RECONFIGURABLE HARDWARE (FPGA) IMPLEMENTATION OF CRYPTOGRAPHIC ALGORITHMS - AES ENCRYPTION

Paul BURCIU, Ionuț Mihai SIMA
University of Pitești, Electronics Communications & Computer Faculty
E-mail: pburciu@yahoo.com, mihaibb2000@yahoo.com

Keywords: hardware implementation, cryptographic algorithm, FPGA platform, cryptographic module, ASIC chip, AES, VHDL code, hardware simulation.

Abstract: *The hardware implementation of cryptographic algorithms is a timely method, providing efficient security solutions, both regarding the processing speed and the consumed power. The present-day FPGA platforms, which are the physical groundwork for such implementations, however they are not new engineering solutions, assert as the most efficient way to practically transpose the cryptographic algorithms, resulting optimized (concerning diversified aspects) cryptographic modules, respectively in the end, unmatched (concerning performance) ASIC chips. This paper presents a new hardware implementation for the enciphering block of the AES (Advanced Encryption Standard) symmetric cryptographic algorithm, using VHDL programming language and a hardware simulation of the resulted enciphering module.*

1. INTRODUCTION

The cryptographic algorithms became the main proceeding for protection of very important data, the security objective called *confidentiality* being the one taken into account by their hardware implementation and by their integration into the present-day communication systems.

Among the diverse cryptographic algorithms, the symmetric algorithms may be considered as the most susceptible of being hardware implemented, because the mathematical mechanisms used by them contain arithmetical operations which can be executed by logical combinational or sequential circuits, namely both by ordinary logical gates and by Finite State Machines (FSM), according to Church-Turing Thesis [1], in other words by VLSI digital integrated circuits.

The international contest organized by the National Institute of Standards and Technology (NIST) in 1997 for selection of the new symmetric cryptographic algorithm, which was intended to replace the old DES, succesfully attacked and proved to be insecure, imposed in 2000 the Belgian algorithm RIJNDAEL as the winner and designated as the American cryptographic standard, under the name of Advanced Encryption Standard (AES), by the FIPS PUB 197 in 2001 [2].

The hardware implementation of cryptographic algorithms is a timely method, providing efficient security solutions, both regarding the processing speed and the consumed power. The present-day FPGA platforms, which are the physical groundwork for such implementations, however they are not new engineering solutions, assert as the most efficient way to practically transpose the cryptographic algorithms, resulting optimized (concerning diversified aspects) cryptographic modules, respectively in the end, unmatched (concerning performance) ASIC chips.

This paper presents a new hardware implementation for the enciphering block of the AES (Advanced Encryption Standard) symmetric cryptographic algorithm, using VHDL programming language and a hardware simulation of the resulted enciphering module.

2. THE BASIC CRYPTOGRAPHIC ARCHITECTURE

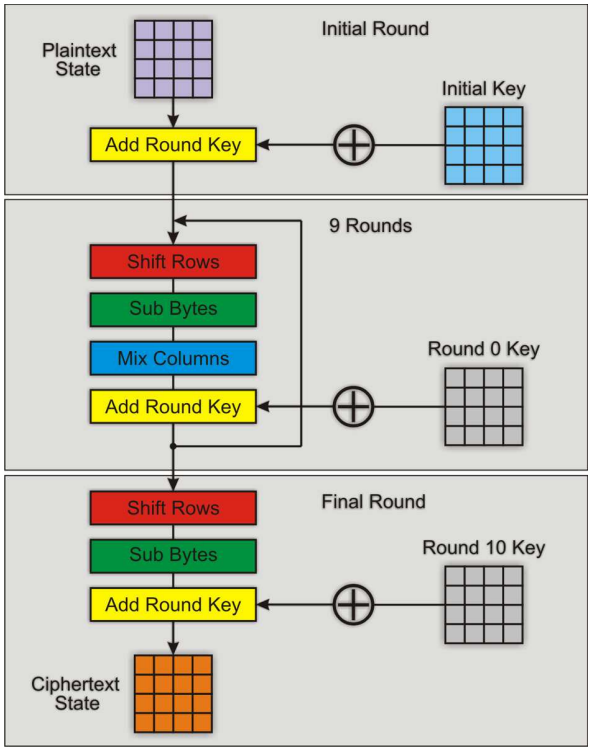

Fig. 1 - The basic AES-128 cryptographic architecture.

