Library Progress International
Vol.44 No.3, Jul-Dec 2024: P.19738-19744

Print version ISSN 0970 1052
Online version ISSN 2320 317X

Original Article

Available online at www.bpasjournals.com

A Highly Efficient, Modular and Portable FPGA Implementation of AES Cryptography

Dr. Manisk Kuamr
Professor,
Poornima University, Jaipur, Rajasthan,India
manish.kumar1@poornima.edu.in

Ms. Shikha Sharma
Professor,
Poornima University, Jaipur, Rajasthan,India
hod.ce@poornima.edu.in

Dr. Ashish Avasthi *Professor,*
Poornima University, Jaipur, Rajasthan,India
Ashish.avasthi@poornima.edu.in

Abstract – **This article describes the core implementation of an Advanced Encryption Standard - AES in Field Programmable Gate Array - FPGA. The core was implemented in both Xilinx Spartan-3 and Xilinx Virtex-5 FPGAs. The algorithm was implemented for 128 bits word and key. The implementation was very efficient, achieving 318MHz on a Xilinx Spartan-3, representing at 50% faster than other reported works. The implementation can achieve 800MHz on a Xilinx Virtex-5. The main goal of this work was the implementation of a fast and modular AES algorithm, as it can be easily reconfigured to 128, 196 or 256 bits key, and can find a wide range of applications. Nevertheless, all the reported works used as comparison basis to our work were also implemented using 128 bits key.**

Keywords: Cryptography, AES, DES, FPGA, efficient encryption/decryption implementation.

I. INTRODUCTION

In 1997, the NIST (National Institute of Standards and Technology) released a contest to choose a new symmetric cryptograph algorithm that would be called Advanced Encryption Standard – AES to be used to protect confidential data in the USA. The algorithm should meet few requirements such copyright free, faster than the 3DES, cryptograph of 128 bit blocks using 128, 192 and 256 bit keys, possibility of hardware and software implementation, among others. In 2000, after analysis by cryptography experts, it was chosen the winner: Rijndael. The algorithm was created by the Belgians Vincent Rijmen e Joan Daemen [1][2].

The algorithm was implemented in FPGA due to its flexibility and reconfiguration capability. A reconfigurable device is very convenient for a cryptography algorithm since it allows cheap and quick alterations.

Section II provides a brief introduction of AES and its processing phases. Section III describes the chosen FPGA and the circuit implementation. Section IV compares the results of this work with others presented in the literature. Section V presents the conclusions of this work and finally Sections VI

shows the authors expectations and proposals for continuing this work.

II. AES RIJNDAEL

In order to better understand the AES structure it is necessary to know the definition of state, in the algorithm. State is the matrix of bytes that is processed between the many stages, or rounds, and therefore, it will be modified in each stage. In the Rijndael algorithm, the matrix size depends on the block size being used, composed of 4 lines and Nb columns. Here, Nb is the number of bits in the block, divided by 32, since 4 bytes represent 32 bits. Since the AES algorithm uses 128 bit blocks, the state will be composed by 4 lines and 4 columns [3].

The key is grouped by the same fashion as the data block, whereas Nk is the number of columns. Nr is the number of rounds that will be run during the algorithm. The number of runs in the AES will depend on size of the key, where Nr will be 10, 12 and 14, for Nk equals to 4, 6 e 8, respectively [1].

On the encryption algorithm, there will be 4 phases: AddRoundKey, SubBytes, ShiftRows and MixColumns. Nevertheless, on the last stage, the MixColumns operation is suppressed. The decryption algorithm will use the respective inverse operations: InvAddRoundKey, InvSubBytes, InvMixColumns and InvShiftRows. As it was in the encryption phase, the InvMixColumns is suppressed on the last stage of decryption algorithm [2].

The algorithm will be explained based on its specification. The values shown in the example are presented in hexadecimal format.

A. SubBytes

Each state byte is replaced by another in the S-box (replacement Box), as indicated in Fig. 1. The replacement follows a matrix, where the first hexadecimal value corresponds to the line positioning, and the second hexadecimal value corresponds to the column positioning. The inverse operation (decryption) is called InvSubBytes, and uses an inverse S-Box.

