An FPGA Hardware implementation of the Rijndael block cipher

Chorfi DHOHA, Slim BEN OTHMAN, Slim BEN SAOUD

LECAP-EPT / INSAT, Tunisia
boslim@yahoo.fr, slimbensaoud@fulbrightweb.org

Abstract—**In this paper, we present a hardware implementation of an Advanced Encryption Standard (AES) Rijndael (128-bit block and 128-bit key) using Xilinx development tools and Spartan FPGA circuits. All the modules in this core are described by using VHDL language.**
The developed Rijndael core is aimed at providing sufficient performance with good area efficiency. In fact, the encryption/decryption data path operates at 29,45MHz resulting in a throughput of 289,98 Mbits per second for the encryption and 157,1 Mbits per second for decryption. Encryption/decryption circuit will fit in one Xilinx Spartan XC2S600E circuit taking approximately 87% of the area (6068 Slices).
Compared to software implementation, migrating to hardware provides higher level of security and faster encryption speed.

Keywords— AES, Rijndael, Encryption, Decryption, hardware implementation.

I. INTRODUCTION

In 1997, The National Institute of Standards and Technology (NIST) initiated an effort towards developing a new encryption standard, called Advanced Encryption Standard (AES) [1,2]. The efficiency of hardware implementations of the AES candidates has been one of the major criteria used by NIST to select the new federal standard from among five final candidates.

The cipher Rijndael is one of the five finalists of the Advanced Encryption Standard. The algorithm has been designed by Joan Daemen and Vincent Rijmen [3,4].

In October 2000, Rijndael was accepted as AES to achieve security and privacy while processing, transferring and storing the data. Since then, there have been many different hardware implementations [5,6].

Field Programmable Gate Arrays (FPGAs) are an ideal platform for performing hardware acceleration of computationally intensive operations. They can easily be reconfigured and provide much greater flexibility than Application-Specific Integrated Circuit (ASIC) devices [7].

In this paper we present the hardware implementation with 128-bit blocks and 128-bit keys, using Xilinx FPGA and development environment.

II. RIJNDAEL BLOCK CIPHER

Rijndael is a 128-bit symmetric block cipher which encrypts variable size blocs with variable size keys. The specification allows the utilization of key sizes of 128, 192 and 256 bits, and blocks of 128, 192 and 256 bits too. The Rijndael block cipher is comprised of 10, 12 or 14 rounds.

II.1. Rijndael encryption and decryption

In AES-128, the cipher uses 10 rounds. Each round executes the same functions in the same order. Only one round, the first one in decryption and the last one in encryption, does not execute the Mix Column (or, in the decryption case, the InvMixColumn) as it is illustrated in Figure1.

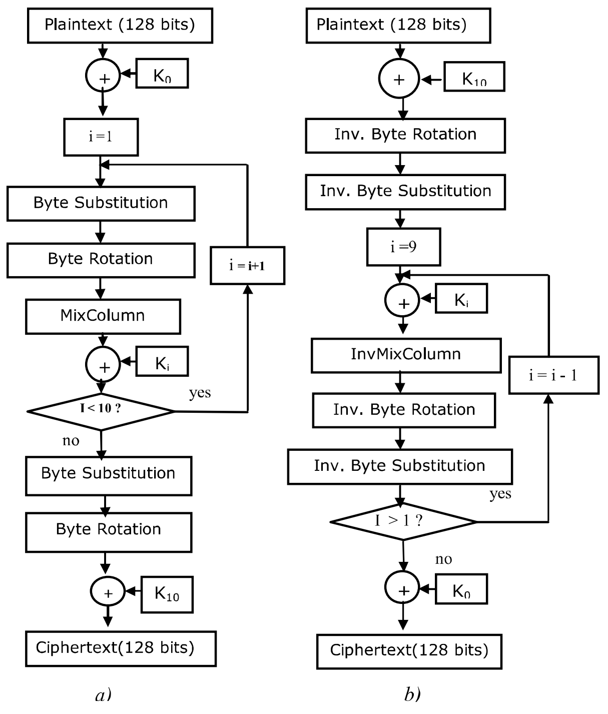

Figure 1. Encryption (a) and Decryption (b) Rijndael Diagram Blocks

The whole operation in the Rijndael algorithm is based on a matrix arrangement, called State. The Figure 2 below illustrates this structure.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$