The architecture of the AES block enciphering algorithm contains, as described in Figure 1, an initial round where at its input a 128 bits plaintext is applied, consisting of 4 columns with 4 bytes each, or in other words, 4 words with 32 bits each.

The plaintext is XORed with the initial key, also represented as a data block consisting of 4 words with 32 bits each. The algorithm specification contained by [2] mentions 3 possible versions for the AES encryption algorithm, respectively N_k = 4, 6, 8 (the number of 4 bytes columns, or 32 bits words, of the key) corresponding to AES-128, AES-192 and AES-256 versions, also uniquely N_b = 4 (the number of 4 bytes columns, or 32 bits words, of the plaintext), namely 128 bits. The architecture presented by Figure 1 refers to the AES-128 version.

For the 3 constructive versions we have 3 different round numbers, as shown by the presented table in Figure 2.

	Key Length (N_k words)	Block Size (N_b words)	Number of Rounds (N_r)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Fig. 2 - AES versions characteristics.

After the initial round AES-128 contains 9 encryption rounds. Each of these rounds has a corresponding round key supplied by the key schedule mechanism which starts the key generation process with the initial secret key. Every round consists of 4 main operations: Shift Rows (every row of the State is shifted by a specific number of positions), Sub Bytes (the substitution of every byte of the State using a special substitution table), Mix Columns and the Add Round Key (the round key is XORed with the resulted State of the previous operation, namely Mix Columns). This sequence of operations is iterated 9 times.

Mix Columns is a Galois Field (2^8) multiplication between the State, each of its columns being polynomials over GF(2^8), and a fixed polynomial given by $a(x)=\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$ or

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}, \text{ for } 0 \le c < N_b .$$

The final round differs from the others by the lack of the Mix Columns operation. Thus, this round only consists of Shift Rows, Sub Bytes and an Add Round Key operation. Finally, the result of this last round is the ciphertext State, which is a matrix with 4 words, exactly like the plaintext State.

The key schedule uses a key expansion routine which provides the encryption algorithm with N_b (N_r + 1) words, namely 44 key words, or 11 round keys, for the AES-128 encryption version. The resulting key schedule consists of a linear array of 4 byte words, denoted [w_i], with i in the range $0 \le i < N_b(N_r + 1)$. The expansion routine uses some specific functions, like SubWord(), which takes a 4 byte input word and applies the substitution S-box to each of the 4

bytes to produce an output word, and RotWord(), which takes a word $[a_0,a_1,a_2,a_3]$ as input, performs a cyclic permutation, and returns the word $[a_1,a_2,a_3,a_0]$.

The round constant word array, Rcon[i], contains the values given by $[x^{i-1},\{00\},\{00\},\{00\}]$, with x^{i-1} being powers of x, where x is denoted as {02} in the field GF(2^8). Every following word, w[i], is equal to the XOR of the previous word, w[i-1], and the word N_k positions earlier, w[i-N_k]. For words in positions that are a multiple of N_k, a transformation is applied to w[i-1] prior to the XOR, followed by a XOR with a round constant, Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word (RotWord()), followed by the application of a table lookup to all four bytes of the word (SubWord()).

