FPGA Implementation of Efficient Hardware for the Advanced Encryption Standard

Amandeep Kaur, Puneet Bhardwaj, Naveen Kumar

***Abstract**: We present an efficient hardware architecture design & implementation of Advanced Encryption Standard (AES) – Rijndael cryptosystem. The AES algorithm defined by the National Institute of Standard and Technology (NIST) of United States has been widely accepted. All the cryptographic algorithms developed can be implemented with software or built with pure hardware. However with the help of Field Programmable Gate Arrays (FPGA) we tend to find expeditious solution and which can be easily upgraded to integrateany concordat changes. This contribution investigates the AES encryption and decryption cryptosystem with regard to FPGA and Very High Speed Integrated Circuit Hardware Description language (VHDL). Optimized and Synthesizable VHDL code is developed for the implementation of both 128-bit data encryption and decryption process. Xilinx ISE 10.1 software is used for simulation. Each program is tested with some of the sample vectors provided by NIST and output results are perfect with minimal delay. The synthesis results found from FPGA implementation by Xilinx Synthesis Tool on Virtex II pro kit shows that the computation time for generating the ciphertext by AES with 4 sbox and 2 dual port RAM is 6.922 ns.*

***Index Terms**: Cryptography, Advanced Encryption Standard, Rijndael, S-box, key expansion, cipher text.*

I. INTRODUCTION

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication[1]. Cryptography is not the only means of providing information security, but rather one set of techniques. A fundamental goal of cryptography is to adequately address these four areas in both theory and practice. Cryptography is about the prevention and detection of cheating and other malicious activities. Symmetric-key cryptography, also called secret key cryptography, is the most intuitive kind of cryptography. It involves the use of a secret key known only to the participants of the secure communication. It is characterized by the use of a single key to perform both the encrypting and decrypting of data.

The United States standard for Symmetric-key cryptography, in which the same key is used for both encryption and decryption, is the Data Encryption Standard (DES) [2]. This is based upon a combination and permutation of shifts and exclusive OR operations and so can be very fast when implemented directly on hardware (1 GByte/s throughput or better) or on general-purpose processors. DES uses a 56-bit key and maps a 64-bit input block of plaintext onto a 64-bit output block of ciphertext. 56 bits is a rather small key for today's computing power, the key size is indeed one of the most controversial aspects of this algorithm. The mainstream cryptographic community has long held that DES's 56-bit key is too short to withstand a brute-force attack from modern computers [3]. The current key size of 56 bits (plus 8 parity bits) of DES is now starting to seem small, but the use of larger keys with triple DES (3DES) can generate much greater security. If security is the only consideration, then 3DES will be an appropriate choice for a standardized encryption algorithm for decades to come. However, the principle drawback of 3DES is that the algorithm is relatively sluggish in software. The original DES was designed for mid 1970s hardware implementation and does not produce efficient software code. 3DES, which has three times as many rounds as DES, is correspondingly slower. A secondary drawback is that both DES and 3DES use a 64- bit block size. For reasons of both efficiency and security, a larger block size is desirable. Because of these drawbacks, 3DES is not a reasonable candidate for long-term use[8].

As a replacement, NIST (National Institute of Standards and Technology) of USA in 1997 issued a call for proposals for a new Advanced Encryption Standard (AES), which would have security length equal to or better than 3DES and significantly, improved efficiency. In addition to these general requirements, NIST specified that AES must be a symmetric block cipher with ablock length of 128 bits and support for key lengths of 128, 192, and 256 bits. The AES algorithm was selected in October 2001 after a multi-year evaluation process led by NIST with submissions and review by an international community of cryptography experts and the Rijndael algorithm [4], invented by Joan Daemen and Vincent Rijmen, was selected as the standard, which was published in November 2002. NIST's intent was to have a cipher that will remain secure well into the next century.AES uses a substitution-permutation network in a more general sense. Each round of processing in AES involves byte-level substitutions followed by word-level per- mutations. Speaking generally, DES also involves substitutions and permutations, except that the permutations are based on the Feistel notion of dividing the input block into two halves, processing each half separately, and then swapping the two halves. The nature of substitutions and permutations in AES allows for a fast software implementation of the algorithm.

