Journal of Engineering and Applied Sciences 13 (Special Issue 13): 10564-10570, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

A VHDL Implementation Encryption and Decryption of the Advanced Encryption Standard (AES) Algorithm

Nasseer M. Basheer and Enas Ali Ahmed
Department of Computer Technology Engineering, Technical Engineering College of Mosul, Mosul, Iraq

Abstract: The implementation of Advanced Encryption Standard (AES) algorithm for encryption and decryption using a Finite-State Machine (FSM) is proposed in this study. The programming language used is the VHDL (Very High speed integrated circuit hardware Description Language). Xilinx_ISE_10.1 Software is used to simulate the synthesizable VHDL code. The obtained results are being synthesized and simulated through using Xilinx ISE 10.1 Software. A Finite State Machine (FSM) approach with block and key size of 128 bits is used in this design. Other previous reported works are also mentioned. A throughput of 1.349 Gbps for encryption process and 1.012 Gbps for decryption process are resulted by the implementation of the algorithm using (FSM) approach. Depending on the synthesis report the number of slices is 2156 (46% of total resources) for encryption and 2354 (50% of total resoures) for decryption. VHDL test bench waveform of Xilinx ISE10.1 is used to test the work of the implemented algorithm. The hardware kit used is the Spartan3e XC3s500e FPGA.

Key words: AES algorithm, encryption, decryption, FSM, FPGA ,VHDL

INTRODUCTION

The urgent need for automated devices to protect files and other information is an increasing matter, particularly with the growing data in computer networks and communication technologies. This data needs to be kept secure and protected from many attacks such as automated spying or hacking. Everyday, users interchange large amount of information in different fields such as medical reports and bank services using the Internet and these applications should be put under severe security. Therefore, cryptography has become a vital issue. The main goal of using cryptographic algorithms is for security purposes. From the last decade, Data Encryption Standard (DES) was usedin cryptography. DES is replaced by the rijndael algorithm because of its short key length. The Rijndael algorithm, however has become a standard in the cryptographic field, called AES. As a matter of fact, Data Encryption Standard (DES) is considered insecure for many applications because of the small size of 56-bit key. In January, 1999 distributed net and the Electronic Frontier Foundation collaborated (EFF) announced breaking the DES key in 22 h and 15 min. Therefore, in September 1997 the National Institute of Standards and Technology (NIST) expressed their need for algorithms. As a result, a group of fifteen algorithms were declared in August 1998 (Tonde and Dhande, 2014; Ankita *et al*., 2017; Gurpinder and Singh, 2014). Five algorithms were chosen by NIST: RC6, Mars, Serpent, Rijndael and Twofish. These algorithms were the final competitors chosen in August 2000. On October 2000, NIST declared that the Rijndael algorithm was selected for this purpose (Tonde and Dhande, 2014).

The Rijndael algorithm is also known as AES was developed by two Belgian scientists Vincent Rijmen and Joan Daemen. Rijndael is specified by having key and block sizes in any multiple of 32 bit with 128 bits as a minimum and 256 bits as a maximum of. As a result, breaking the key has become more complicated. The rijndael algorithm is a symmetric block cipher which processes data blocks of 128 bits using encryption keys with 128, 192 or 256 bits. Among the reasons why the Rijndael algorithm was chosen was its ability to deal with additional block sizes and key lengths (Kosaraju, 2003). Also beacuse it recorded the best results in terms of performance, flexibility, efficiency, implementation and security (Manteena, 2004; Tibrewal and Kumar, 2014).

Literature review: Singh and Dod (2016) proposed an effective hardware architecture design and implementation of AES. In this design, a proposed FPGA-based implementation of the AES algorithm is presented. For simulation and optimization of the synthesizable, VHDL code Software is used to implement 128-bit data encryption process. Xilinx ISE 12.3i Software is adopted for simulation. An iterative approach is used to simulate all the transformations of algorithm to minimize the

Corresponding Author: Nasseer M. Basheer, Department of Computer Technology Engineering, Technical Engineering College of Mosul, Mosul, Iraq

hardware consumption. The algorithm achieves throughput reaching the value of 1609 Mbit/sec for encryption process with device XC6vlx240t of Xilinx Virtex family. The number of slices registers is 954 out of 301440 and the number of bonded IOBs blocks is 65% (Singh and Dod, 2016).

