International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064

Low Power and Area Optimized VHDL Implementation of AES

Suja Chackochan[1], K.Mathan[2]

[1]Student (M.E VLSI Design & Embedded System), Dept of Electronics and Communication Engineering
Hindusthan Institute of Technology, Coimbatore – 641 032, Tamil Nadu, India

[2]Assistant Professor, Department of Electronics and Communication Engineering,
Hindusthan Institute of Technology, Coimbatore – 641 032, Tamil Nadu, India

Abstract: ***In this era of information, need for protection of data is more pronounced than ever. Secure communication is necessary to protect sensitive information in military and government institutions as well as private individuals. Current encryption standards are used to encrypt and protect data not only during transmission but storage as well. Data Encryption Standard was introduced in early 1970s as a standard cryptographic algorithm to protect data. However, due to its short 56-bit key length, simple brute force attacks cracked it in less than 10 hrs. Another disadvantage was also the possibility of weak and semi weak keys. In the year 2000, Rijndael Encryption algorithm or AES was chosen by National Institute of Standards and Technology (NIST) to be adopted by the U.S. Government as the new Encryption standard to replace the outdated and easily crackable DES. The major advantage lay in the non-linearity of the key schedule which eliminated the possibility of weak and semi weak keys. This encryption algorithm is virtually crack-proof till date. It is more computationally robust compared with previous algorithms and the security level is higher. However the execution time required is more because of long calculations and several iterations. The goal of this project is to study AES and improve the performance of this algorithm in terms of speed, area and power. In this paper the concept of pipelining for maintaining the speed of encryption is introduced. This design has been implemented in VHDL using Xilinx ISE 14.2i platform.***

Keywords: Encryption, Decryption, Pipelining, VHDL

1. Introduction

In cryptography, encryption is the process of encoding messages (or information). The unauthorized parties cannot read this message, but authorized parties can. In an encryption scheme, the message or information (referred to as plaintext) is encrypted using an encryption algorithm. The plaintext is converted to an unreadable ciphertext. The encryption is done with the use of a key. The key specifies how the message is to be encoded. Any unauthorized person who can see the ciphertext should not be able to determine anything about the original message. However, an authorized party is able to decode the ciphertext using a decryption algorithm that requires a decryption key that unauthorized people do not have access to.

The information security has aroused high attention with the rapid development and wide application of computer and communication networks. Information security is applied to the political, military & diplomatic and common fields of people's daily lives. The long-serving algorithm; DES with 56-bit key length has been broken because of the defect of short keys, with the continuous development of cryptographic techniques. The "Rijndael encryption algorithm" had been chosen as the standard AES (Advanced Encryption Standard) algorithm whose packet length is 128 bits and the key length is 128 bits, 192 bits, or256 bits.

Rijndael is invented by the Belgian cryptographers Joan Daemen and Vincent Rijmen. Since 2006, the Rijndael algorithm of advanced encryption standard has become one of the most popular algorithms in symmetric key encryption. AES can resist various currently known attacks. One new AES algorithm with 128-bit keys (AES-128) was described in this paper, which was realized in VHDL. The 128-bit plaintext, 128-bit key and the 128-bit output data were all divided into four 32-bit consecutive units. The new algorithm achieved a balance between encryption speed and chip area because of the pipelining technology which is used in the intermediate nine round transformations.

1.1 Background of the Algorithm

A proposal for the Advanced Encryption Standard, (AES) was solicited by the National Institute of Standards and Technology, (NIST). The AES is a Federal Information Processing Standard, (FIPS). Electronic data is protected by this cryptographic algorithm. This symmetric block cipher can encrypt, (encipher), and decrypt, (decipher), information. Encryption converts plain-text to cipher-text. Decryption of the cipher-text converts the data back into its original form. The AES algorithms can use128, 192, or 256 bits cryptographic key to encrypt and decrypt data in blocks of 128 bits. From twelve different nations many algorithms were originally presented by researchers. Fifteen, (15), algorithms were selected. After a study and selection process five, (5), were chosen as finalists. MARS, RC6, RIJNDAEL, SERPENT and TWOFISH were the five selected algorithms. The five Competitors had similar characteristics. On October 2nd 2000, NIST announced that the Rijndael Algorithm was the winner of the contest. The Rijndael Algorithm was chosen since it had the best overall scores in security, performance, efficiency, implementation ability and flexibility. The Rijndael algorithm was developed by Joan Daemen of Proton World International and Vincent Fijmen of Katholieke University at Leuven. The Rijndael algorithm is a symmetric block cipher that can process data blocks of 128 bits through the use of cipher keys with lengths of 128, 192, and 256 bits. The Rijndael algorithm was also designed to handle additional block sizes and key lengths. However, the additional features were not adopted in the AES.

2. Description of the Algorithm

The AES algorithm is a symmetric-key scheme. In symmetric key schemes, the keys of encryption and decryption are the same. Thus before they wish to communicate, communicating parties must agree on a secret key. It is also known as private-key scheme. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. AES operates on a 4X4 order matrix of bytes, which is known as state, although some versions of Rijndael have a larger block size and have additional columns in the state. The numbers of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext are specified by the key size. The number of cycles of repetition is as follows:

- 10 cycles for 128-bit keys.
- 12 cycles for 192-bit keys.
- 14 cycles for 256-bit keys.