As an example, the S-box outputs 24 for the input value A6. On the same way, the inverse S-Box outputs A6 for the input value 24.

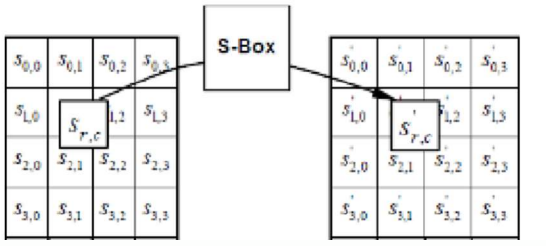

Figure 1 - SubBytes operation process.

B. ShiftRows

It consists of a left shift on the state lines, replacing therefore their byte position, as indicated in Fig. 2. Line 0 suffers 0 shifting. Line 1 is shifted by one position and line 2 undergoes do 2 shifting positions. Line 3 is shifted by 3 positions.

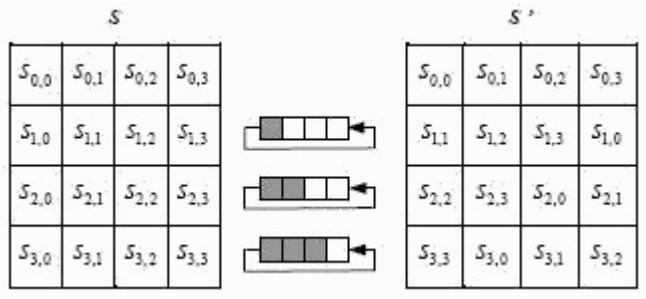

Figure 2 - ShiftRows operation process.

The decryption algorithm performs the inverse operation InvShiftRows that consists of similar shiftings as the ShiftRows, but shifted to the right.

C. MixColumns

In this operation, the state bytes are treated as polynomials of Galois Field algebra GF(2^8) [4]. The operation can be represented as a matrix multiplication, as indicated in Fig. 3, where S is the initial state and S´ is the final state, after the operation.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Figure 3 - MixColumns operation process.

The inverse operation, the InvMixColumns, consists of the multiplication using the inverse matrix.

In the last round, on both the encryption and decryption algorithms, the MixColumns operation is suppressed.

The C matrix (used in the encryption) and C´ matrix (used in the decryption) are:

$$C = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \qquad C' = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$

D. AddRoundKey

It is an XOR operation between the state and the round key that it is generated from the main key through the Key Generation. The matrix of keys is represented by w columns or $k_{x,y}$ cells. AddRoundKey is used both in the encryption and decryption algorithms. The XOR is conducted on byte basis, as indicated in Fig 4, where the new byte $s'_{x,y}$ is given by $s_{x,y} \oplus k_{x,y}$.

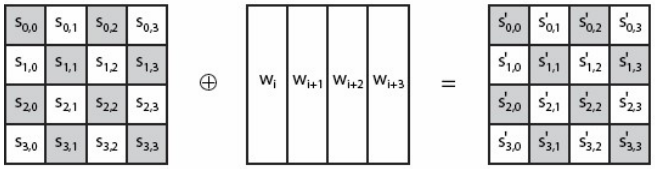

Figure 4 - AddRoundKey operation process

E. Key Expansion

The Key size defines the number of rounds in the encryption/decryption algorithm, and it also defines its expansion process. Basically, the Key Expansion operation consists of three operations, as presented in Fig. 5. The first operation, RotWord, makes a one byte circular shifting on the word. The second operation, SubWord replaces each byte of the input word according to the S-Box. The third operation consists of XOR operations, as indicated in Fig.5.