Manuscript published on 28 February 2013.

***Correspondence Author(s)**
Amandeep Kaur, electronics and communication, Thapar University, Patiala, Punjab, India.
Puneet Bhardwaj, electronics and communication, Thapar University, Patiala, Punjab, India.
Naveen Kumar, electronics and communication, Thapar University, Patiala, Punjab, India.

II. AES ALGORITHM

In AESinput to the encryption and decryption algorithms is a single 128-bit block. This block of input is depicted as a square matrix of bytes. This block is copied into the state array, which is modified at each stage of encryption or decryption. After the final stage, state is copied to an output matrix. These operations are depicted in Fig.1. Similarly, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and total key schedule is 44 words for the 128-bit key. The ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the in matrix, the second four bytes occupy the second column and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the w matrix. It wasbasically designed to have the following characteristics:

– Resistance against all known attacks.
– Speed and code compactness on a wide range ofplatforms.
– Design Simplicity.

A. *AES algorithm Process*

The encryption and decryption process consist of a number of different transformation applied consecutively over the data block bits, in affixed number of iteration, called rounds. The number of rounds depends on the length of the key used for the encryption process. For key length of 128 bits, the number of iteration required are 1o that is N_r= 10.Each of the first N_r-1 rounds consists of 4 transformations:
SubBytes(), ShiftRows(), MixColumns() &AddRoundKey(). The four different transformations are given in detail below.

Sub Bytes Transformation: It is a non-linear replacement of bytes that operates autonomously on each byte of the State employing a substitution table (S box). This S-box which is invertible is constructed by first taking the multiplicative inverse in the finite field GF (2^8) with irreducible polynomial m(x) = $x^8 + x^4 + x^3 + x + 1$. The element {00} is mapped to itself. Then affine transformation is applied (over GF (2)).

Shift Rows Transformation: Cyclically move the rows of the State over unlike offsets. The operation is equally the similar in the decryption process except at the point that the shifting offsets have dissimilar values.

Mix Columns Transformation: This transformation operates on the State column-by-column, considering each column as a four-term polynomial. The columns are taken as polynomials over GF (2^8) and multiplied by modulo $x^4 + 1$ with a fixed polynomial a(x) = {03} x^3+ {01} x^2+ {02} x.

Add Round Key Transformation: In this transformation is having Round Key which is added to the State by a simply XORing operation. Every Round Key contain of N_b words from the key expansion. Those N_b words are added into the columns of the State. Key Addition is the same for the decryption process.

Key Expansion: Each round key is a 4-word (128-bit) array induced as a product of the previous round key, a constant that changes each round, and a series of S-Box lookups for each 32-bit word of the key. The Key schedule Expansion results in a total of N_b $(N_r + 1)$ words [5].

The decryption process is exact inverse of the encryption process. All the transformations implied in encryption process are opposite to encryption process. That means the last round values of both the data and key are first round inputs for the decryption process and follows in decreasing order.