Cristian *et al*. (2015) used FSM approach which is used to encrypt and decrypt the input blocks of data. The Rijndael cipher is estimated in terms of the implementation in FPGA. The new architecture that have been presented achieves the implementation of the Rijndael cipher with high speed of encryption\decryption. The results showed that the overall throughput of encryption and decryption is 739 Mbit/sec which is done by using single FPGA device Virtex II XC2V1000-4. The used resources of FPGA are up to 84% out of the total resources (Cristian *et al*., 2015).

According to Tonde and Dhande (2014) used an iterative looping approach having 128 bits of block and key size. Very high speed integrated circuit hardware descriptive language is used to code the design. The collected results have been synthesized and simulated by using Xilinx ISE and ModelSim Software, respectively. The algorithm fulfills the throughput reaching the value of 672.52 Mbit/sec for encryption and 603.59 Mbit/sec for decryption. AES algorithm is simulated on ModelSim Software and implemented on Xilinx XC3S500E Spartan-3E FPGA kit (Tonde and Dhande, 2014).

According to Hoang Trang and Nguyen Van Loi (2012) presented in their study an effective FPGA implementation of 128 bit block data and 128 bit key AES algorithm. The design is coded using verilog HDL. The whole results are being synthesized and simulated based on the quatus 9.0, the ModelSim-Altera 6.4a and EP 20 K 400 CB 652 C7 device. The adopted approach in this proposed design is the iterative approach for cryptographic algorithms. The throughput for encryption is 1054 Mbit/sec and 615 Mbit/sec for decryption (Cristian *et al*., 2015).

Advanced Encryption Standard (AES): The Rijndael algorithm is considered an iterated block cipher having different key length. The key length is be specified to 128.192 or 256 bits independently. The AES algorithm consists of:

- An initial data and original key
- 9 rounds for 128 bits, 11 rounds for 192 bits or 13 rounds for 256 bits of the standard rounds
- A final round that is different from the standard rounds

Table 1: Number of rounds that depends on key length and block size (Ahmed *et al*., 2013)

Variables	Key length (Nk)	Block length (Nb)	No. of rounds
AES_128	4	4	10
AES_192	6	4	12
AES_256	8	4	14

B0, 0	B0, 1	B0, 2	B0, 3
B1, 0	B1, 1	B1, 2	B1, 3
B2, 0	B2, 1	B2, 2	B2, 3
B3, 0	B3, 1	B3, 2	B3, 3

Fig. 1: Block state

K0, 0	K0, 1	K0, 2	K0, 3
K1, 0	K1, 1	K1, 2	K1, 3
K2, 0	K2, 1	K2, 2	K2, 3
K3, 0	K3, 1	K3, 2	K3, 3

Fig. 2: Key state

The number of standard rounds is highly dependent on the key length. However, the original key can be expanded generating the round keys in which each size is equal to block length and each round of the algorithm gets a new round key taken from the key schedule module. Hence, due toitsregular structure, AES could be implemented in a very efficient way in hardware and software. The implementation in hardware makes the time of encryption and decryption faster (Kosaraju, 2003). Table 1 illustrates the length of the expanded key (N_K) for different key sizes where N_r refers to the number of rounds where, $N_r = 10$ when $N_k = 4$ words, $N_r = 12$ when $N_k = 6$ words and $N_r = 14$ when $N_k = 8$ words.