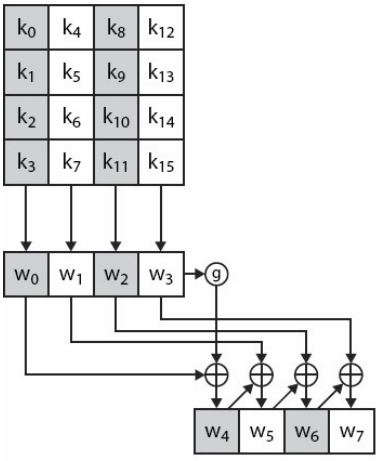

Figure 5 - KeyExpansion operation process

III. IMPLEMENTATION

The AES hardware was implemented in three modules: the encryption, the decryption and the key expansion module. All the modules were independently tested and characterized,

and therefore they can be used in any combination, according to the application.

In order to conduct tests on all blocks, it was assembled a 128 bits encryption - decryption AES set in a Xilinx Spartan-3 FPGA. The test results are presented in Section IV.

After the tests on the Xilinx Spartan-3 FPGA, the hardware was also tested on a Xilinx Virtex-5 FPGA.

The VHDL description implemented on both FPGAs is exactly the same, and no change was made in the VHDL description to fit any of the FPGAs. Another important information is that the code is totally portable, the code can be used in any FPGA family because was developed using the VHDL patterns.

The hardware implemented is illustrated in Fig. 6. It is composed of two 128 bit inputs that receive the key and the initial word to be encrypted. The signal *in_aes_mode* defines an encryption or decryption operation. The *load* inputs are used to indicate that the data at the input is valid and can be loaded. The signal *out_busy* signals if the circuit is busy processing a word or is available for a new word.

On a Xilinx Virtex-5 FPGA, the initial encryption load takes 20ns and the decryption loads takes 30ns. The decryption loading process takes 10 cycles longer than the encryption since it requires loading and process the entire key in order to start the decryption process.

On the encryption process, at each key expansion it is possible to encrypt the word at next cycle.

On a Xilinx Virtex-5 FPGA, the cryptograph of each word runs at approximately 60MHz, since the hardware takes 12 clock cycles to process it. The FPGA operates at approximately 800MHz, as it can be seen from the listing shown in Fig 7.

The implementation of encryption/decryption algorithms using pipelining is left as a suggestion to improve this work. With this suggestion we believe that the hardware efficiency will be improved achieving high frequencies. Using a pipeline with 10 levels the frequency of cryptography of each word will be near to 800MHz.

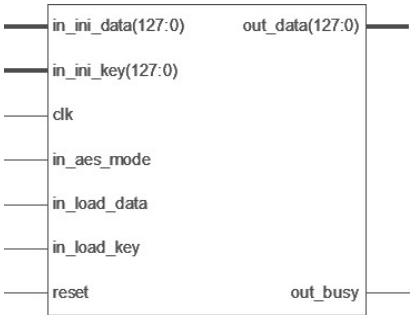

Figure 6 - Inputs and Outputs of developed AES128.

```
FPGA device: Spartan-3 XC3S4000

Speed Grade: -5
Minimum period: 3.140ns
Maximum Frequency: 318.492MHz
Minimum input arrival time before clk: 9.834ns
Maximum output req time after clk: 6.216ns

FPGA device: Virtex-5 XC5VFX70T
Speed Grade: -1
Minimum period: 1.116ns
Maximum Frequency: 896.057 MHz
Minimum input arrival time before clk: 2.300ns
Maximum output req time after clk: 3.524ns
```

Figure 7 – Summary of FPGA speed achieved.

IV. Results Comparison

It was chosen the Xilinx Spartan-3 (XC3S4000) to conduct performance comparison of our work with others [5][6][7][8][9][10][11], since it was used to implement many of them. Table I summarizes the performance comparison.

As can be observed from the table, our work is at least 50% faster that the fastest circuit reported. Nevertheless, the Xilinx Virtex-5 offers an even higher speed.