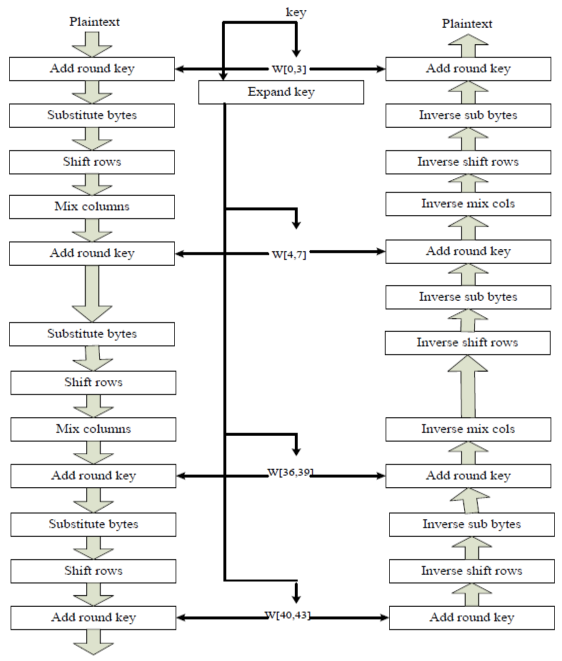


Fig. 1AES encryption and decryption

III. EFFICIENT IMPLEMENTATION OF AES

The AES design is developed and tested adhering to the Xilinx ISE design flow. The tools primarily used are the Xilinx ISE and ModelSim for simulation, synthesis and implementation. Fig. 1 shows the design flow implemented. This shows testing at various stages of the design. Initially the Behavioral testing once the Design entry or the Coding part is done, then the Post place and route simulation and after the circuit is implemented the On-chip verification.For the AES design, a software model of the AES algorithm was initially developed in VHDL that would read a binary text file and then output the encoded bit stream into another binary text file. Once the results from the VHDL code matcheswith the results in the behavioral simulation the design is further synthesized and checked for behavioral simulation again. The Design is then implemented in a FPGA.

A. *Design Entry*

The design entry for this project is basically the VHDL codes for the AES. The top-level module for the design is called aes_core.vhd.
The design is further divided into 5 parts.

1. The AES_fsm_encrypt.vhd
2. The key_expansion.vhd
3. The sbox.vhd
4. The addkey.vhd
5. The mix_col.vhd

The algorithmic core is divided into two separate data paths one for encryption and a second for decryption operation. The two data paths are independent, however they share the key_expansion component, which provides decrypt, and encrypt keys (which are the same only in opposite order).

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Each data path is controlled by its own FSM. If configured by the generic DECRYPTION the decryption data path is included and some multiplexers are generated for the shared signals, e.g. result or roundkey_index. For example the encryption data path of aes_core.vhd is given in Fig.2.The keyexpansion component computes one column of a roundkey in two clock cycles. In the first cycle the column is substituted through the s-box, in the second cycle the shift-operation is executed. The AES core computes one iteration (round) of the Rijndael-Algorithm each clock cycle, thus a 128 Bit data block is encrypted or decrypted in 10 cycles plus an initial round.

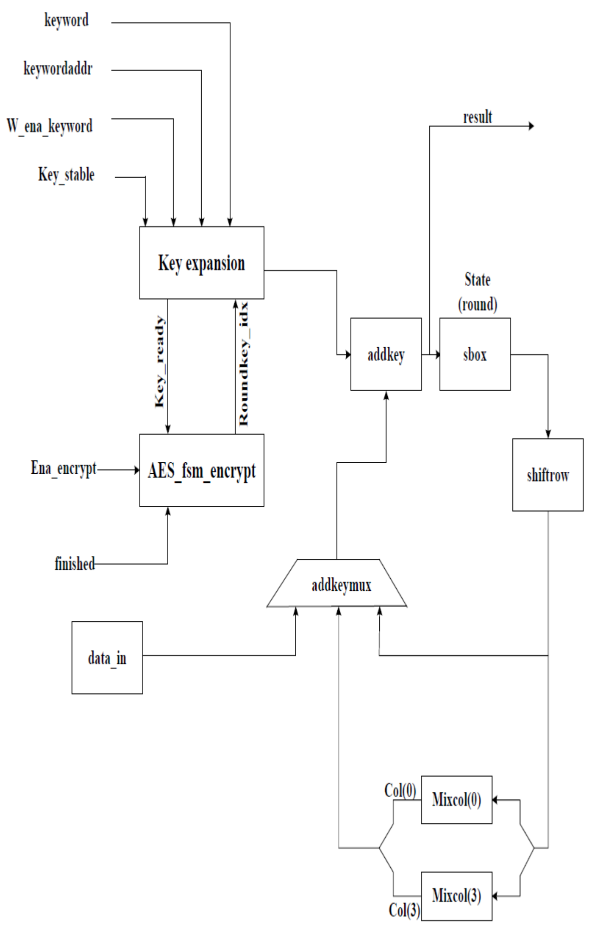


Fig.2 Block diagram of encrypt data path of the AES core as implemented inaes_core.vhd.

