# FPGA Implementation of AES Algorithm

Ana Krkljić, Branko Dokić, and Velibor Škobić

*Abstract* – This paper deals with FPGA implementation of Advanced Encryption Standard with the key length 128 bits. The QuartusII software is used for synthesis and place and route, hile design was described using hardware description language – VHDL. while simulation of implemented design was done using ModelSim simulation software.

*Keywords* – AES, FPGA.

## I. Introduction

Two basic techniques for encrypting information are: symmetric encryption (also called secret key encryption) and asymmetric encryption (also called public key encryption). Symmetric algorithms are faster, but their main weakness is key distribution. On the other hand, asymmetric encryption overcomes key security problem, but these algorithms are generally slower. Some systems use asymmetric encryption for secure key exchange combined with symmetric algorithms for fast data encryption. One of well-respected symmetric algorithms is AES (Advanced Encryption Standard), AES is encryption standard established by the U.S. National Institute of Standards and Technology (NIST) in 2001, based on Rijndael algorithm.

The algorithm implemented in this paper is Rijndael, named after its authors Joan Daemen and Vincent Rijmen, two Belgian cryptographers. Rijndael is an iterated block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192 or 256 bits [1]. As it became a standard, called AES ( Advanced Encryption Standard), the block length was fixed to 128 bits, while the key lengths are as mentioned.

Some papers regarding hardware implementation of AES are [2], [3], [4] and [5]. Tradeoffs in hardware implementations of AES are area efficiency and speed. Throughput achieved in the above mentioned papers is 352 Mbps, 182.86 Mbps, up to 28.4Gbps and 462 Mbps respectively, for AES with 128-bit key.

## II. AES Algorithm

This paper considers AES algorithm with the block length and the key length of 128 bits. Data block is organized as 4x4 matrix of 16 bytes, considered as a state on which the following transformations are done:

- SubBytes: a non-linear byte substitution, operating on each of the state bytes independently.
- ShiftRows: the rows of the state are cyclically shifted over different offsets. row 0 is not shifted, row 1 is shifted over 1 byte, row 2 over 2 bytes and row 3 over 3 bytes.
- MixColumns: columns of the state are multiplied by a predefined matrix of constants.
- AddRoundKey: XOR operation on the round key and the result of the previous transformations

Described transformations are applied to the plaintext in 11 iterations, also called rounds. As depicted by Figure 1, initial round include only AddRoundKey, which means that plaintext is bitwise XORed to the initial round key. Following 9 rounds include all described transformations, while in the last round MixColumn is skipped.

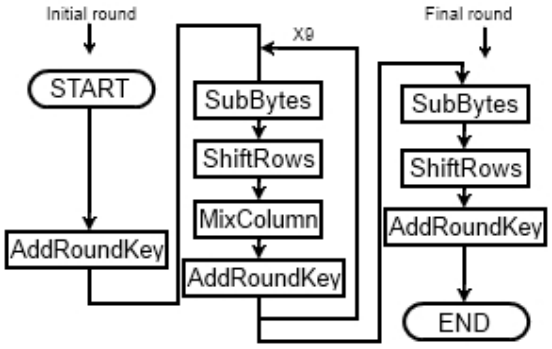


Fig. 1. AES algorithm iterations

The round key is obtained through key expansion, described by following recursive formula:

```
KeyExpansion(byte Key[16] word[44])
{
for (i=0; i<4; i++)
  W[i]=(Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3]);
 for(i=4; i<43; i++)
  {  temp=W[i-1];
     if(i%4==0)
           temp=SubByte(RotByte(temp) xor Rcon(i/4));
W[i]=W[i-4] xor temp;
} }
```

More detailed explanation of the formula is given in [1]. Key expansion makes an array of 44 columns representing 11 round keys (4 columns for each round key), for initial round as well as 10 following rounds.

## III. Implementation

Listed transformation were implemented as separate hardware entities and then connected together as depicted

by Figure 3.1. The SubBytes transformation was implemented as a look up table, hence it occupies a lot of logic resources, but provides fast encryption. Control logic module is based in a counter modulo 11, which generates a round number. That number is used in RKG module to obtain the round key. RKG means RoundKeyGeneration, which does the key expansion and returns the round key at the output. Control logic also provides appropriate connection between transformation modules. Thus in initial round, SubBytes, Shiftrows and MixColumns are skipped and *plaintext* is connected to ARK module (AddRoundKey transformation). In final round MixColumn is skipped while in the other rounds all modules are used. MEM block is used to store the value calculated in each round in order to be passed to the SubBytes input in the next round. Output of the MEM block is connected to output bus *ciphertext* only in the final round. In round 11 system does not encrypt any data, it only waits for new data to be ready at the input bus, *plaintext*. When new data is ready at the input bus, the signal at the *ready* input of control logic module is asserted. This allows synchronization with previous module that provide a 128-bit data block.

