ISSN 2348 – 9928
IJAICT Volume -1, Issue-3, July 2014 Doi:01.0401/ijaict.2014.01.11 Published Online 05 (05) 2014

DELAY OPTIMIZATION IN ADVANCED ENCRYPTION STANDARD ARCHITECTURE

Ms. M. Aswathi Mohan
PG Scholar,
Coimbatore Institute of Engineering and Technology,
Coimbatore, Tamilnadu, India.

Mr. M. Ramkumar Raja
Assistant Professor,
Coimbatore Institute of Engineering and Technology,
Coimbatore, Tamilnadu, India.

Abstract— ***A high speed security algorithm is always important for wired/wireless environment. The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data. This is called Rijndael. The algorithm described by AES is a symmetric-key algorithm that is the same key is used for both encrypting and decrypting the data. AES has the advantage of being implemented in both hardware and software. The AES hardware was implemented in three modules contains of the encryption, the decryption and the key scheduling module. The implementation of pipelined cryptography hardware is used to improve performance in order to achieve higher throughput and greater parallelism. Here the delay is been reduced ten times than that of normal AES architecture. Registers in pipelined structure can be replaced by low power register for power optimization. Xilinx design suite 14.5 is used for the implementation***

Keywords— Cryptography, AES, DES, Efficient encryption decryption implementation, Pipeline.

I. INTRODUCTION

National Institute of Standards and Technology (NIST) solicited proposals for the Advanced Encryption Standard, (AES). The AES is a Federal Information Processing Standard, (FIPS) that is used to protect confidential data. The algorithm should meet some of the requirements such as copyright free, faster than 3DES etc. After analysis Rijndael was chosen as the winner. The algorithm was developed by Joan Daemon of Proton World International and Vincent Fijmen of Katholieke University at Leuven.

The AES algorithm is a symmetric block cipher that can encrypt, (encipher), and decrypt, (decipher), information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. A new architecture was developed using pipelines. The implementation of pipelined AES was to improve the performance in order to achieve higher throughput and reduced delay.

II. AES RIJNDAEL

In order to understand the AES architecture in detail it is necessary to know definition of state in algorithm. State is the matrix of bytes that is processed between many stages or rounds and therefore it will be modified in each stage. In this algorithm the block size being used determines the matrix size. The key size is represented by Nk and block size is given as Nb.

Since the AES algorithm uses 128 bit blocks, the state will be composed by 4 rows and 4 columns. On the encryption algorithm, there will be 4 sub rounds: SubBytes, ShiftRows MixColumns, and AddRoundKey, Nevertheless, on the last stage, the MixColumns operation is avoided.

The decryption algorithm will use the respective inverse operations: InvSubBytes, InvShiftRows InvMixColumns and Inv AddRoundKey. As it was in the encryption phase, the InvMixColumns is avoided on the last stage of decryption process.

2.1 Sub Bytes

Each state byte is replaced by another in the S-box (replacement Box), as indicated in Fig 1. The replacement follows a matrix, where the first hexadecimal value corresponds to the row positioning, and the second hexadecimal value corresponds to the column positioning

ISSN 2348 – 9928
IJAICT Volume -1, Issue-3, July 2014 Doi:01.0401/ijaict.2014.01.11 Published Online 05 (05) 2014

shown in Fig 2 .The inverse operation (decryption) is called Inv Sub Bytes, and uses an inverse S-Box. Shown in Fig 3.

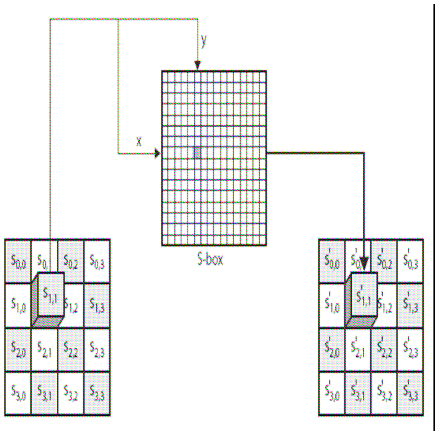

Fig 1: SubBytes operation process

x \ 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

Fig 2: S-Box

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

Fig 3: Inv S-Box

2.2 Shiftrows

It consists of a left shift on the state lines, replacing therefore their byte position, as indicated in Fig. 4. First line suffers no shifting. Second line is shifted by one position and Third line undergoes do 2 shifting positions. Fourth line is shifted by 3 positions respectively. Instead of line one we can also call it as line zero.

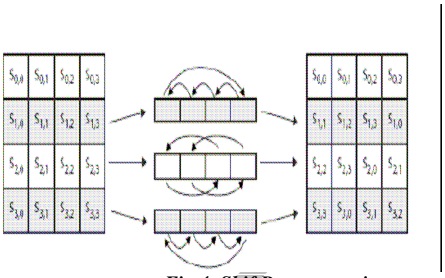

Fig 4: ShiftRows operation process.