The AES standard fixes the data block length to reach to 128 bits. The input (plain text) and output (cipher text) of the algorithm are being processed in blocks, each block consists of 128 bits. Each data block, to be encrypted is converted to an array of 16 bytes with each encryption process, being byte-oriented. The AES algorithm has variable transformations in which the data block and the intermediate results are known as states. The block state is shown as a square array of the bytes where the key state and the block state are being put in column order. The coding of intermediate values are shown as a block state matrix of ($4*N_b$) bytes, N_b = (block length)/32 as shown in Fig. 1. In a similar way as illustrated in Fig. 2 the original and round keys of 128 bits are shown as a key state matrix of ($4*N_k$) bytes, N_k = (key length)/32. Therefore, for 128 bits block and 128 bits key, the values of $N_b = 4$, $N_k = 4$ and $N_r = 10$. A Rijndael round moves the data by using nonlinear substitutions, additions and multiplications (Kosaraju, 2003; Ahmed *et al.*, 2013; Soumya and Kishore, 2013). Each standard round consists of four major transformations performed on the arraysof 16 bytes which are:

- Bytes sub transformation
- Shift rows transformation
- Mix columns transformation
- Round key addition

The final round is similar to other rounds, except it does not include mixcolumns transformation. However, decryption is achieved by applying the inverse transformations of encryption. Hence, the standard round operations of decryption are differentfrom those in encryption. The computations in decryption are different from encryption because at the first the inverse transformations existed in the rounds are more complicated than those of the encryption. The inverse mixcolumns transformation is more complicated than the mixcolumns transformation due to the coefficients of the polynomial that are used in decryption are being of a higher order.

AES encryption: The encryption of AES algorithm is illustrated in Fig. 3a. The block diagram shows the specifications of AES consisting of a number of transformations that are applied on the data block bits and in a fixed number of rounds. The length of the key determines the number of rounds used in encryption.

SubBytes transformation Non-linear bytes substitution is included in the SubBytes transformation, operating in an independent way on each of the state bytes through using a once-precalculated substitutive table known as S-box containing 256 values and their corresponding values (Tonde and Dhande, 2014; Manteena, 2004; Soumya and Kishore, 2013).

ShiftRows transformation: ShiftRows transformation includes the state rows which are left-shifted cyclically.

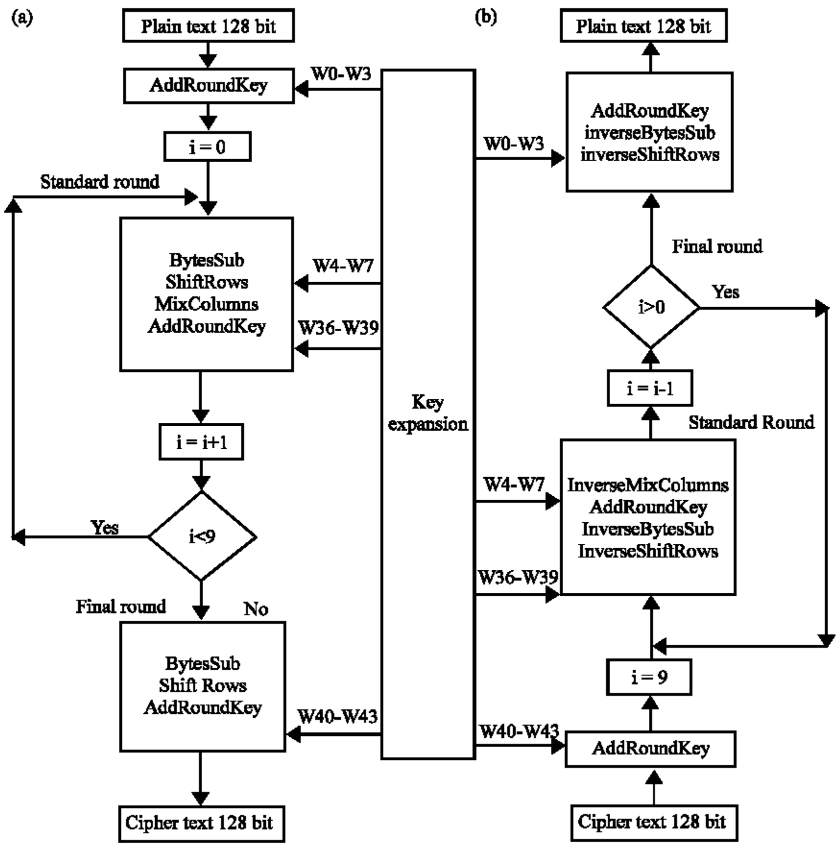

Fig. 3: a) AES encryption process and b) AES decryption process

Row 0 remains unchanged, row 1 shifts 1 byte to the left side; row 2 shifts 2 bytes to the leftside and row 3 shifts 3 bytes to the left (Tonde and Dhande, 2014; Manteena, 2004; Soumya and Kishore, 2013).