3. FPGA IMPLEMENTATION OF AES ENCRYPTION

The main goal of this hardware implementation is not speed, but the area & resource limitations of a specific target FPGA device, respectively Xilinx Virtex-4 (model XC4VFX12 - FF668) which is a low resource platform, namely: 5472 slices, 320 I/O buffers, 10944 LUTs with 4 inputs. A summary of the occupied resources is presented in Figure 3.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	1,137	10,944	10%
Number of 4 input LUTs	3,515	10,944	32%
Logic Distribution			
Number of occupied Slices	2,261	5,472	41%
Number of Slices containing only related logic	2,261	2,261	100%
Number of Slices containing unrelated logic	0	2,261	0%
Total Number of 4 input LUTs	3,515	10,944	32%
Number of bonded IOBs			
Number of bonded	261	320	81%
Number of BUFG/BUFGCTRLs	1	32	3%
Number used as BUFGs	1		

Fig. 3 - The occupied device resources

The implementation uses the VHDL programming language, which nowadays is a well-established commonly used language for FPGAs. The design & simulation software is Xilinx ISE 10.1.

The encryption block is represented in Figure 4, where the main signals used by the implementation are shown.

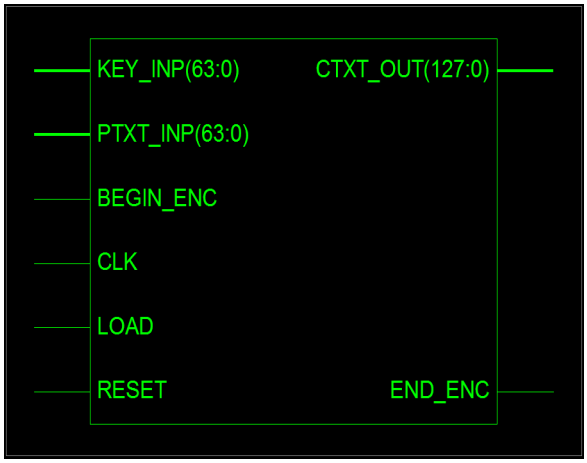

Fig. 4 - The AES Encryption Block

The limitations of this device determined the use of 64 bit inputs, consequently loaded, firstly on the LOW-HIGH transition, and secondly on the HIGH-LOW transition, both in case of the input key and in case of the input plaintext. The use of 128 bit inputs would easily lead to exceeding of the I/O buffer resources. No buffer limitations are imposed to the output, which is a 128 bit ciphertext.

The main signals are: the system clock (CLK), the system reset (RESET), LOAD signal which loads the key and the plaintext in the initial round, and BEGIN_ENC/END_ENC which starts/ends the encryption process. The loading process, which is described above, will be immediately followed by BEGIN_ENC signal which will start the encryption process. The entire encryption process takes exactly 12 clock periods from the HIGH-LOW transition of the LOAD signal. The signals diagram is represented in the simulation chapter in Figure 5.

In theory there are 2 main types of hardware implementations of the block ciphers: iterative and pipelined implementations [3]. Some characteristics of the iterative architectures are: knowing that the block ciphers are themselves of iterative nature (n iterations of the same algorithm are made for a single encryption/decryption, and n clock cycles ar used for a single encryption/decryption), the iterative structure is a natural choice for implementation, occupying small areas of the FPGA devices, with the disadvantage of low throughput. Iterative architectures implement a reduced number of rounds, usually one, in an independent way. The

iterative architectures would be useful for applications where hardware area is limited and speed is not critical, which is the case of our target FPGA device. A pipelined version of the AES implementation was studied but easily exceeded the buffer resources of the target device.

The other choice, the pipelining, allows to process multiple blocks of data at the same time, at different stages, to have higher throughput. Pipelining is achieved by putting rows of registers among different stages.

This implementation for the AES encryption reffers to the case of Electronic Code Book (ECB) mode of operation.