Here, The algorithm is used in encrypting data which is stored in blocks of 128-bit (here) and is encrypted with a 128- bit cipher key which is essential to decrypt the information locked in the encrypted data. The Encryption program needs two pieces of inputs one, is the input data which is to be encrypted and the other being the Cipher key with which the information will be locked. The Algorithm is in turn divided into two distinct parts:

- Encryption
- Key Generation

2.1 Encryption Algorithm

The Encryption process contains 10 rounds and each round contains 4 different transformations at the end of the 10 rounds the encrypted values are produced. The Transformations are:

- Substitution Bytes
- Shift Rows
- Mix Columns
- Add Round Key

Each Round need not use all the transformations. The first round does only the $4t^h$ transformation while the last round that is round 10 does only 1^{st},2^{nd} &4^{th} transformations.

2.1.1Substitution Bytes

This is the first transformation to be done on the input data i.e., the input matrix. This step is also known as SubBytes. In the SubBytes step, each byte in the array is updated using an 8-bit substitution box, the Rijndael S-box. The nonlinearity in the cipher is provided by this operation. The S-box used is derived from the multiplicative inverse over Galois Field (2^8), known to have good non-linearity properties. The S-box is constructed by combining the inverse function with an invertible affine transformation to avoid attacks based on simple algebraic properties,. The Affine Transformation is used to generate the Sub Box which is the lookup table from which the values are substituted. The procedures to be followed to substitute the bytes in the matrix are:

- Select any element say 18 which is in hexadecimal notation.
- The S-box element in the 1^{st} row and 8^{th} column is to be selected and substituted in its place (i.e., ad).
- Similarly for other elements in the matrix.

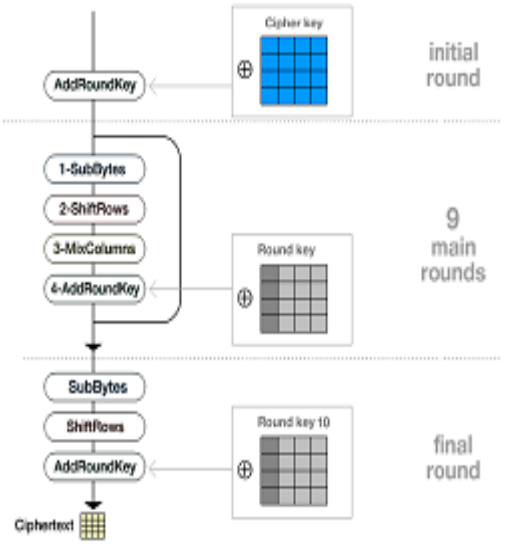

Figure 1: AES Encryption Process

2.1.2 Shift Rows

This is the Second Transformation in the series of 4 Transformations and is extremely simple to implement. It involves rotating the rows of the input matrix circularly upwards.

2.1.3 Mix Columns

This is probably the most complicated of the 4 Transformations in the Algorithm and involves the Modulo Matrix Multiplication of the input matrix with the following matrix;

02 03 01 01
01 02 03 01
01 01 02 03
03 01 01 02

Each column is multiplied with the matrix and the answer thus obtained will replace the input data and will go ahead to undergo further transformations in the consecutive rounds.

2.1.4 Add Round Key

This is the final transformation in which the Round Key which is specific to the particular round is added to the result of the above three transformations, Addition in this case is again modulo addition which is simply the bit-wise XOR of the participating values. Round Key in the case of the first Round is simply the Cipher Key and in the other Rounds is the Key generated specifically for that particular round using the Key Generation Algorithm. The Add Round key in the current situation in which we are only considering the first round thereby the round key is just the cipher key.

hex	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

x

Figure 2: Substitution Box (S-Box)

2.2 Key Generation Algorithm

There are a set of 11 Rounds in the Encryption process counting the initial round also, and every Round without fail follows the Add Round Key Transformation. Hence we need a total of 10 keys which are to be generated from the single key given as the input. Key schedule is the process of generating the required keys. The following are the steps involved in the process of generating the Keys required.

Write down the entire sequence of 11 keys we need a matrix which has 4 rows and 44 columns wherein the Round keys have to be calculated. Consider the Cipher key below:

2b 28 ab 09
7e ae f 7 c f
15 d2 15 4 f
16 a6 88 3c

The entire Sequence of keys when written out in a matrix form with the first column numbered 0 to last column being numbered 43 can be divided into two distinct groups i.e., multiples of 4 and non-multiples of 4.

2.2.1Generating the columns C4m

These columns are filled in a different way i.e., in the case of the Multiples of 4 i.e., (C4; C8; C12 ::::::::). The following is the algorithm followed: Consider the Column which is to be calculated then the ones just before it will be Cm 1 this column has to undergo two different transformations:

- Rotation circularly upwards
- Substitution of bytes in the rotated column
- Substitution of bytes in the rotated column. The 4*10 round constant matrix happens to be :

01 02 04 08 10 20 40 80 1b 36
00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00

- To the resulting column the column Cm 4 in the key generating Sequence has to be added the result is the value of the column Cm. This completes the generation of the 4m Column.