The decryption algorithm performs the inverse operation InvShiftRows that consists of similar shifting as the ShiftRows, but shifted to the right.

2.3 Mixcolumns

In this operation, the state bytes are treated as polynomials of Galois Field algebra GF (2^8). The operation can be represented as a matrix multiplication, indicated in Fig. 5, where S is the initial state and S´ is the final state, after the operation.

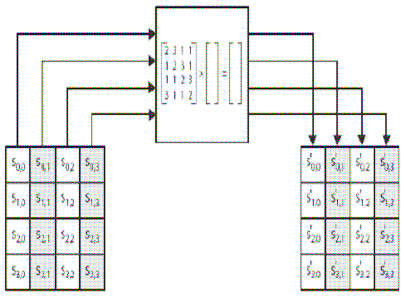

Figure 5: MixColumns operation process

In the inverse operation, the InvMixColumns, consists of the multiplication using the inverse matrix. In the last round, on both the encryption and decryption algorithms, the Mix Columns operation is avoided.

2.4 AddRoundKey

It is an XOR operation between the state and the round key that it is generated from the main key through the Key Generation process. The matrix of keys is represented by w columns. AddRoundKey is used both in the encryption and decryption algorithms. The XOR is conducted on byte basis, as indicated in Fig 6.

ISSN 2348 – 9928
IJAICT Volume -1, Issue-3, July 2014 Doi:01.0401/ijaict.2014.01.11 Published Online 05 (05) 2014

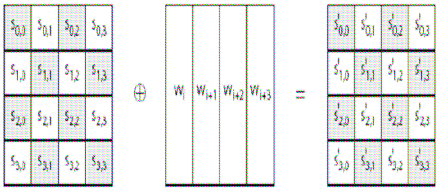

Fig 6: AddRoundKey operation process

2.5 Key Expansion

The Key size defines the number of rounds in the encryption/decryption algorithm, and it also defines its expansion process. Basically, the Key Expansion operation consists of three operations, as presented in Fig. 7. The first operation, RotWord, makes a one byte circular shifting on the word. The second operation, SubWord replaces each byte of the input word according to the S-Box. The third operation consists of XOR operations, as indicated in Fig 7.

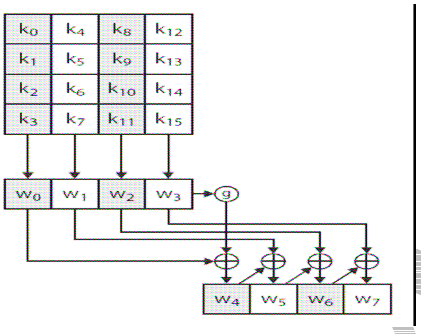

Fig 7: Key Expansion operation process

III. AES IMPLEMENTATION

The AES hardware was implemented in three modules: the encryption, the decryption and the key expansion module. The hardware is implemented as illustrated in Fig.8. It is composed of two 128 bit inputs that receive the key and the initial word to be encrypted. Each module was developed independently from the others, and then they were mounted together.

The process consists of 10 rounds for 128 bits data .Each round consists of Substitute bytes, Shift rows, Mix columns, Add round key blocks. After 10 rounds the obtained is the encrypted result which is called the cipher text. The encrypted value is decrypted in another 10 rounds which are reverse to encryption. The decryption cycle consists of 10 rounds .Each round consists of Inverse Shift Rows, Inverse Sub bytes, Add round key, Inverse mix columns. The AES algorithm were implemented using Xilinx Design suite 12.3

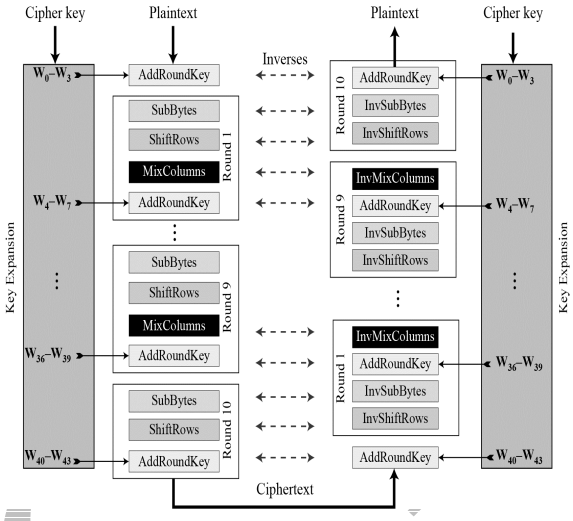

Fig 8: Block Diagram of AES Algorithm

IV. PIPELINED AES IMPLEMENTATION

Pipelining is one of the most efficient means of improving performance in high-end processor architectures in order to achieve higher throughput and also reduced delay. Here registers are added in between the ten rounds. The block diagram of pipelined AES is shown in Fig: 9.

