International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Research Article

# Implementation of Advanced Encryption Standard (AES) Algorithm Based on FPGA

Ashwini R. Tonde[Á*] and Akshay P. Dhande[Á]

[Á]P. R. Patil College of Engg. &Tech.Maharashtra, India

Accepted 10 April 2014, Available online 25 April 2014, **Vol.4, No.2 (April 2014)**

***Abstract***

*The importance of cryptography applied to security in electronic data transactions has acquired an essential relevance during the last few years. A proposed FPGA-based implementation of the Advanced Encryption Standard (AES) algorithm is presented in this paper. The design has been coded by Very high speed integrated circuit Hardware Descriptive Language. All the results are synthesized and simulated using Xilinx ISE and ModelSim software respectively. This implementation is compared with other works to show the efficiency. The design uses an iterative looping approach with block and key size of 128 bits, lookup table implementation of S-box. This gives low complexity architecture and easily achieves low latency as well as high throughput. Simulation results, performance results are presented and compared with previous reported designs.*

***Keywords****: AES, FPGA, VHDL, encryption, decryption and block cipher.*

## 1. Introduction

Everyday millions of users generate and interchange large volumes of information in various fields such as medical reports, and bank services via Internet. All these applications deserve a special treatment from the security point of view, not only in the transport of such information. Here, cryptography techniques are especially applicable. DES considered being insecure for many applications. This is due to the 56-bit key size being too small, in January, 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes. And this is the reason, why the National Institute of Standards and Technology (NIST) opened a formal call for algorithms in September 1997. So a group of fifteen AES candidate algorithms were announced in August 1998.

Five algorithms were selected by NIST: Mars, RC6, Rijndael, Serpent and Twofish as the final competitors in August 2000. These algorithms were subject to further analysis prior to the selection of the best algorithm for the AES. Next, on October 2, 2000, NIST announced that the Rijndael algorithm was the winner. Rijndael can be specified with key and block sizes in any multiple of 32, with a minimum of 128 bits and a maximum of 256 bits. Therefore, the problem of breaking the key becomes more difficult. In cryptography, the AES is also known as Rijndael. AES has a fixed block size of 128 bits and a key size of 128, 192 or 256 bits. The AES algorithm can be efficiently implemented by hardware and software.

## 2. Literature review

The design uses an iterative looping approach with block and key size of 128 bits (Hoang Trang and Nguyen Van, 2012). The design has been coded by Verilog HDL. All the results are synthesized and simulated basing on the Quatus 9.0, the Model Sim. So the latency of encryption is 51 clock cycles on Xilinx platform. Similarly, the latency of decryption is 51 clock cycles. The algorithm achieves a low latency and the throughput reaches the value of 1054Mbit/sec for encryption and 615 Mbit/sec for decryption.

An implementation of high speed AES algorithm based on FPGA is presented, in order to improve the safety of data in transmission (WANG Wei, CHEN Jie & XU Fei, 2012). Mathematic principle, encryption process and logic structure of AES algorithm are introduced. So as to reach the purpose of improving the system computing speed, the pipelining and parallel processing methods were used. The simulation results show that the high-speed AES encryption algorithm implemented correctly. Design was tested on Xilinx Virtex-5 FPGA. All the processes including Implementation and Simulation are finished in Model Sim ISE 13.3 development platform. Results show that the system could complete the whole process correctly in a 200MHz clock rate.

This system aims at reduced hardware structure (Yang Jun Ding Jun Li Na Guo Yixiong, 2010). And this system has high security and reliability. The advantage of this design is the fact that we do not need to store the round key since they are currently calculated in accordance that AES algorithm is used in the low requirements of the terminal throughput at present, the high safety and cost-

*Corresponding author: **Ashwini R. Tonde**

effective reduced AES system is designed and validated on the Altera Cyclone, aiming at reduced hardware structure. Furthermore, this system can be widely used in the terminal equipments which less demand on the throughput. Throughput found for encryption and decryption process is 593.45Mbps and 267.63Mbps respectively.