MixColumns transformation MixColumns transformation includes columns of the states that are polynomials over Galois Fields where GF (2^8) and is multiplied by (x^4+1) within a fixed polynomial c(x) that is given by: $c(x) = [03]x^3+[01]x^2+[01]x+[02]$ (Tonde and Dhande, 2014; Manteena, 2004; Soumya and Kishore, 2013).

ADDRoundKey transformation: The state matrix resulted from mixcolumns is XORed with a round key. The round key for every round is generated from the original key by using the key expansion algorithm. The encryption and decryption algorithm requires eleven times of 128-bit round key, from round key (0) up to round key (10) for 9 standard rounds and 1 non-standard (Tonde and Dhande, 2014; Manteena, 2004; Soumya and Kishore, 2013).

AES decryption: AES decryption is illustrated in Fig. 3b. Decryption is direct inverse of encryption. The entire transformations applied in encryption are being inversely applied indecryption. The output values of the final round of encryption of both data and key become the input values for first round for the decryption (Manteena, 2004).

AddRoundKey: AddRoundKey function for decryption is the same that of encryption. However, in decryption the cipher text state XOR with round key which is generated from key expansion algorithm. The round keys have to be selected in reversely (starting from round 10-0, respectively).

InvShiftRows transformation: InvShiftRows functions are similar to ShiftRows, except that shifting rows are performed to the right cyclically instead of left.

InvSubBytes transformation: InvSubBytes is being achieved by a substitutive table called InvS-box containing 256 values which ispre-calculated with their, respective values.

InvMixColumns transformation: InvMixColumns transformation includes, polynomials degree that is <4 over GF (2^8) coefficientins are the elements existed in the columns of the state are multiplied by (x^4+1) through a fixed polynomial $d(x) = [0B]x^3+[0D]x^2+[09]x+[0E]$ in which 0B, 0D, 09, 0 E refer to hexadecimal values (Ankita *et al.*, 2017; Varhade and Kasat, 2015).

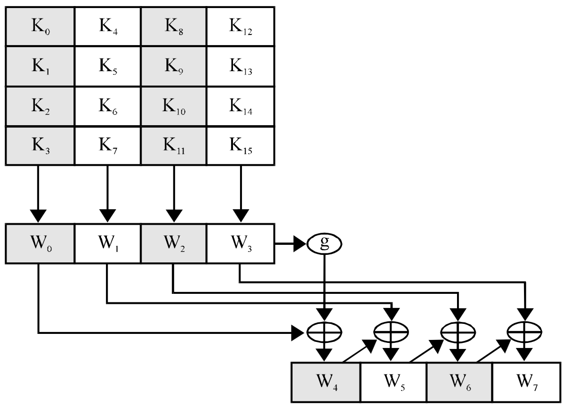

Fig. 4: Key expansion (Soumya and Kishore, 2013)

Key expansion: By expanding the initial key, the roundkeys are generated. The 128 bit original key needed to be expanded to N_b. $[N_r+1]$ of 32 bit words where $N_b = 4$ and $N_r = 10$. This results in 44 words (each with 32-bits). The length of the expanded key has to be 128 bits resulting in 11 keys of 128 bits. The orginal key is the encryption key that is used in the first round. The function g would take the previous word and shifts it 1 byte to the left, after that s-box operation is performed on every byte of the shifting result. In the final step, the substituted words resulting from subBytes operation are XORed with a round constant hexadecimal word array "RC (i), 0, 0, 0" in which $1 \leq i \leq 10$. RC given in atable in hexadecimal for every round. The main goal of using round constants is for eliminating symmetries and similarities by making the 4-word expanded key for each round. The same expanded key algorithm for encryption and decryption is used. For encryption, the round keys are used successively from round 0-10. In decryption, the round keys are used reversely from round 10-0. Fig. 4 illustrates the key expansion process of AES algorithm (Kosaraju, 2003). Key in hex (128 bits) = $k_0, k_1, k_2, ..., k_{15}$.