TABLE I
COMPARISON WITH OTHER FPGA IMPLEMENTATIONS

Implementation	*Platform Device*	*Data Path*	*Frequency (MHz)*
C. Chien [5]	Xilinx Virtex-II (XC2V1000)	128	75
I. Aigredo-Badillo [6]	Xilinx Virtex-II (XC2V1000)	128	96.42
J. Zambreno [7]	Xilinx Virtex-II (XC2V4000)	128	110.16
E. J. Swankoski [8]	Xilinx Virtex-II Pro (XC2VP50)	128	145.05
E. Lopez-Trejo [9]	Xilinx Spartan-3 (XC3S4000)	128	100.08
A. Aziz & N. Ikram [10]	Xilinx Spartan-3 (XC3S50)	128	165
Dur-e-Shahwar, Zaka, Qurat-Ul-Ain and Aziz [11]	Xilinx Spartan-3 (XC3S4000)	128	206.28
Our Design	**Xilinx Spartan-3 (XC3S4000)**	**128**	**318.49**
	Xilinx Virtex-5 (XC5VFX70T)	**128**	**896.05**

V. Conclusions

This article presented a new and efficient AES cryptography hardware structure that can have many applications. The circuit implementation is very efficient and can be customized to a wide range of applications. It represents an improvement to support new applications.

VI. Future Work

The Microelectronics Group at Universidade Federal de Itajuba intends to use this work as part of larger projects, including smart metering and cryptography interface.

References

[1] FIPS-197, "Federal Information Processing Standards Publication FIPS-197, Advanced Encryption Standard (AES)", http://csrc.nist.gov/publications!fips!fips 1 97/1Ips-197.pdf, October 1999.

[2] Daemen, J. and Rijmen, V. (2002). The design of Rijndael: AES — The Advanced Encryption Standard. Springer-Verlag.

[3] Daemen, J. and Rijmen, V. A Specification for The AES Algorithm. NIST (National Institute of Standards and Technology). http://csrc.nist.gov/archive/aes/rijndael/wsdindex.html , October 2010.

[4] Klima, R. E., Sigmon, N., and Stitzinger, E. (2000). Applications of abstract algebra with Maple. CRC Press, Boca Raton, FL.

[5] C. Chien, D. Chien, C. Chien, I. Verbauwhede and F. Chang, "A hardware implementation in FPGA of the Rijndael algorithm", *The 2002 45th Midwest Symp. Circuits and Systems (MWSCAS-2002),* Vol. 1,4--.7 August 2002, pp. 507-509.

[6] I. Algredo-Badillo, C. Feregrino-Uribe and R. Cumlido-Parra, "Design and implementation of an FPGA-based 1.452 Gbps non-pipelined AES architecture', *The 2006 Int. Con! Computational Science and Its Applications (ICCSA 2006),* Lecture Notes in Computer Science, Vol. 3982 (Springer-Verlag, 2006), pp. 446--455.

[7] J. Zambreno, D. Nguyen and A. Choudhary, "Exploring area/delay tradeoffs in an AES FPGA implementation", *Proc. Int. Colif, FieldProgrammable Logic and Its Applications (FPL),* Lecture Notes in Computer Science, Vol. 3203 (Springer-Verlag 2004), pp. 575-585.

[8] E. J. Swankoski, V. Narayanan, M. Kandemir and M. J. Irwin, "A parallel architecture for secure FPGA symmetric encryption", *18th Int. Parallel and Distributed Processing Symp. (IPDPS'04) - Workshop,* Santa Fe, New Mexico, 26-.30 April 2004, p. 123.

[9] E. Lopez-Trejo, F. Rodriguez-Henriquez and A. Diaz-Perez, "An efficient FPGA implementation of CCM using AES", *The 8th Int. Con! Information Security and Cryptology (ICJSC'05).* Lecture Notes in Computer Science (Springer 2005), pp. 208-215.

[10] Arshad Aziz and Nassar Ikram, "Memory efficient implementation of AES S-boxes on FPGA", *Journal of Circuits, Systems, and Computers,* Vol. 16, No.4 (2007) 603--611.

[11] Dur-e-Shahwar Kundi, Saleha Zaka, Qurat-Ul-Ain and Arshad Aziz "A Compact AES Encryption Core on Xilinx FPGA" (2009) of 2nd IEEE International Conference on Computer, Control & Communication (IEEE IC4-2009) Karachi, Pakistan Vol:1 pp:1-4