## 3. AES Algorithm

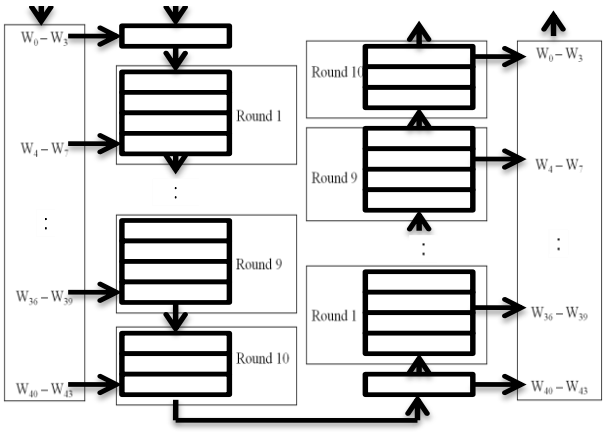


**Fig.1.** Encryption and decryption process of AES algorithm

1. AES encryption

The AES algorithm operates on a 128-bit block of data. The key length is 128, 192 or 256 bits in length respectively. The pre-rpound and last rounds differ from other rounds, there is an AddRoundKey transformation in pre-round and no MixCoulmns transformation is performed in the last round as shown in fig. 1. In this paper, we use the key length of 128 bits as a model for general explanation.

### *1.1 SubBytes Transformation*

The SubBytes transformation includes non-linear byte substitution, operating on each of the state bytes independently. This is done by using a once-precalculated substitution table called S-box. S-box table contains 256 numbers (from 0 to 255) and their corresponding resulting values. Advantage of performing the S-box computation in a single clock cycle, reducing the latency and avoids complexity of hardware implementation.

### *1.2 ShiftRows Transformation*

ShiftRows transformation includes, the rows of the state are cyclically left shifted. Row 0 remain unchange; row 1 does shift of one byte to the left; row 2 does shift of two bytes to the left and row 3 does shift of three bytes to the left.

### *1.3 MixColumns Transformation*

MixColumns transformation includes, the columns of the state are considered as polynomials over GF ($2^8$) and multiplied by modulo $x^4 + 1$ with a fixed polynomial c(x), given by: $c(x)=\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.

### *1.4 AddRoundKey Transformation*

AddRoundKey transformation includes, a Round Key is added to the State - resulted from the operation of the MixColumns transformation - by a simple bitwise XOR operation. The RoundKey for each round is derived from the main key using the KeyExpansion algorithm. The encryption/ decryption algorithm needs eleven 128-bit RoundKey, which are denoted by RoundKey[0] to RoundKey[10].

2. *AES decryption*

Reverse of encryption which inverses round transformations to compute the original plaintext from cipher-text in reverse order called as decryption. The rounds of transformation of decryption use the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes successively as shown in fig. 1.

### *2.1 AddRoundKey*

AddRoundKey is its own inverse function because the XOR function is its own inverse. Here ciphertext state XOR with rounkey . The round keys obtained from key expansion algorithm selected in reverse order.

### *2.2 InvShiftRows Transformation*

InvShiftRows functions in the same way as the ShiftRows, only in the opposite direction. The first row is not shifted, while the second, third and fourth rows gets shifted to right by one, two and three bytes respectively.

### *2.3 InvSubBytes transformation*

From once-precalculated substitution table called InvS-box, InvSubBytes transformation is done. InvS-box table contains 256 numbers (from 0 to 255) and their corresponding values.

### *2.4 InvMixColumns Transformation*

InvMixColumns transformation includes, ,polynomials of degree less than 4 over GF ($2^8$) which coefficients are the elements in the columns of the state, are multiplied modulo ($x^4 + 1$) by a fixed polynomial $d(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$, where {0B}, {0D}; {09}, {0E} denote hexadecimal values.

## 3. Key Expansion Process

The AES algorithm takes the Cipher Key, 'K', and performs a Key Expansion process to generate keys shown in fig. 2. SubWord() is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. Function RotWord() takes a word $[a_0,a_1,a_2,a_3]$ as input, after performing a cyclic

permutation returns the word $[a_1,a_2,a_3,a_0]$. Rcon[i], the round constant word array, contains the values given by $[x^{i-1},\{00\},\{00\},\{00\}]$.