2.2.2 Generating the Columns C4m+1;4m+2;4m+3

The next set of columns to be generated in this case are the non multiples of 4 which are relatively simple to generate and the process is as follows: The column to be generated if considered Cn then selecting the earlier column Cn 1 and adding it to the column. Cn 4 will generate the rest of the columns. Following the same algorithm will generate the entire Key Sequence. Each of the key are used in the different Rounds in the corresponding Add Round Key Transformation in which the key will be added to the resulting matrix and the resulting answer is used as the input to the next Round.

2.3 Decryption Algorithm

The Decryption Algorithm follows the exact same transformations with their effect negated, the transformations are:

2.3.1 Inverse Substitution Bytes

In this step the value in the input is looked up at the S-Box and the corresponding Row and Column are recorded and substituted in the place of the input matrix to give out the result.

2.3.2 Inverse Mix Columns

In this step input matrix is to be Modulo multiplied with the matrix shown below.

14 11 13 09
09 14 11 13
13 09 14 11
11 13 09 14

Here the elements of the matrix are all in decimal notation and not in the traditional hexadecimal notation.

2.3.3 Inverse Shift Rows

This Transformation is again simply the inverse of its counterpart where in the input matrix was circularly rotated to the left, in this case the input matrix's rows are rotated to the right.

2.3.4 Inverse Add Round key

The Transformation as its counterpart is simply the bit wise addition of the input with the corresponding Round Keys. The only thing that sets this Transformation apart from its counter in Encryption process is that, here in the first Round we add to the input the last four Columns of the Key sequence which is different from the Add Key in Encryption process where in the first round would add the input to the Cipher Key. In this case though we add the Cipher Key to the final step i.e., before we end up getting the final decrypted value.

3. Pipelining

The pipelining between each of the 10 rounds will achieve a high performance encryption.. The data generated in each individual round is successively utilized as the input in the next round. There is a pipeline stage between each round, the design is fully pipelined.

4. Result and Conclusion

Optimized and Synthesizable VHDL code is developed for the implementation of both encryption and decryption process. The result shows that the chip area can be optimized effectively with the pipelining technology. Since the power consumption is directly related to the chip area, this design reduces power consumption to some extent. Delay is also minimized.

Table 1: Analytical Comparison

	Power (mW)	*Area (Slices/LUTs)*	*Delay (ns)*
Base paper	142.9	3802/204000	2.958
Modified paper	117.06	629/204000	1.859

5. Future Scope

The chip area and power can still be optimized because the S-box is implemented by look-up-table in this design. So the future work should focus on the S-box implementation mode.

References

[1] Alam.M, S.Ghosh, M.J.Mohan, D.Mukhopadhyay, D.R.Chowdhury, and I.S.Gupta, "Effect Of Glitches Against Masked AES S-Box Implementation And Countermeasure," IET Inf. Security, Vol. 3, No. 1, Pp. 34–44, Feb. 2009.
[2] Canright.D and L.Batina, "A Very Compact 'Perfectly Masked' S-Box For AES," In Proc. ACNS LNCS, 2008, Vol. 5037, Pp. 446–459.
[3] Carlet.C, L.Goubin, E.Prouff, M.Quisquater, And M.Rivain, "Higherorder Masking Schemes For S-Boxes," In Proc. FSE LNCS, 2012, Vol. 7549, Pp. 366–384.
[4] David Kenney, "Energy Efficiency Analysis And Implementation Of AES On An FPGA".
[5] Frank vater, Peter Langendorfer, "An area efficient realisation of AES for wireless devices".
[6] Gaj.K and P.Chodowiec, "Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays," in Proc. CT-RSA LNCS, 2001, vol. 2020, pp. 84–99.
[7] Goli´c J.D, "Techniques for random masking in hardware," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 54, no. 2, pp. 291–300, Feb. 2007.
[8] Goubin.L and J.Patarin, "DES and differential power analysis (the 'duplication' method)," in Proc. CHES LNCS, 1999, vol. 1717, pp. 158–172
[9] Kirat Pal Singh, Dilip Kumar, "Performance Evaluation of Low Power MIPS Crypto Processor based on Cryptography Algorithms", International Journal of Engineering Research and Applications, May-Jun 2012.
[10] Kocher P, J. Jaffe, and B. Jun, "Differential power analysis," in Proc.CRYPTO, 1999, vol. LNCS 1666, pp. 388–397.
[11] Yi Wang and Yajun Ha , "FPGA-Based 40.9-Gbits/s Masked AES With Area Optimization for Storage Area Network", ieee Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 60, No. 1, January 2013

Author Profile

K. Mathan is working as Assistant Professor, Electronics and Communication Department, Hindusthan Institute of Technology, Coimbatore, Tamilnadu.

Suja Chackochan is pursuing M Tech in VLSI Design and Embedded Systems from Hindusthan Institute of Technology.