$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

Figure 2. State Array

The encryption round of Rijndael requires realization of four operations: ByteSub, ShiftRows, MixColumns and AddRoundKey while the decryption round requires four inverse operations: InvByteSub, InvShiftRows, InvMixColumn and AddRoundKey. Each step has, therefore, its own particular function:

- ByteSub/InvByteSub: nonlinearity
- ShiftRow/InvShiftRow: inter-column diffusion
- MixColumn/InvMixColumn: inter-byte diffusion within columns
- Round key addition : Makes round function key-dependent

II.2. Key scheduling

For the AES-128, as the number of rounds are 10, it must be generated a total number of 11 round keys. The first round key is the initial key which is expanded to generate several other different round keys. The key expansion uses some shift operations, Exclusive-OR operations and the same ByteSub operation explained above (Figure 3).

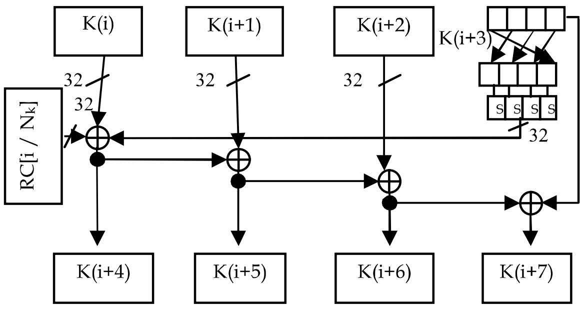

Figure 3. Encryption Key Schedule

III. FPGA IMPLEMENTATION

Rijndael is a symmetric key block cipher with a variable key size 128, 192, and 256 bits and a variable input/output block size 128, 192, and 256 bits. Our implementation is limited to the block size of 128-bits, which is the only block size required by Advanced Encryption Standard. Only one block of data is encrypted at a time, the number of clock cycles necessary to encrypt a single block of data is equal to the number of cipher rounds.

Our key scheduling unit is referred to 128-bits block size too. Synchronisation between the two blocks is triggered by the appropriate control signals.

The basic hardware concept used in this implementation, to implement one round in a single clock cycle, is the “Iterative Looping” that is defined as: “One round of hardware design, which forces the algorithm to reuse the same hardware”.

Below we describe the way of implementing all component operations of Rijndael, and then present how these basic operations are combined together to form the entire encryption/decryption unit.

III.1. The encryption and decryption units

As an example, we present in Figure 4, the encryption unit datapath.

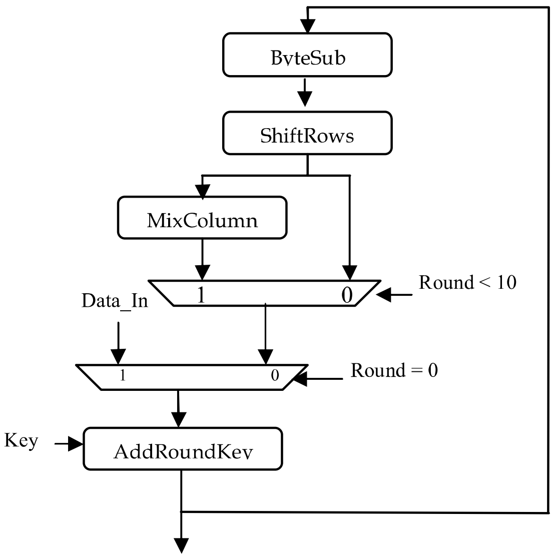

Figure 4. The Encryption Datapath

Implementation of ByteSub/ InvByteSub: The S-box is implemented using a Lookup table. In order to achieve parallelism and finish one round of encryption in one clock cycle, the same S-box is duplicated 32 times.

Implementation of InvByteSub requires only the inverse of the S-box.

Implementation of ShiftRows/ InvShiftRows: The 4 lines of the block are shifted to the left by 0, 1, 2 and 3, so we need all the columns of one block to get the resulting column. For the encoding, this is implemented as putting counter mod 4: the new write-address (indicates which block data is written in) is given by a counter plus the read-address (indicates which block data is read in) mod 4.

For the decoder: This is implemented by setting the initial values of the counter to their opposite.

Implementation of MixColumns/ InvMixColumns : The implementation of MixColumns is based on the algorithm developed by authors of the cipher. It uses the *XTime*( ) function representing multiplication by two of one byte in GF(2^8). Final structures based on this algorithm representing matrix multiplication are presented in Figure 5 for *MixColumn* function.