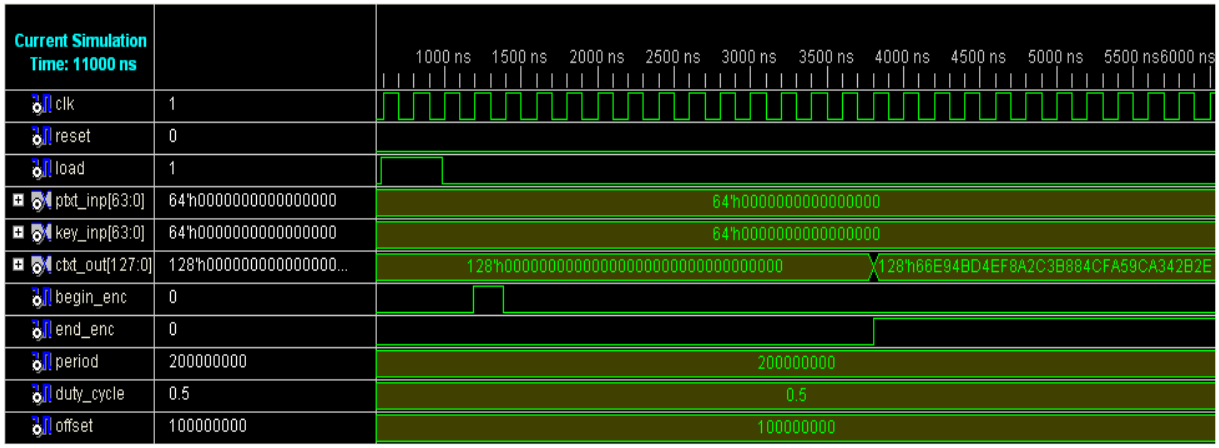

Fig. 5 - Simulation of the encryption block

4. SIMULATION & SECURITY ASPECTS

A detailed diagram of the simulation process for the AES implementation is presented above, in Figure 5. The total duration of the encryption process is 2400 ns or 2.4 µs (approx. 53 Mb/s), calculated as described in the previous section. The resulted maximum clock frequency is 219.695 MHz ≈ 220 MHz, corresponding to a clock period of 4.552ns. The simulation used a sequence of '0' of 128 bits, both for the input plaintext and the secret input key. The output sequence is as shown in Figure 5: '66E94BD4EF8A2C3B884CFA59CA342B2E' (in hexadecimal code).

To be remarked that the END_ENC signal announces the final result of the encryption: when '1', the ciphertext appears on the output.

The evaluation of the security provided by this implementation must take into consideration the security requirements for cryptographic modules stipulated by [4]. In fact, the security of the cryptographic module stands on the security provided by the cryptographic algorithm itself. In [5], a complex evaluation of the 5 finalists of the NIST's international contest concludes that using S-boxes as non-linear components, 'Rijndael appears to have an adequate security margin, but has received some criticism suggesting that its mathematical structure may lead to attacks.

On the other hand, the simple structure may have facilitated its security analysis during the timeframe of the AES development process.' Even if some critiques were formulated (e.g. 'the key schedule does not have high diffusion'), Rijndael was highly appreciated for the provided level of security. Regarding attacks on implementations, [5] remarks that 'the operations used by Rijndael are among the easiest to defend against power and timing attacks.

5. CONCLUSIONS

This paper has presented a brief description of the implementation of the AES block encryption algorithm, underlining the benefits of this modern design concept. An FPGA implementation of an encryption algorithm is a cryptographic module device in which the structure is software implemented.

The FPGA implementations allow us to increase flexibility, lower costs, and reduce time to release enhanced cryptographic equipment,

providing a satisfactory level of security for communication applications, or other electronic data transfer processes where security is needed.

REFERENCES

[1] Wenbo Mao, 'Modern Cryptography. Theory And Practice', Prentice Hall PTR., ISBN: 0-13-066943-1, U.S.A., 2003.

[2] Federal Information Processing Standards, 'FIPS PUB 197 – Announcing the Advanced Encryption Standard (AES)', U.S.A., 2001.

[3] Francisco Rodriguez-Henriquez, Nazar Abbas Saqib, Cetin Kaya Koc, 'Cryptographic Algorithms on Reconfigurable Hardware', Springer Science+Business Media, ISBN: 0-387-33883-7, U.S.A., 2006.

[4] Federal Information Processing Standards, 'FIPS PUB 140-3 - Security Requirements For Cryptographic Modules (Draft)', U.S.A., 2007.

[5] National Institute of Standards and Technology, 'Report on the Development of the Advanced Encryption Standard (AES)', 2000.