Where:

g	=	(XOR round constant (s-box(circular byte left shift of w_3)))
w_4	=	w_0 XOR g
w_5	=	w_1 XOR w_4
w_6	=	w_2 XOR w_5
w_7	=	w_3 XOR w_6

MATERIALS AND METHODS

Design methodology: FSM approach which is a mathmatical mode for computation is used in this study to implement AES algorithm for encryption/decryption. FSM method in this design, consists of four states: initialize-state where round keys are generated, standard round state including nine rounds, final round state and done state, occuring when encryption or decryption is completed. Each round needs one clock cycle and generated round keys takeone clock cycle, subsequently, twelve clock cycle are needed to encrypt data.

FSM encryption: At the beginning, the system receives the 16 byte input data block (plain text) in initialize_state. When the data block becomes ready, the system moves towards the next state (s0). The s0 state represnts the first round of encryption in which the original data (plain text) is being xored with the original input cipher key. This XORed data is to be applied to S-Box, then ShiftRow transformation and Mix Column sholud be performed. The MixColumns transformation is considered a complex process, to decrease the complexity and eventually enhancethe speed, pipelined architecture is used by splitting the MixColumns transformation to 4 states of FSM, one is given for each column. After the intial round algorithm moves towards the second state s_1 where the output of first round is being XORed with the subkey generatedby using the original key and consequently, isapplied to the four transformations. For the states in the next eight rounds, this state is being repeated in the similar way. In the final round all the process stays the same except the MixColumn transformation which is excluded here. When the last round (10th round) is fully completed, the encryption is finished for the first 16 byte data block and the system is ready to encrypt the next 16 byte data block.

FSM decryption: Decryption is the reverse process of encryption. The system moves to s_0 state where the first round of decryption is performed which is similar to the last round of encryption. The InvMixColumn layer is not avilable in first round of decryption. Therefore, it is switched to next state where operations are beingrepeated in the same way for the coming nine rounds including InvMixColumn transformation. After the final round is completed, the system decrypts the cipher text back to plain text (input data) that provides a decrypt signal for indicating that decryption is being done for the first 16 byte data block.

RESULTS AND DISCUSSION

At the beginning, the AES algorithm is solved manually for certain 16 byte block available on the net. The manual and practical results are matched. Using VHDL language, the design of AES algorithm is coded. AES algorithm is simulated and synthesized on Xilinx ISE 10.1 Software. The AES encryption/decryption on a Xilinx XC3S500E Spartan-3E FPGA kit is also implemented.

Table 2: AES encryption plain text, key, cipher text

AES encryption	Values
Plain text 128 bit (Input)	54776F204F6E65204E696E652054776F
Encryption key 128 bit (Input)	5468617473206D79204B756E67204675
Ciphered text 128 bit (Output)	29C3505F571420F6402299B31A02D73A

Table 3: Synthesis result of AES encryption

	Device utilization summary (estimated values)		
Logic utilization	Used	Available	Utilization (%)
No. of slices	2156	4656	46
No. of slice flip flops	599	9312	6
No. of 4 input LUTs	3738	9312	40
No. of bonded IOBs	129	232	55
No. of GCLKs	1	24	4

Table 4: AES decryption cipher text, key, plain text

AES decryption	Values
Ciphered text 128 bit (Input)	29C3505F571420F6402299B31A02D73A
Decryption key 128 bit (Input)	5468617473206D79204B756E67204675
Plain text 128 bit (Onput)	54776F204F6E65204E696E652054776F

Table 5: Synthesis result of AES decryption

	Device utilization summary (estimated values)		
Logic utilization	Used	Available	Utilization (%)
No. of slices	2354	4656	50
No. of slice flip flops	315	9312	3
No. of 4 input LUTs	4092	9312	43
No. of bonded IOBs	129	232	55
No. of GCLKs	1	24	4

Simulation of encryption: Table 2-4 illustrates the inputs and output of the encryption. For AES Encryption, 128 bit plain text and encryption key are used as an input and 128 bit cipher text as an output. Figure 5 shows part of the simulation waveforms.