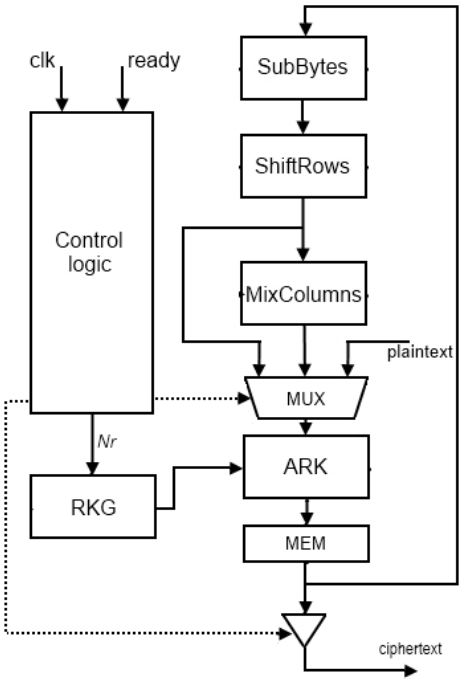


Fig. 2. Block diagram of AES encryption system

## IV. SIMULATION AND TESTING

Logic verification of the design and simulation results were obtained using ModelSim simulation software. Firstly the encryption module and the decryption module were simulated separately, using test vectors provided by NIST. These vectors, given in [1] in appendix D, are following:

```
Plaintext: 3243  F6A8 885A 308D 3131 98A2 E037 0734
Key:       2B7E  1516 28AE D2A6 ABF7 1588 09CF 4F3C
Ciphertext:3925 841D 02DC 09FB DC11 8597 196A 0B32
```

Fig. 3. The vectors provided by NIST

In this simulation, the key was fixed, without loss of generality. It is written in code, in RoundKeyGeneration as well as INVRoundKeyGeneration entity, rather than making it an input singnal. Figure 4 depicts the simulation of the encryption process. It is seen that, for given plaintext vector at the input, the ciphertext vector was obtained at the output. Similarly, the Figure 5 depicts the simulaton of the decryption process, having the *ciphertext* vector at the input and the *plaintext* at the output of the decryption module.

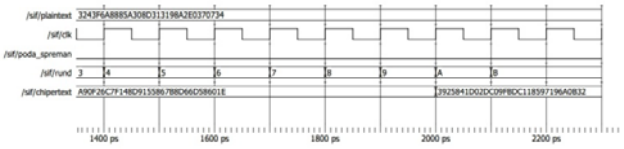

Fig. 4. Logic simulation of the encryption module

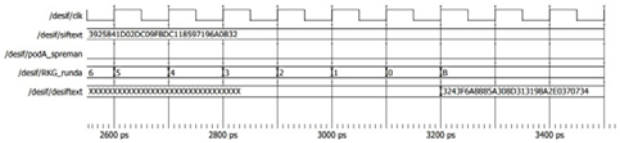

Fig. 5. Logic simulation of the decryption module

The result depicted by Figure 6 is obtained by following. Encryption and decryption module were connected via 128-bit bus and test signal in a sine wave form is applied to the input of the encryption module. Sine wave at the output also verifies the correctness of encryption and decryption.

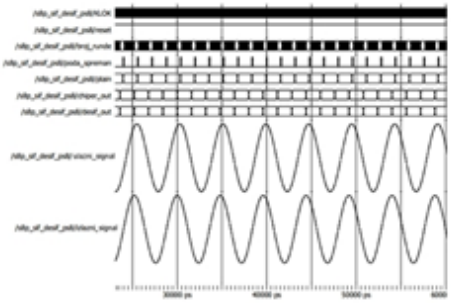

Fig. 6. Logic simulation of encryption and decryption

Implemented solution was tested on a DE1, Altera's development board. The board features Wolfson WM8731 audio CODEC with line-in, line-out and microphone-in jacks. Sine wave from signal generator is applied to the mic-in and to the oscilloscope channel 1, while oscilloscope channel 2 is connected to the line-out of the board. Input signal is digitized, then encrypted and then serialized. Serial data is transmitted via general IO pins to

the decryption module, in this case on the same FPGA device, as depicted by Figure 7.