B. Implementation

VHDL is very high-speed hardware description language because it offers adjustability to exchange among environments. The software used for this work is Xilinx ISE This is used for writing, debugging and optimizing efforts, and also for fitting, simulating and checking the performance results using the Xilinx xst simulation tools available on Web pack design software. An iterative method of design is implemented to minimize the hardware utilization.

The encryption process is shown in Fig. 1. In order to allow a full parallel process of the state, it is necessary to implement all the transformations over 128 bits. The most expensive one is the Byte substitution, because it is a table lookup operation.Fig. 3 depicts the AES core block as seen in the Xilinx ISE.Table 1 shows Synthesis ResultsofAES core program.

The decryption implementation results are similar to the encryption implementation. The key expansion module is modified in the reverse order. In which last round key is treated as the first round and decreasing order follows.

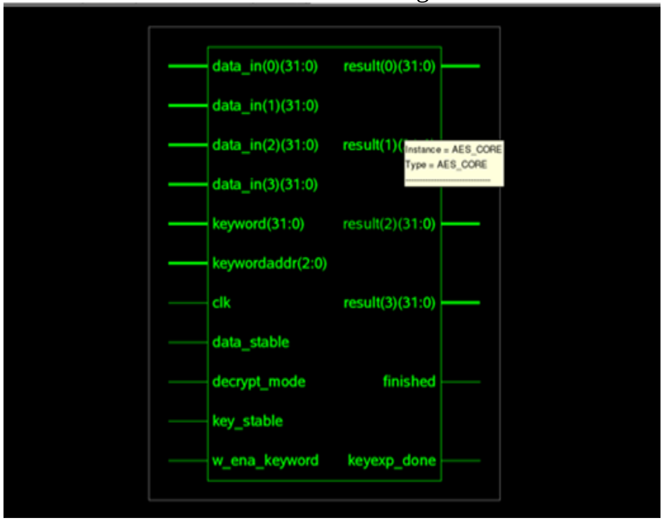


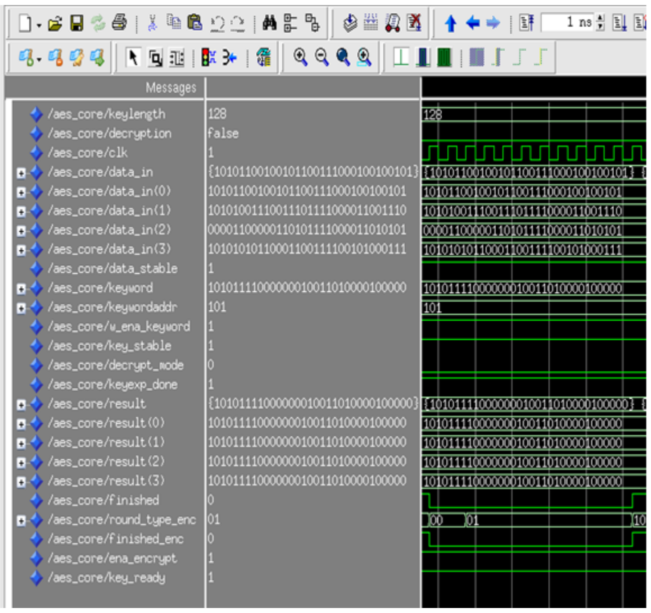
Fig.3: AES core block

IV. EXPERIMENTAL RESULTS

All the results are based on simulations from the Xilinx ISE 9.2i tools, using Test Bench Waveform Generator. All the individual transformation of both encryption and decryption are simulated usingSpartan – 3E (XC3S500) FPGA kit is used for implementation of encryption algorithms [7].