Figure 5 illustrates the simulation of AES algorithm through the using of the FPGA platform in the Xilinx ISE 10.1 Software. It shows the result of transformation of plain text to cipher text by using theencryption key in the encryption stage.

Table 5 gives the synthesis results for AES encryption, giving the number and percentage usage of the kit hardware resources.

Simulation of decryption: The inputs and output of the decryption is shown in Table 3. In AES decryption 128 bit cipher text and decryption key as an input are used, we have the 128 bit plain text (original text) as an output. The decryption key used here is the same encryption key used in simulating the AES encrytion. Figure 6 shows part of the simulation waveforms.

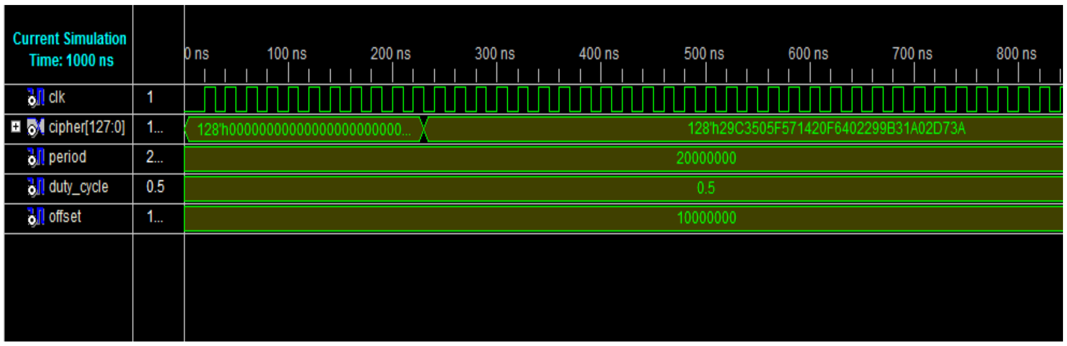

Fig. 5: Simulation of encryption

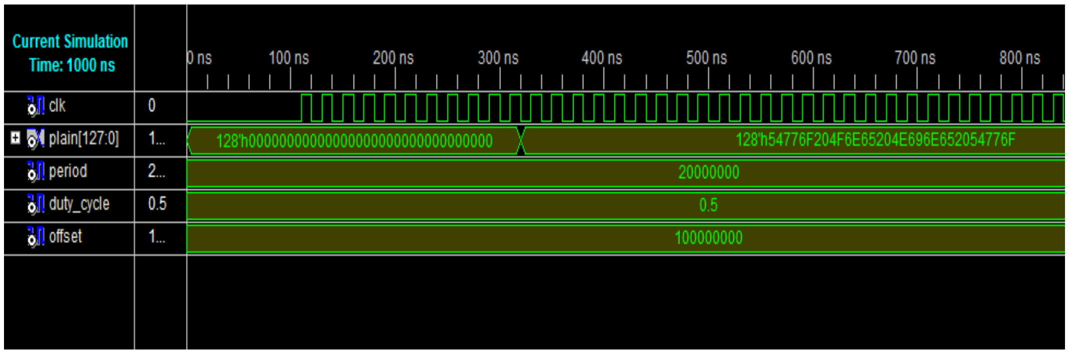

Fig. 6: Simulation of decryption

Table 6: Results (for 128 bit)

Parameters/designs	Results
Maximum frequency	
Encryption	116 MHz
Decryption	87 MHz
Throughput	
Encryption	1.349 Gbps
Decryption	1.012 Gbps
Delay	
Encryption	95 nsec
Decryption	126 nsec

Figure 6 shows the simulation of AES algorithm using FPGA platform in the Xilinx ISE 10.1 Software. Figure 6 shows the resultof the transformation of ciphered text to plain text through using decryption key in decryption.