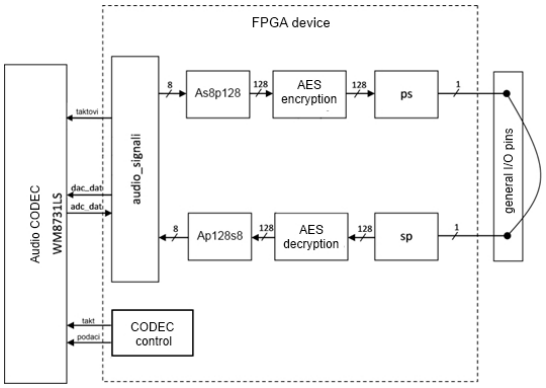


Fig. 7. Block diagram of a system tested on the DE1 board

Received serial data is parallelized, decrypted and then converted into analogue form which is observed at the line-out jack. Figure 8 depicts, both, channel 1 and channel 2 of the oscilloscope.

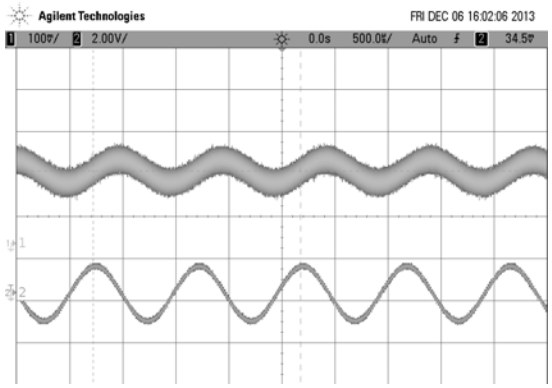


Fig. 8. Results obtained using DE1 board, signal generator and oscilloscope

Implemented system for encryption and decryption occupies 10 488 logic elements, which is 56% logic resources of the *EP2C20F484C7* device. Maximum operating frequency estimated by Quartus II is 132.68 MHz. It takes 11 clock cycles for system to generate the *ciphertext*, so the encryption throughput is as follows:

Throughput [Mbps] = (data block length[bits]) * (maximum operating frequency[MHz]) / number of clock cycles = 128*132.68/11 = 1543 [Mbps]

## IV. Conclusion

AES encryption/decryption system is implemented on *EP2C20F484C7,* device from Cyclone II FPGA series of Altera. The QuartusII software is used for hardware description, synthesis and place and route., while simulation of implemented design was done using ModelSim simulation software. System is tested on DE1, Altera's development board. Implemented solution occupies 56% logic elements of the *EP2C20F484C7* device, meaning encryption as well as decryption system on the same device. The system encrypts data at 1543 Mbps rate, for the key length 128 bits. Future development would include an effort to reduce the logic resources utilization, using, for example, embedded memory blocks.

## Acknowledgement

This paper is result of undergraduate thesis *"New technologies and families of programmable logic devices",* defended by Ana Krkljić, at Faculty of Electrical Engineering, University of Banjaluka, december 2013

## References

[1] Joan Daemen and Vincent Rijmen, The Rijndael Block Cipher, Retrieved November 01, 2013, from http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf

[2] Ghewari, P., Patil, J., Chougule, A., *"Efficient Hardware Design and Implementation of AES Cryptosystem"*, International Journal of Engineering Science and Technology, Vol. 2, 2010, pp. 213-219.

[3] Mali, M., Novak, F., Biasizzo A., *"Hardware Implementation of AES algorithm"*, Journal of Electrical Engineering, Vol. 56, No. 9-10, 2005, pp. 265-269.

[4] C.P. Fan *J. K.* Hwang, *"FPGA Implementations of High Throughput Sequentialand Fully Pileplined AES Algorithm ",* International Journal of Electrical Engineering, Vol.15, No.6 PP. 447-455 (2008)

[5] Adib, S., Raissouni, N., *"AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparasion of 128, 192 and 256 bits KEY"*, International Journal of Reconfigurable and Embedded Systems, Vol. 1, No. 2, July 2012, pp. 67-74.