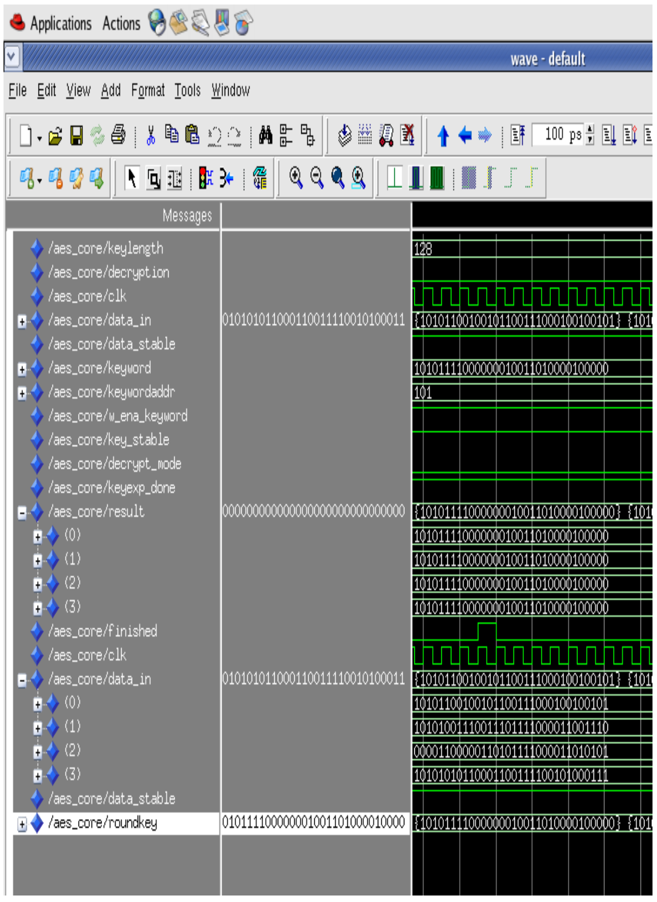
A. Simulation Results

In fig 4, 32-bit data is given four times and 4-bit address; so that it gives 128-bit data gets encrypted after processing through the whole encryption process including 10 rounds. The simulation results are shown in fig. 4 and fig. 5. Table 1 and Table 2 gives the description of various input output signals used in the entity of AES algorithm.Table 3 AES algorithm various performance measures.



```
Messages
/aes_core/keylength        128
/aes_core/decryption       false
/aes_core/clk              1
/aes_core/data_in          {10101100100101100111000100100101}
/aes_core/data_in(0)       10101100100101100111000100100101
/aes_core/data_in(1)       10101001110011101111000011001110
/aes_core/data_in(2)       00001100000110101111000011010101
/aes_core/data_in(3)       10101010110001100111100101000111
/aes_core/data_stable      1
/aes_core/keyword          10101111000000010011010000100000
/aes_core/keywordaddr      101
/aes_core/w_ena_keyword    1
/aes_core/key_stable       1
/aes_core/decrypt_mode     0
/aes_core/keyexp_done      1
/aes_core/result           {10101111000000010011010000100000}
/aes_core/result(0)        10101111000000010011010000100000
/aes_core/result(1)        10101111000000010011010000100000
/aes_core/result(2)        10101111000000010011010000100000
/aes_core/result(3)        10101111000000010011010000100000
/aes_core/finished         0
/aes_core/round_type_enc   01
/aes_core/finished_enc     0
/aes_core/ena_encrypt      1
/aes_core/key_ready        1
```

Fig. 4 Modelsim result of AES core block



```
/aes_core/keylength                                              128
/aes_core/decryption
/aes_core/clk
/aes_core/data_in        010101011000110011110010100011         {10101100100101100111000100100101} {101
/aes_core/data_stable
/aes_core/keyword                                                101011110000000010011010000100000
/aes_core/keywordaddr                                            101
/aes_core/w_ena_keyword
/aes_core/key_stable
/aes_core/decrypt_mode
/aes_core/keyexp_done
/aes_core/result         0000000000000000000000000000000         {101011110000000010011010000100000} {101
  (0)                                                            101011110000000010011010000100000
  (1)                                                            101011110000000010011010000100000
  (2)                                                            101011110000000010011010000100000
  (3)                                                            101011110000000010011010000100000
/aes_core/finished
/aes_core/clk
/aes_core/data_in        010101011000110011110010100011         {10101100100101100111000100100101} {101
  (0)                                                            10101100100101100111000100100101
  (1)                                                            10101001110011101111000011001110
  (2)                                                            00001100000110101111000011010101
  (3)                                                            10101010110001100111100101000111
/aes_core/data_stable
/aes_core/roundkey       0101111000000001001101000010000         {101011110000000010011010000100000} {101
```