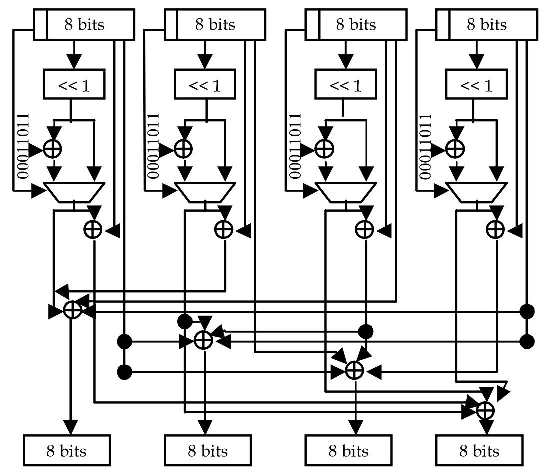

Figure 5. The MixColumn Implementation

The InvMixColumn reverse the process and act over the columns too.

Implementation of AddRoundKey / InvAddRoundKey: In the implementation of the AddRoundkey each byte is XORed with the key. Key length is 128 bits.

III.2. The key expansion

The Key Scheduling module is totally independent of the encryption module, and it also involves lookup tables and XOR operations (Figure 6).

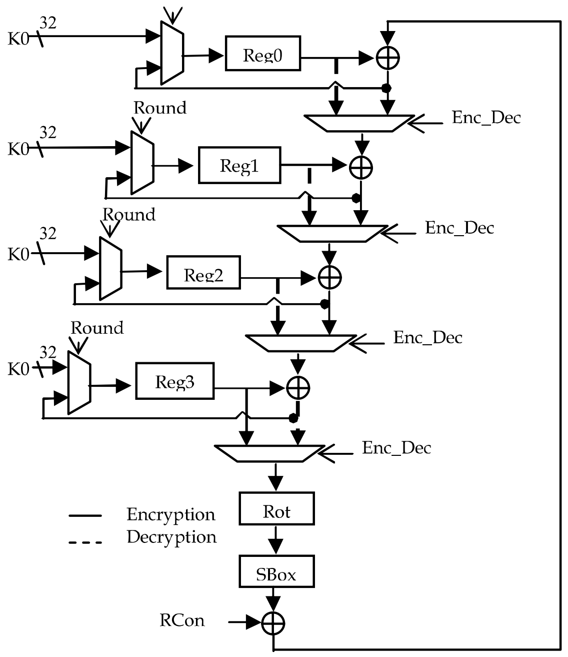

Figure 6. Datapath of the Key Expansion

IV. ENCRYPTION DATAPATH

The datapath of Rijndael is composed by a ROM for substitution, then the ShiftRows transformation which is only a routing, in addition to 4 XOR for MixColumn and a final operation XOR for AddRoundKey (Figure 7).

V. IMPLEMENTATION RESULTS

Each part of the AES encryption unit was captured in VHDL (VHSIC Hardware Description Language) and functionally simulated using ModelSim XE II. Synthesis, place and routing and timing analysis was carried out with ISE 6.2i from Xilinx. According to the synthesis result (Table 1), the complete design consumes 6065 slices (87%).

The final design accepts 128 bits of plaintext every 11 clock cycles, based on the experimentally measured clock frequency (27.91 MHz) achieving a throughput of 274, 85 Mbits/sec.

Table 1. Results of FPGA implementation

FPGA device	Spartan XC2S600E-
Input pins	260
Output pins	129
Memory (Kbits)	66720
Slices	6068
Clock (ns)	33,954
Percentage occupied (Slices)	87%
Latency (cycle)	Encryption : 13
	Decryption : 24
throughput (Mbits/s)	Encryption : 289,98
	Decryption : 157,1

VI. CONCLUSION

In this paper, we focus on implementing AES Rijndael algorithm using the reconfigurable hardware technology based on Field Programmable Gate Arrays (FPGAs). The architecture of an iterative AES processor is presented. The chip can accept data and keys of 128 bits. It can achieve a maximum throughput of 289,98 Mbits/s using 6068 Slices of Spartan XC2S600E FPGA. A standard computer laboratory was used, composed of generic station running Windows XP operational systems.

Our work supplements and extends other research efforts based on the C++ implementation of DES, IDEA, BlowFish, and Rijndael.

