm

Implementation	FPGA Device				
	Total pins	Total logic elements	Peak virtual memory Megabyte	Total registers	Total memory bits
Crypt_aes_pip	386	13552	232	2432	327680
Decry_aes_pip	386	15827	242	2432	327680
Crypt_aes_pip_ SBOXxor key[i]	386	57366	496	3840	0
Decrypt_aes_pip_ SBOXxor key[i]	386	19471	379	2560	327680

Note: in this different implementation we use two optimization:

- Architectural optimization by using pipelined architecture of AES-128 [12], [13].
- Algoritmic optimization by using modified structure of mixcolumn block (Properties of the binary calculation) [26], [27].

Depending on comparative table we can notice that the first architectural of the Crypt_aes_pip implementation in. occupied more than (13552 units) of the unit when the second Decry_aes_pip require implementing approximately in. (15827 units) in total capacity of the device, on the other

Journal of Theoretical and Applied Information Technology
20th July 2013. Vol. 53 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the third architecture, Crypt_aes_pip_SBOXxor key[i], need (57366 tranches) of the device, and last implementation,Decrypt_aes_pip_SBOXxor key[i], occupied (19471 logic elements).

The first conclusion here ,the advantage in our new implementation is the SBOX and key expansion is now dependent on entire initial key (first byte of master key(case 1) or Xor-ing between each byte of initial key (case 2)).
But the disadvantage is that it consumes little extra time and more logic elements.
Second Conclusion we find our new implemention (crypt_aes_pip_SBOXxor key[i] & Derypt_aes_pip_SBOXxor key[i]) is more efficacious than architecture of the first and second implementation showing in table 5, the occupying number of resources of the device.

6. SIMULATION & INTERPRETATION

The schemas descripted of the simulation the processes for the setting in implementation crypt_aes_pip_SBOXxor key[i] (AES-128 based to change the static sbox by using XOR operation between each byte of fixe Sbox (showing in Tble 1and the first byte of the master key (case1) or (case2)) are presented below, in Figure (15). The overall length of the encryption process is (126 S), otherwise the simulation of decrypt_aes_pip_SBOXxor key[i] (change the sbox by using XOR operation between each byte of fixe Sbox(showing in Table 1) and the first byte of the initial key (case1) or by using case2 when we Xor-ing each byte of masetr key), presented following, in figure (16).encryption is the time to (40 s) and some decoding (s).

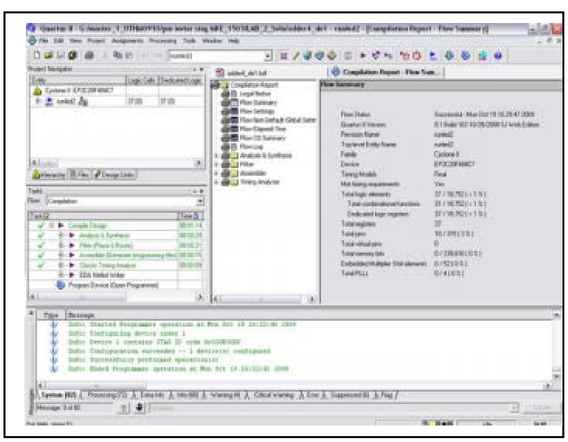

Figure 14 : Compile The Circuit

Ciphering :(figure15)

Plaintext: hamdoun_&_tragha

Key: arragsliman_miti

cyphertext:,A![208]2[237]<W[160]XI[218][225][19]|[30]

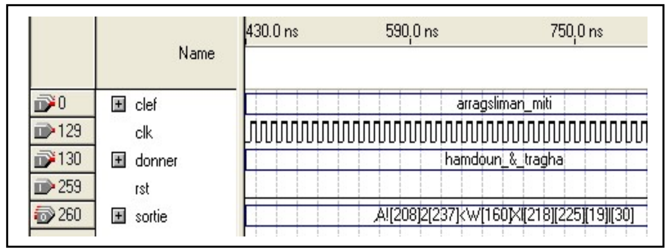

Figure 15:Simulation Of Crypt_Aes_Pip_Sboxxor Key[I]

Deciphering :(figure 16)

plaintext :,A![208]2[237]<W[160]XI[218][225][19]|[30]

Key: arraglsiman_miti

Cyphertext: hamdoun_&_tragha

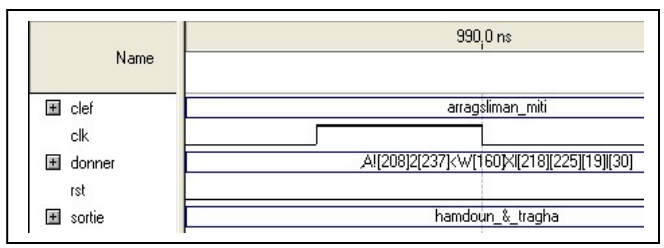

Figure 16:Simulation Of Decrypt_Aes_Pip_Sboxxor Key[I]

Note: during our implementation and simulation we used two different software,the first is altered UP Simulator and the second is Quartus II v9.1.

7. CONCLUSION

The first conclusion, in this paper new substitution matrices have been developed by XOR with byte key and chosen from existing AES S-BOX. These matrices were tested with simulation software developed by Quartus II V.9.1. Analysis of the results shows that the characteristics of the new 256 S-BOX are identical, based on which the conclusion was reached that it is possible to use

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

each for encryption. It will not lead to a deterioration of the stability of the AES linear cryptanalysis and differential. An algorithm for the use of these matrices is proposed, as it is based on a preselected byte of the key used, and depending on the result in XOR operation, one of the S-BOX 256 is selected.