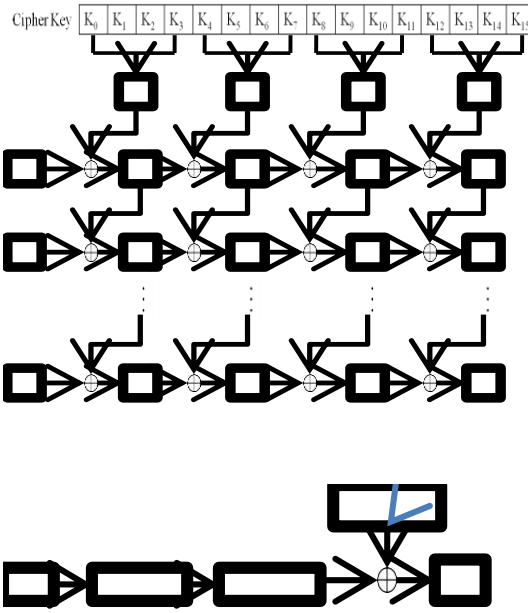


**Fig. 2.** Key Expansion Process

## 4. Implementation and Results

The design of AES algorithm is coded using VHDL language. Simulation and synthesis of AES algorithm is done on ModelSim software and Xilinx ISE software respectively. We implemented the AES Encryption/Decryption module on a Xilinx XC3S500E Spartan-3E FPGA kit.

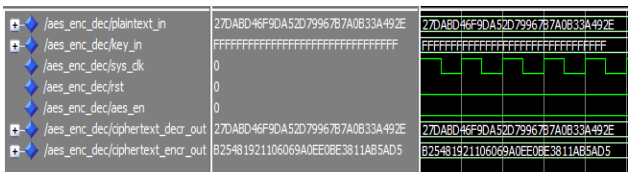


**Fig. 3** Simulation result of AES encryption and decryption design

| Device Utilization Summary (estimated values) | | | [-] |
|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** |
| Number of Slices | 3699 | 960 | 385% |
| Number of Slice Flip Flops | 2338 | 1920 | 121% |
| Number of 4 input LUTs | 6436 | 1920 | 335% |
| Number of bonded IOBs | 515 | 66 | 780% |
| Number of BRAMs | 2 | 4 | 50% |
| Number of GCLKs | 5 | 24 | 20% |

**Fig. 4** Synthesis result of AES design

In synthesis report number of logic utilised is shown in Fig. 3. Number of bonded IOBs used are 515 but available IOBs are 66, due to this reason encryption and decryption design are implemented separately on Xilinx XC3S500E Spartan-3E FPGA kit. From synthesis report we calculate parameters like throughput, fmax and delay, for encryption and decryption design separately, and finally all parameters are again calculated for whole AES algorithm design as shown in table 1.

The test results show that the system could complete the whole process correctly in a 230.92MHz clock rate. Under the simulation clock 50MHZ, the simulation waveform is shown in Fig.2 and in Fig.3, in which plaintext_in and key_in are applied as input. When rst signal is '0' and sys_clk is applied and aes_en is '1', we will get encrypted data called as ciphertext after some delay. And when we will change sttus of signal aes_in to '0' we will get decrypted date back called as plaintext.

**Table 1.** Comparison in implementation of AES algorithm result

| Designs | Our design (Xilinx) | | (Altera) |
|---|---|---|---|
| Throughput (Mbps) | Encryption | 672.52 | 593.45 |
| | Decryption | 603.59 | 267.63 |

## Conclusion

Efficient implementation of AES algorithm is presented in this paper. High throughput is achieved in this design. Results are compared with previous reported designs result to show efficiency. Simulation of AES algorithm is done on ModelSim software and implemented on Xilinx XC3S500E Spartan-3E FPGA kit.

## References

Hoang Trang and Nguyen Van (2012), An efficient FPGA implementation of the Advanced Encryption Standard algorithm *IEEE* 978-1-4673-0309-5/12.

WANG Wei, CHEN Jie & XU Fei (2012), An Implementation of AES Algorithm Based on FPGA, *IEEE* 978-1-4673-0024-7/10.

Yang Jun Ding Jun Li Na Guo Yixiong (2010), FPGA based design and implementation of reduced AES algorithm, *IEEE* 978-0-7695-3972-0/10.

Adam J. Elbirt, W. Yip, B. Chetwynd, and C. Paar- (2001), An FPGA-Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists, *IEEE* 1063–8210/01

Nalini C, Nagaraj, Dr. Anandmohan P.V, & Poornaiah D.V, V.D.kulkarni (2006), An FPGA Based Performance Analysis Pipelining and Unrolling of AES Algorithm, *IEEE* 1-4244-0716-8/06.

Ahmad, N. Hasan, R., Jubadi, W.M. (2010), Design of AES S-Box using combinational logic optimization, *IEEE Symposium on Industrial Electronics & Applications (ISIEA)*, pp. 696-699,

Alex Panato, Marcelo Barcelos, Ricardo Reis, (2002) An IP of an Advanced Encryption Standard for Altera Devices, *SBCCI* pp. 197-202.