Table 6 shows the synthesis results for AES decryption, showing the number and percentage usage of the kit hardware resources.

By noting the synthesis results for both parts of the AES system, it is noted that the hardware resources are not enough to synthesis the encryption and decyption on 1 kit. The number of bonded I/O blocks is 55% of the full resources for each of the operations. From the synthesis reports, maximum frequencyis resulted and it is used to calculate throughput and delay. The throughput and delay (Cristian *et al*., 2015) are given by:

Throughput = (Block length/No. of clock)* max. frequency

Throughput = 128 bits/11 clocks *116 MHz = 1.349 Gbps for encryption

Throughput = 128 bits/11 clocks *87 MHz = 1.012 Gbps for decryption

Delay = 1/(max. frequency/No. of clock)

Delay = 1/(116MHz/11) = 95 nsec for encryption

Delay = 1/(87MHz/11) = 126 nsec for decryption

The results of the simulation are summerized in Table 4.

CONCLUSION

AES had been implemented using the FPGA in this study. FSM is used for AES represention. The encryption\decryption modules are implemented with datablock and key that are equal to 128 bits. The 128 cipher key is used to provide high security, due to the face that 128 bit cipher key is hard to be broken. The encryption and decryption had been both simulated using he Xilinx ISE_10.1 simulator. The aim of design was to get the maximum performance and through put. Part of the desgin made use of the pipeline principles to get better performance. The encryption and decryption were implemented seperatly becuase the kit resources are not enough to handle them both. The througput is 1.349 Gbps while for the decryption is 1.012 Gbps. It is intended to apply the pipelining principle thoroughly for all the encryption and decryption states to get better throughput and higher clock frequency.

RECOMMENDATIONS

For the future research, AES algorithm will be implemented for encryption and decryption using a pipeline design in the software implementation by using VHDL language.

REFERENCES

Ahmed K.S., A. Liakot, M.B. Karim and S.M.T. Ahmad, 2013. FPGA implementation of an AES processor. J. Inf. Commun. Technol., 1: 1212-1218.

Ankita, J., C. Prajakta, G. Tejaswini and S.C. Wagaj, 2017. Implementation of Advanced Encryption Standard (AES) on FPGA. Intl. J. Eng. Sci. Comput., 7: 12313-12317.

Cristian, U.C., C. David, C. Charles, V. Ingrid and C. Frank, 2015. A hardware implementation in FPGA of the Rijndael algorithm. Master Thesis, Department of Electrical Engineering, University of California Los Angeles, Los Angeles, California.

Gurpinder, K. and S.A. Singh, 2014. Efficiently high speed implementation of AES algorithm on FPGA. Intl. J. Res. Appl. Sci. Eng. Technol., 5: 1088-1093.

Kosaraju, N.M., 2003. A VLSI architecture for Rijndael, the advanced encryption standard. BA Thesis, College of Engineering, University of South Florida, Tampa, Florida, USA.

Manteena, R., 2004. A VHDL implementation of the advanced encryption standard-rijndael algorithm. Master Thesis, College of Engineering, University of South Florida, Tampa, Florida, USA.

Singh, K.P. and S. Dod, 2016. An efficient hardware design and implementation of Advanced Encryption Standard (AES) algorithm. Intl. J. Recent Adv. Eng. Technol., 4: 5-9.

Soumya, K. and G.S. Kishore, 2013. Design and implementation of rijndael encryption algorithm based on FPGA. Intl. J. Comput. Sci. Mob. Comput., 2: 120-127.

Tibrewal, A. and V. Kumar, 2014. FPGA implementation of advanced encryption standard. BA Thesis, Department of Electronics and Communication Engineering, National Institute of Technology, Rourkela, India.

Tonde, A.R. and A.P. Dhande, 2014. Implementation of Advanced Encryption Standard (AES) algorithm based on FPGA. Intl. J. Curr. Eng. Technol., 4: 1048-1050.

Varhade, S.A. and N.N. Kasat, 2015. Implementation of AES algorithm using FPGA and its performance analysis. Intl. J. Sci. Res., 4: 2484-2492.