Second conclusion, we find the changes proposed (Sbox dependent of initial key) in our paper although consumes little extra time and more logic elements, but can be implemented without changing the block size keys (128, 192 or 256). Even if the original AES algorithm is very secure, These proposed changes in the treatment of the algorithm will encrypt the information by performing high diffusion and confusion. It also increases the complexity of the AES algorithm several times,so AES will be much stronger.

8. POSSIBLE FUTURE WORK

From our perspective is to do:

- Implementation and simulation the other optimize structure and description based on that same principle of AES algorithm to encrypt and decrypt data such as images and sounds.
- Study of FPGA components and choosing the most suitable architectures for the implementation of encryption algorithms.
- A combination between the AES algorithm and genetic algorithm to increase the security.
- Implementation of the AES pipeline architecture to optimize the encryption and decryption time and number of clock cycles.
- New instructions ensure a faster and more affordable encryption of data, as well as better security.

REFRENCES:

[1] NIST, Advanced Encryption Standard (AES), Springfield,VA, Nov. 2001

[2] Simon Singh : Histoire des codes secrets. De l'Egypte des pharaons à l'ordinateur quantique. Paru chez J.-C. Lattès, Paris, 1999.

[3] Stallings W. "Cryptography and Network Security: Principles and Practices."4th ed. Pearson Education, Inc. pp. 63-173. 2006.

[4] Bhupathi Kakarlapudi and Nitin Alabur," FPGA Implementations of S-box vs. T-box iterative architectures of AES".

[5] Olivier Frider ETR6 « Advanced Encryption System », école d'ingénieurs du Canton de Vaud, Mai 2004.

[6] Ashwini M. D, Mangesh S. D and Devendra N. K ",FPGA Implementation of AES Encryption and Decryption", Proceeding of International Conference On Control, Automation, Communication And Energy Conservation - 2009.

[7] Daemen J. and Rijmen V., "Rijndael: The Advanced Encryption Standard". Dr. Dobb's Journal, March 2001.

[8] NIST, "DRAFT NIST Special Publication 800-131, Recommendation for the Transitioning of Cryptographic Algorithms and Key Sizes", Federal Information Processing Standards Publication (FIPS PUB) 197, National Institute of Standards and Technology (NIST), January, 2010.

[9] Nikolai Stoianov, AES S-BOX generator: analysis of requirements, International Science Conference 2009" Communication and information systems", Shoumen, Bulgaria,2010.

[10] Qin H., Nonmember, SASAO T. and IGUCHI Y.,Members ,"A Design of AES Encryption Circuit with 128 bit keys using Look-UP Table Ring on FPGA",IEICE TRANS. INF. & SYST.,VOL.E89-D,NO.3 MARCH 2006.

[11] Rahman T., Pan S. and Zhang Q., "Design of a High Throughput 128-bit (Rijndael Block Cipher)", Proceeding of International Multiconferrence of Engineers and computer scientists 2010 Vol II IMECS 2010, March 17-19,2010, Hongkong.

[12] Hodjat A. and Varbauwhede I.,"A 21.54 Gbits Fully Pipelined AES Processor on FPGA", IEEE Symposim on Field-Programmable Custom Computing Machines,April 2004.

[13] Jarvinen et al, "A fully pipelined memoryless 17.8 Gbps AES-128 encrypter",International Symposium on Field Programmable Gate arrays,pp.207-215.2003.

[14] INDECT Consortium, D8.2: Evaluation of Components, June, 2010, http://www.indect-project.eu/files/deliverables/public/deliverable 8.2.

[15] Rijndael mix column, available at: http://en.wikipedia.org/wiki/Rijndael_mix_columns

[16] Mroczkowski P., "Implementation of the block cipher Rijndael using Altera FPGA", May 2000.

[17] Eli Biham and Nathan Keller, Cryptanalysis of Reduced Variants of Rijndael, In Proceedings

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Of The Third Advanced Encryption Standard Conference. NIST, April 2000.

[18] Zambreno J., Nguyen D. and Choudhary A., "Exploring Area/Delay Tradeoffs in an AES FPGA Implementation",FPL 2004, LNCS 3203, pp. 575–585, 2004.

[19] T. Jacobsen and L. R. Knudsen, The Interpolation Attack on Block Ciphers, Fast Software Encryption, LNCS 1267, E. Biham, Ed., Springer – Verlag, 1997, pp. 28 – 40.

[20] Kenny D., "Energy Efficiency Analysis and Implementation of AES on an FPGA", University of Waterloo,2008.

[21] X.Ji-peng ,Z.Xue-cheng, G.Xu, "Ultra-low power S boxes architecture for AES", The journal of China Universities of post and telecommunications.vol.15,issue1,March 2008.

[22] M.Priya Zach, K.Rahimunnisa and K.Suresh Kumar, "Compact AES Architecture Using Efficient S-Box Implementation", IEEE International Conference on Computational Intelligence and Computing Research, 1373-1376, 2011.

[23] Nikolai Stoianov, One Approach of Using Key-Dependent S-BOXes in AES, MCSS 2011, CCIS 149, pp. 331–337, 2011. Springer-Verlag Berlin Heidelberg, 2011.

[24] William Stallings, Cryptography and Network Security, Third Edition, Pearson Education, 2003.

[25] Edwin NC Mui," Practical Implementation of Rijndael S-Box Using Combinational Logic".

[26] A. Sliman, H. Abdellatif, T.Abderrahim, K. Salah eddine. "Implementation of The Encryption algorithm AES under VHDL language In FPGA, by using different architecture of mixcolumn". Call of paper WOTIC'11 13-15 Octobre, ENSEM de Casablanca.

[27] A. Sliman, H. Abdellatif, T.Abderrahim, K. Salah eddine, "Implementation of The Encryption algorithm AES under VHDL language In FPGA by using different architecture of mixcolumn," International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.4, August 2012.