Fig. 5 Modelsim Final results after addition of round key

Table 1Values of signals after encryption

S.No.	Signal	Value
1.	keylength	128
2.	decryption	False
3.1	data_in(0)	10101100100101100111000100100101
3.2	data_in(1)	10101001110011101111000011001110
3.3	data_in(2)	00001100000110101111000011010101
3.4	data_in(3)	10101010110001100111100101000111
4	data_stable	1
5.	keyword	101011110000000010011010000100000
6.	keywordaddr	101
7.	w_ena_keyword	1
8.	key_stable	1
9.	decrypt_mode	0
10.	keyexp_done	1
11.1	result(0)	101011110000000010011010000100000
11.2	result(1)	10101111000010010011010000100000
11.3	result(2)	1010111111000000010011010000100000
11.4	result(3)	101011110000000010011010000100000
12.	finished	1
13.	round_type_enc	10
14.	finished_enc	1
15.	ena_encrypt	1
16.	key_ready	1
17.	ready	1

Table 2 shows Synthesis ResultsofAES core program

S.No	Parameter	Value
1.	FPGA Device Package	xc2vp30-5-ff896
2.	Number of slices	1,127 out of 13,696
3.	Number of slice Flip Flops	459 out of 27,392
4.	Number of 4 input LUTs	2,029 out of 27,392
5.	Number of IOB flip flops	33
6.	Number of bonded IOBs	75 out of 416
7.	Minimum period	4.043 ns
8.	Maximum frequency	247.365 MHz
9.	Power(X-power)	399mW
10.	Slack(Setup)	4.866 ns
11.	Slack(Hold)	6.161 ns
12.	Number of GCLKs	1 out of 16

Table 3 AES algorithm various performance measures

S.No.	Parameter	AES ARCHITECTURE 4 sbox & 2 dual port RAM
1.	**Slice LUTs**	335
2.	**Delay**	6.922 ns
3.	**Power**	124 mW

V. CONCLUSION

The Advanced Encryption Standard-Rijndael algorithm is an iterative private key symmetric block cipher which process data blocks of 128 bits through the use of cipher keys with lengths of 128, 192, and 256 bits. An efficient FPGA implementation of 128-bit block and 128 bit key AES-Rinjdael cryptosystem has been presented in this paper. Optimized and Synthesizable VHDL code is developed for the implementation of both 128 bit data encryption and decryption process & description is verified using ISE 9.2i functional simulator from Xillinx. All the transformations of algorithm are simulated using an iterative design approach in order to minimize the hardware consumption. Each program is tested with some of the sample vectors provided by NIST.

REFERENCES

1. Menezes, A. and Vanstone, S. “Handbook of Applied Cryptography”, CRC Press, Inc. 1996.
2. National Bureau of Standards, NBS FIPS PUB 46, “Data Encryption Standard”, U.S. Department of Commerce, January 1977
3. Daemon, J., and Rijmen, V. “Rijndael: The Advanced Encryption Standard.”, Dr. Dobb’s Journal, 3, March 2001, 137-139.
4. B. Smith, “An approach to graphs of linear forms (Unpublished work style),” unpublished.
5. Daemon, J., and Rijmen, V. “The Design of Rijndael: The Wide Trail Strategy Explained.” New York, Springer – Verlag, 2000.
6. I. M. Verbauwhede, P.R. Schaumont, and, H. Kuo, "Design and Performance Testing of a 2.29 Gb/s Rijndael Processor,"
7. IEEE J. of Solid State-Circuit, Vol.38, No. 3, March 2003, pp. 569 – 572.
8. K. Gaj and P. Chodowiec, Comparison of the hardware performance of the AES candidates using reconfigurable hardware, inThe Third AES Candidates Conference, printed by the National Institute of Standards and Technology.
9. J. Wang, “Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication),” *IEEE J. Quantum Electron.*, submitted for publication.
10. A. J. Elbirt, W. Yip, B. Chetwynd, C. Paar, "An FPGA implementation and performance evaluation of the AES block cipher
11. candidate algorithm finalists," Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference.