Design and Implementation of Advanced Encryption Standard Security Algorithm using FPGA

Adnan Mohsin Abdulazeez,
Duhok Polytechnic University
And
Ari Shawkat Tahir
University of Zakho

Abstract- In this paper, two architectures have been proposed, one for AES Encryption 128-bit process, and the other for AES Decryption 128-bit process. Both architectures are based on an iterative structure and modifications such as merging transformation (SubByte and ShiftRow in Encryption process, and Inverse SubByte and Inverse ShiftRow in Decryption process), Look Up tables for decryption, generating keys, and optimization of each clock cycle to incorporate maximum number of operations to improve the throughput and reducing hardware resources. The design has been described by VHDL and simulated by using Xilinx ISE 9.2i.The architectures have been implemented on reconfigurable platforms FPGAs. Accomplishment when implemented on Xilinx_Virtex4 (device xc4vlx80, package 12ff1148) which confirms that the proposed architectures have minimum hardware resource, where only 9% of the chip resources are used for AES Encryption design with realizable operating clock frequency of 382.988MHz, and only 9% of the chip resources are used for AES Decryption design with realizable operating clock frequency of 382.988MHz.

Index Terms— — Cryptography, Advanced Encryption Standard (AES).

1 Introduction

The importance of cryptography applied to security in electronic data transactions has required an essential relevance during the last few years. Cryptography is the art and science of protecting information from undesirable individuals by converting it into a non-recognizable form by its attackers while stored and transmitted. Data cryptography mainly is the scrambling of the content of data, such as text, image, audio, video and so forth to make the data unreadable, invisible or unintelligible during transmission or storage. This is called encryption [1]

2 ALGORITHM DESCRIPTIONS

The Advanced Encryption Standard (AES) was announced by the National Institute of Standards and Technology (NIST) in November 2001. It is the successor of Data Encryption Standard (DES), which cannot be considered as safe any longer, because of its short key with a length of only 56 bits [2].

AES is a block cipher with a block length of 128 bits. The initial input to the algorithm is stored in a 4X4 byte matrix called State and operations are performed on this matrix. Three different key lengths of 128 bits, 192 bits and 256 bits are supported. The numbers of rounds are 10, 12 and 14 for key lengths 128, 192 and 256 bits, respectively. Each round consists of the following operations, namely, Substitute Bytes, ShiftRows, MixColumns and AddRoundKey[3]. As shown in Fig (1.a)

The first step in AES algorithm is "Add round key". It is followed by n-1 rounds (n depends on the key length as mentioned before) each comprising Substitute bytes, Shift rows, Mix columns and Add round key operations. The last round contains Substitute bytes, Shift rows and Add round key operations. The algorithm starts and ends with Add round operation [3].

The SubBytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (Sbox). This Sbox, which is invertible, is constructed by composing two transformations [4]:

Take the multiplicative inverse in the finite field GF (28), the element {00} is mapped to itself.

Apply the following affine transformation (over GF(2)):

1. Take the multiplicative inverse in the finite field GF(2^8), the element {00} is mapped to itself.
2. Apply the following affine transformation (over GF(2^8)):

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \qquad (2.1)$$

The ShiftRow operation changes the byte position in the State matrix. Each row of the matrix is rotated with different offsets to obtain a new State matrix; the first row is unchanged; where the second, the third and the fourth ones are respectively rotated one byte, two bytes, and three bytes to the left

The MixColumns transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial a(x), given by

$$a(x) = \{03\}x3 + \{01\}x2 + \{01\}x + \{02\} . \qquad (2.2)$$

This can be written as a matrix multiplication. Let $S'(x) = a(x) \otimes S(x)$:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb$$

In the AddRoundKey transformation, the state is modified by combining it with a round key with the bitwise XOR operation. A round key is denoted by Expanded Key[i], $0 \le i \le Nr$. The array of round keys ExpandedKey is derived from the cipher key by means of the key schedule (see Sect. 2.7.3). The round key length is equal to the block length [5].

AES algorithm decryption process is shown in Fig.1.b. Inversed its encryption process will be able to decrypt the cipher text.

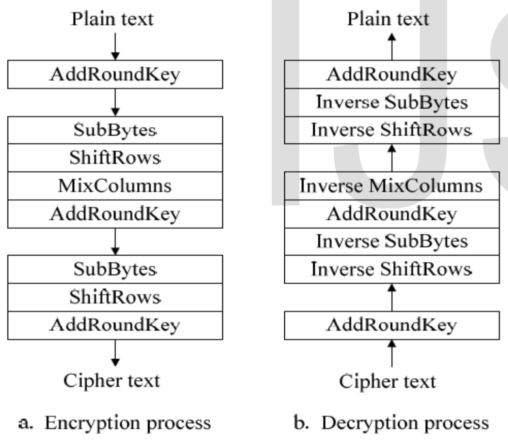


Figure1. AES Encryption/Decryption Processes

3 The Related Works

In 2009, Ghaznavi, et. al., [6] introduced a technique for the FPGA implementation of the MixColumns transformation, an important part of AES. The proposed design provides the smallest hardware usage on an FPGA. Overall, the AES encryption implementation with the proposed MixColumns architecture reduces usage of hardware resources. The improvement is obtained by more efficient resource sharing through expansion and rearrangement of the MixColumns equation with respect to the structure of FPGAs contains 4-input LUTs. The present study can be highly useful for efficient utilization of hardware resources on FPGAs in modes using encryption of AES. The proposed design can be used to provide security services such as confidentiality or authentication by these modes.

In 2009, Banraplang, et. al., [7] presented the hardware implementation of Advanced Encryption Standard (AES) algorithm. In this paper, it worked with an iterative structure and Look Up tables for decryption, and optimization of each clock cycle to incorporate maximum number of operations. The encryption and decryption process of AES algorithm was captured in VHDL language and corresponding FPGA implementation resulted in reduced number of slices and achieved a data throughput of 1.4 Gbit/sec.

In 2010, Zhang, et. al., [8] proposed an implementation of the AES-128 cryptographic algorithm using outer-round only pipelined architecture. The proposed design uses the Block RAM storing the S-box values and exploits two kinds of Block RAM. By combining the operations in a single round, the presented design can reduce the critical delay. Therefore, the presented design can achieve a throughput of 34.7 Gbps at 271.15 MHz and 2389 CLB Slices with 200 BRAM. The AES-128 architecture presented has been implemented using Veilog HDL.

In 2010, Wang, et. al., [4] proposed an architecture design for compact hardware implementation of an AES encryption core. In that research, the area and speed performance of applying a pipelined S-box to compact AES hardware implementations was examined. The proposed design employs a single 4-stage pipelined S-box that is shared by the data path operation and the key expansion operation. It can achieve an increase in throughput of 2.1 times while maintaining a similar gate count, indicating that pipelined S-boxes are applicable to compact implementations of AES for the purpose of speed improvement.

In 2010, Ahmed, [9] presented a modified Rijndael algorithm capable of encrypting a 128 bit input/output/key. The presented algorithm depends on substitution and permutation network (SPNetwork) rather than feistel network. A new mirror stage has been added to increase the complexity of algorithm. The modified algorithm has been realized and simulated using VHDL. The introduced architecture was implemented by VHDL, schematic and core generator – Based Design which are synthesized, placed and routed in Virtex XCV800-6bg432 which resulted in an optimized area (7148) slices and (44) MHz clock speed.

4 The Proposed Algorithm

In addition to the software implementation for the AES algorithm (using C++ for software implementation), the AES architecture has been described by VHDL (Very high speed integrated circuit Hardware Descriptive Language) and simulated by using Xilinx ISE 9.2i.

In this thesis, there are two architectures designed separately.

1. The first architecture is designed for AES Encryption Algorithm.
2. The second architecture is designed for AES Decryption Algorithm.

4.1 Design of the AES Encryption Algorithm

This design used is to generate the encrypted data (ciphertext) from the original data (plaintext). The proposed design of the AES Encryption Algorithm using one 255-bit Roms named Sbox, which is used by KeyExpanisin component to generate a new key, and also used by SubShiftbyte component. This Rom is storing a fixed values, this value can't be changed. Figure (4.1) shows the block diagram of AES Encryption:

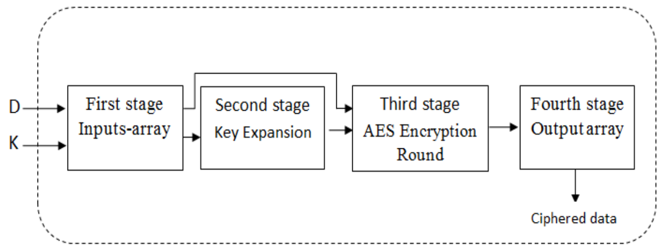


Figure (4.1): the Block Diagram of AES Encryption

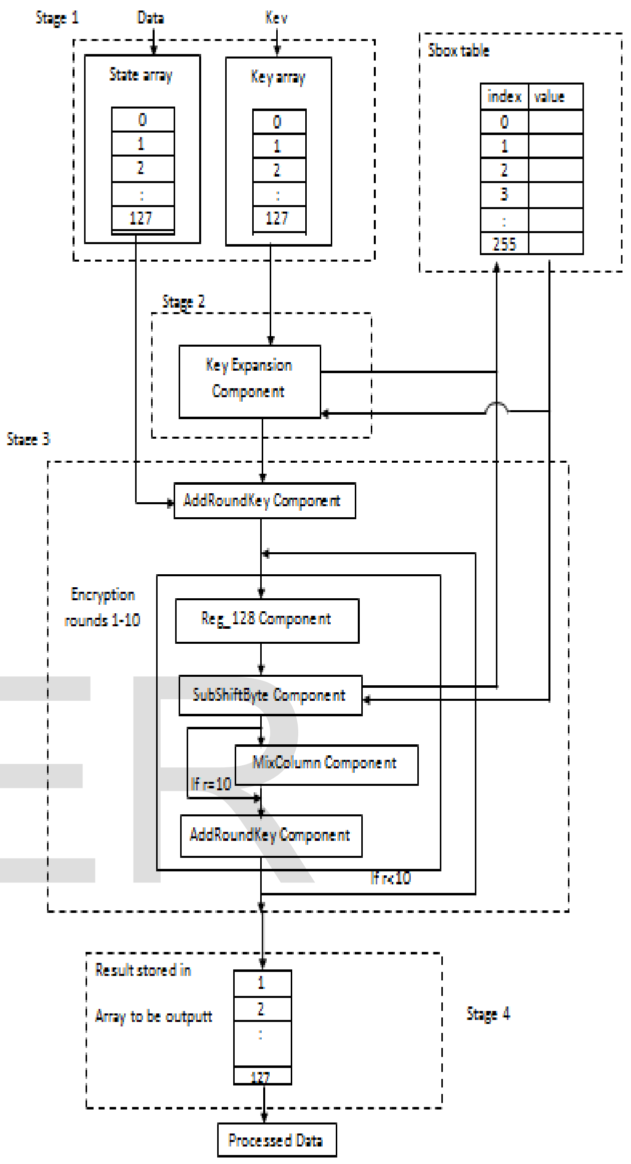
- **The first Stage of AES Encryption Diagram**
  In this stage, reading 128-bit Data and entered to an array called state array and also reading 128-bit Key and it's entered to an array called key array. This array contains 128 rows, each row contains 1-bit. The state array represents the plaintext which is processed to get the ciphertext.

- **The Second Stage of AES Encryption Diagram**
  In this stage, it receives 128-bit key array from the previous stage then it is manipulated to produce new keys to be used in the AES Encryption rounds. it performs three operations on the key array to produce a new key. Some of specified entries of the key array are replaced by another entries located in the ROM memory named Sbox and there is a new variable named Constant Round added to the key during generating new key. The value of the Constant Round is changed from one round to another. Then some entries of the key are Xored with the new entries and Constant Round are producing new key for specified AES encryption round.

- **The Third Stage of AES Encryption Diagram**
  The third stage receives the state array from the first stage and the key array. In this stage, the state array contains changes when going through several components. At the end, the processed state array either from Encryption component is sent to the next stage.
  In this stage, there are four components as shown below, these four components iterated in 10 rounds except round 10 where the MixColumn is not activated.
  1. Reg_128 Component
  2. SubShiftByte Component.
  3. MixColumn Component.
  4. AddRoundKey Component.

- **The Fourth Stage of AES Encryption Diagram**
  The fourth stage receives state array from the previous stage and entered in the new array named output. The output array consists of 128 rows; each row consists of 1-bit. The output array represents the cipher-text. The output represents the processed data. At the end, it gets the 128-bit ciphertext from 128-bit plaintext.

The figure (4.2) shows the AES encryption process



The figure (4.2): AES Encryption Process

4.2 Design of the AES Decryption Algorithm

This design used to generate the original data (plaintext) from the entered data (ciphertext). The proposed design of the AES Decryption Algorithm uses two 255-bit ROMs, the first one is called Sbox used by KeyExpanisin component to generate a new key, and the second Rom is named InvSbox, it is used by InvSubShiftbyte component. Both ROMs store a fixed value, this value cannot be changed. The block diagram of AES Decryption is shown below:

There are four stages in our architecture of the AES Decryption Algorithm as shown in Figure (4.3).

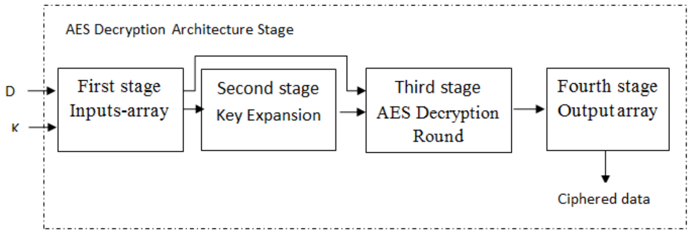

AES Decryption Architecture Stage
D
K
First stage Inputs-array
Second stage Key Expansion
Third stage AES Decryption Round
Fourth stage Output array
Ciphered data

Figure (4.3**):** the Block Diagram of AES Decryption

- **The first Stage of AES Decryption Diagram**

In this stage, reading 128-bit Data and entering to an array is called the state array and also reading 128-bit Key entering to an array is called the key array. These arrays contain 128 rows, each row contains 1-bit. The state array represents the ciphertext which is processing to get the original data.

- **The Second Stage of AES Decryption Diagram**

This stage is the same stage in AES Encryption Diagram, so it is explained in (Sec 4.1).

- **The Third Stage of AES Decryption Diagram**

The third stage receives the state array form the first stage and the key array from the second stage. In this stage, the state' array contain changed when goes through several components. At the end, the processed state array either from Encryption component is sent to the next stage.

In this stage, there are four components as shown below; these four components iterated in 10 rounds except the round 10 where the InvMixColumn is not activated.

1. Reg Component
2. InvSubShiftByte Component.
3. InvAddRoundKey Component.
4. InvMixColumn Component.

The first component is called Register (Reg) which has an input signal Reset for initialization of this register to the value 0x00. The second component is called InSubShiftByte. The InvSubByte and InvShiftRow transformers are combined in a single component named InSubShiftByte.

This component receives state array from the first component, then changes the state array elements by other elements stored in a ROMs memory called InvSbox array at the same time it changes the state array entries positions to get the correct positions. To changes the state array entire, there is a ROM memory created and storing the InvSbox values in the created ROM. Those values are fixed and unchangeable, so InSubShiftByte sends the state entire which will be the index of the Rom and the Rom returns the values from InvSbox table depending on the index.

The third component component tried to XOR the state array with the key array and stored the results in the state array, the last component in the third stage is called InvMixColumn component. It dividing the state array into four columns each of them is multiplied with the specified row in the fixed array to get a value comprising of mixing the four elements in the column. The InvMixColumn is active for all rounds in Decryption process except the last one which will not be activated.

- **The Fourth Stage of AES Decryption Diagram**

The fourth stage receives the state array from the previous stage and entered in the new array named output.
Figure (4.4) shows the AES Decryption Process

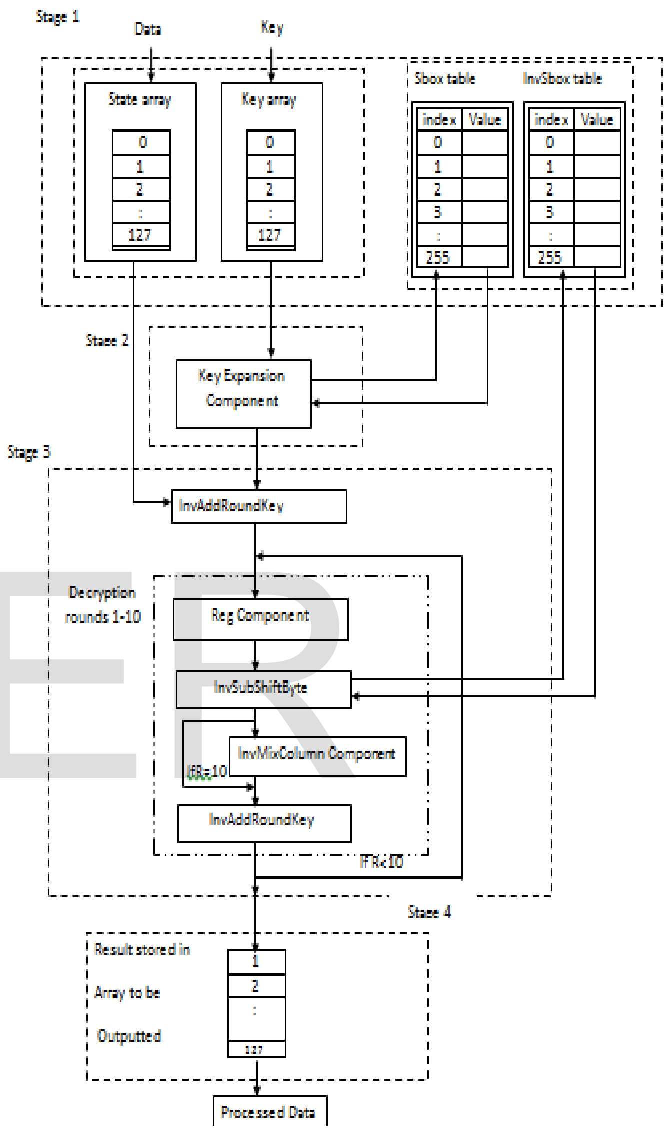


Figure (4.4): AES Decryption Process

5 Experimental Results:

5.1 Results of the proposed architecture of the AES Encryption algorithm

The AES Encryption has been described by VHDL and simulated by using Xilinx ISE 9.2i. Table (5.1) shows the 128-bit input plaintext, 128-bit key, number of rounds and 128-bit ciphertext. The Simulation results for the above input 128-plaintext is shown in Figure (5.1):

Table (5.1): AES Encryption Input **and Output**

Input 128-bit plaintext	00112233445566778899aabbccddeeff
Input 128-bit key	000102030405060708090a0b0c0d0e0f
Number of Round	10 rounds
Output 128-bit ciphertext	69c4e0d86a7b0430d8cdb78070b4c55a

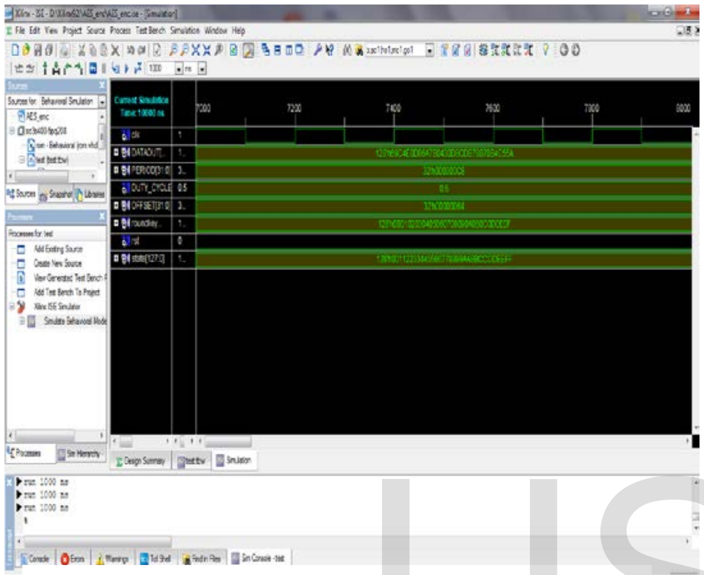


Figure (5.1): Simulation of AES Encryption Process

5.2AES Encryption Architecture Synthesis

The proposed architect-ture has been synthesized and implemented successfully on the device family Virtex4 (device XC4VLX80, package FF1148). The device utilization summary is shown in Table (5.3), and the timing summary and memory summary are shown in Table (5.4).

Table (5.3): Device Utilization Summery of AES Encryption Design

Logic utilization	Used	available	utilization
Number of Slice	3229	35840	9%
Number of Slice Flip Flops	408	71680	0.57%
Number of 4 input LUTs	6293	71680	8%
Number of bonded IOBs	266	768	34%
Number of GCLKs	1	32	3%

Table (5.4): Timing and Memory Summary of AES Encryption Design.

Minimum period	2.611ns (Maximum Frequency: 382.988MHz)
Minimum input arrival time before clock	36.604ns
Maximum output required time after clock	3.793ns
Total memory usage	397740 kilobytes

5.3 Results of the proposed architecture of the AES Decryption algorithm

The AES Decryption has been described by VHDL and simulated by using Xilinx ISE 9.2i. , like the AES Encryption.

The proposed algorithm tries to decrypt the block of 128-bit of plaintext using 128-bit of keys to produce the block of 128-bit of the original data (plaintext). Simulations have been done on the data presented in Table (5.3) choosing that input ciphertext and key randomly. These data entered the architecture then processed through 10 rounds. At the end, the architecture will produce 128-bit plaintext.

Table (5.2) shows the 128-bit input plaintext, 128-bit key, number of rounds and 128-bit ciphertext. The Simulation results for this input 128-plaintext is shown in Figure (5.2):

Table (5.2): AES Decryption Input **and Output**

Input 128-bit ciphertext	69c4e0d86a7b0430d8cdb78070b4c55a
Input 128-bit key	000102030405060708090a0b0c0d0e0f
Number of Round	10 rounds
Output 128-bit plaintext	00112233445566778899aabbccddeeff

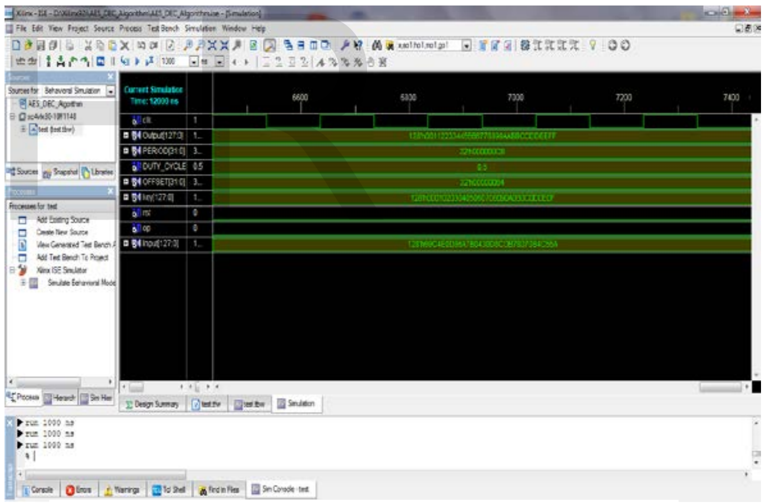


Figure (5.2): Simulation of AES Decryption Process

5.4 AES Decryption Architecture Synthesis

The other step after simulation is the synthesis, for this we have used FPGAs (Field Programming Gate Arrays).The architecture has been synthesized and implemented successfully on the device family Virtex4 (device XC4VLX80, package FF1148), the Virtex4 starter kit board shown in Figure (5.2). The device utilization summary is shown in Table (5.6), the timing summary and memory summary are shown in Table (4.6).

Table (5.5): Device Utilization Summery of AES Decryption Design

Logic utilization	used	available	Utilization
Number of Slice	3283	35840	9%
Number of Slice Flip Flops	880	71680	1%
Number of 4 input LUTs	6359	71680	8%
Number of bonded IOBs	266	768	34%
Number of GCLKs	1	32	3%

Table (5.6): Timing and Memory Summary of AES Decryption Design.

Minimum period	2.482ns (Maximum Frequency: 402.909MHz)
Minimum input arrival time before clock	50.239ns
Maximum output required time after clock	4.677ns
Total memory usage	414316 kilobytes

5.5 Results Comparison

The results are compared to the results produced by other studies presented in the literature survey. So Table (5.7) shows the results produced by this work and the results produced by other studies presented in the literature survey.

Table (5.7): Comparative Results of FPGA-Based AES

6 Conclusions

Design	Device	Freq. (MHz)	Slices
B. Jyrwa [7]	XC2VP30	142.5	6211
A. A. Mohamed [9]	XCV800	44	7148
Yulin , Xinggang [8]	XCVP70	271.150	2829
AES Enc. This work	XC4VLX80	382.988	3229
AES Dec. This work	XC4VLX80	402.909	3283

The Keys have been generated for all rounds before the data goes through the rounds. The SubShiftByte entity merges both the ShiftRow and Subbyte transformation and InvSubShiftbyte merges both InvShift and InvSubByte entities. The Sbox and InvSbox array's elements have been stored in the memory in order to be ready for substitute byte. The pervious steps applied on the AES Encryption and AES Decryption architecture in order to improve the frequency and reducing the hardware utilization. Using pipeline technique's to increase the throughput.

Simulation result s show the validity of the AES encryption design while the synthesis results show that 9% of the chip resource are used when implementing the design on Xilinx_Virtex4 (device xc4vlx80, package 12ff1148). This means minimum resources are used for the design with clock frequency of 382.988MHz.

while the Simulation results for AES Decryption design show the validity of design while the synthesis result show that 9% of the chip resources are used when implemented on Xilinx_Virtex4 (device xc4vlx80, package 12ff1148). This means minimum resources are used for the design with clock frequency of 402.909MHz**.**

References

[1] M. Abomhara, Z. Omar and O. Othman, "An Overview of Video Encryption Techniques", International Journal of Computer Theory and Engineering, Vol. 2, No. 1, pp. 103-110, 2010.

[2] Uli Kretzschmar.," AES128 – A C Implementation for Encryption andDecryption", Texas Instruments, 2009.

[3] K. Tarun, N. Sukumar and B. Santosh, "A Single chip implementation of AES cipher and Whirlpool hash function", *Department of Computer Science and Engineering, Indian Institute of Technology*, 2009.

[4] C. Wang and H. M. Heys, "Using a Pipelined S-Box in Compact AES Hardware Implementations", *IEEE 8th International NEWCAS Conference (NEWCAS)*, Memorial University, pp. 101-104, 2010.

[5] J. Daernen and V. Rijrnen, "The Design of Rijndael", *Springer-Verlag Berlin Heidelberg*, 2002**.**

[6] S. Ghaznavi, C. Gebotys, and R. Elbaz, "Efficient Technique for the FPGA Implementation of the AES MixColumns Transformation", *IEEE International Conference on Reconfigurable Computing and FPGAs*, pp. 219-224, 2009.

[7] B. Jyrwa and R. Paily, "An Area-Throughput Efficient FPGA implementation of Block Cipher AES algorithm", *IEEE International Conference on Advances in Computing, Control, and Telecommunication Technologies*, pp. 328-332, 2009.

[8] Y. Zhang and X. Wang, "Pipelined Implementation of AES Encryption Based on FPGA", *IEEE International Conference on Information Theory and Information Security*, University of Jinan, pp. 170-173, 2010.

[9] A. A. Mohamed, and A. H. Madian, "A Modified Rijndael Algorithm and its Implementation using FPGA", *IEEE International Conference on Electronics, Circuits, and Systems*, pp. 335-338, 2010.