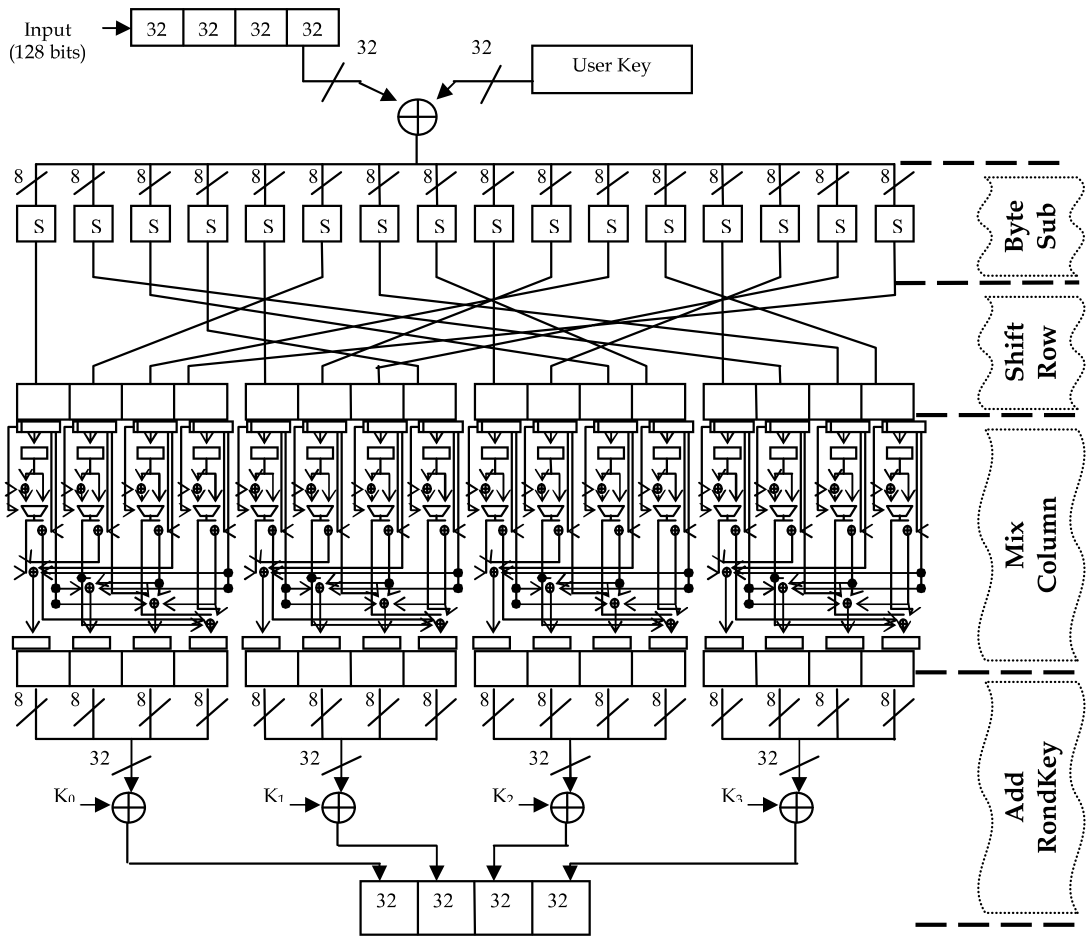

Figure 7. The final Encryption Datapath of Rijndael

ACKNOWLEDGMENTS

The authors would like to thank Xilinx Inc. for supporting this project.

REFERENCES

[1] "Advanced Encryption Standard Development Effort," http://www.nist.gov/aes.

[2] J.Nechvatal, E.Barker, L.Bassham, W.Burr, M.Dworkin, J.Foti and E.Roback, "*Report on the Development of the Advanced Encryption Standard (AES)*", NIST National Institute of standards and technology, 2 October 2000.

[3] J.Daemen and V.Rijmen, "*A Specification for Rijndael, the AES Algorithm*" First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.

[4] J. Daemen and V. Rijmen, " *The Rijndael Block Cipher* " , AES Proposal document version 2 , 1999.

[5] B.Weeks, M.Bean, T.Rozylowicz and C.Ficke, "*Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms*", National Security Agency.

[6] H.Kuo and I.Verbauwhede, "*Architectural Optimization for a 1.82Gbits/sec VLSI Implementation of the AES Rijndael Algorithm*" Electrical Engineering Department, University of California Los Angeles.

[7] Xilinx, "*Xilinx, the programmable logic data book*", Xilinx Inc, USA, 1999.