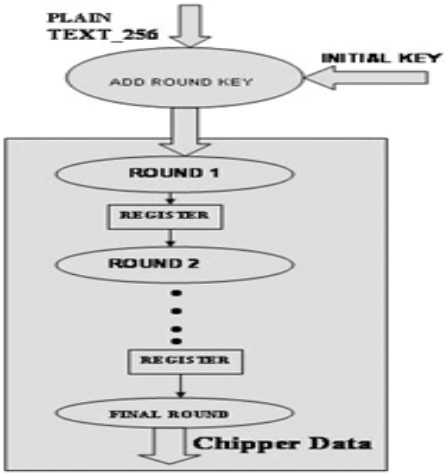

Fig 9: Block diagram of pipelined AES

The output from these rounds is saved in the respective registers in between the rounds. And the saved values in the registers are given as the input to the next round. Mainly there are two types of pipelining: outer round and inner round pipelining. Here outer round pipelining is used. Implementation of pipelined AES was also done using Xilinx design suite 14.5

V. RESULT ANALYSIS

Here the comparison of AES and pipelined AES architectures are done. Only delay of these architecture are been considered. The graph which gives the comparison is shown in Fig: 10. The delay report for the AES architecture shows about 93.791ns and that of the pipelined AES shows 9.624ns. Almost the delay is been reduced ten time in pipelined AES than in normal AES.

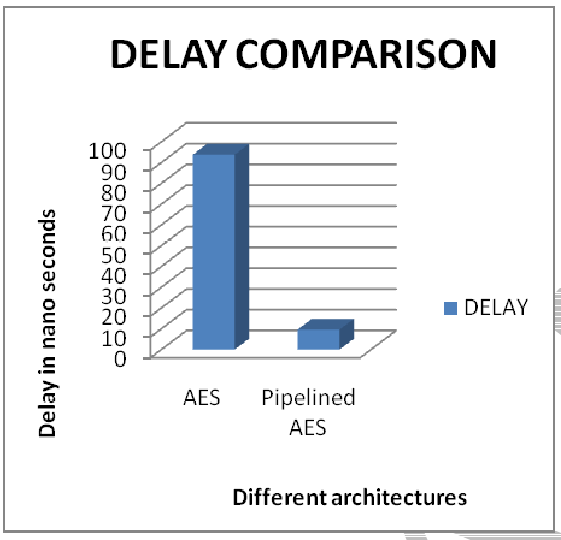

Fig 10: Delay Comparison

VI. CONCLUSION

In the above work the comparison of AES and pipelined AES architectures are done and delay is been compared. As a future work the power can also be compared. The pipelined AES architecture can be improved to low power AES architecture. This will further optimize the power in the entire architecture. The registers in the pipelined AES architecture can be replaced by low power registers.

References

[1] Chih-Pin Su, Tsung-Fu Lin, Chih-Tsun Huang, and Cheng-Wen Wu,National Tsing Hua University"A High-Throughput Low-Cost AES Processor"

[2] Sumanth Kumar Reddy S,SENSE ,VIT University ,Vellore, India and P. Praneeth ,SENSE ,VIT University ,Vellore, India and R.Sakthivel ,SENSE ,VIT University,Vellore, India "VLSI Implementation of AES Crypto Processor for High Throughput" (IJAEST) International Journal Of Advanced Engineering Sciences And Technologies Vol No. 6, Issue No. 1, 022 - 026 .

[3] Guido Bertoni, Luca Breveglieri, Israel Koren, Fellow, IEEE,Paolo Maistri, and Vincenzo Piuri, Fellow, IEEE "Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard" IEEE Transactions On computers, VOL. 52, NO. 4, APRIL 2003

[4] Stefan Mangard Student Member, IEEE, Manfred Aigner, and Sandra Dominikus "A Highly Regular and Scalable AES Hardware Architecture" IEEE Transactions On Computers, VOL. 52, NO. 4, April 2003,

[5] Adam J. Elbirt, W. Yip, B. Chetwynd, and C. Paar "An FPGA-Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists" IEEE Transactions On Very Large Scale Integration (VLSI) Systems, VOL. 9, NO. 4, AUGUST 2001

[6] Xinmiao Zhang, Student Member, IEEE, and Keshab K. Parhi, Fellow, IEEE "High-Speed VLSI Architectures for the AES Algorithm" IEEE Transactions On Very Large Scale Integration (VLSI) Systems, VOL. 12, NO. 9, September 2004

[7] N. Sklavos and O. Koufopavlou, Member, IEEE "Architectures and VLSI Implementations of the AES-Proposal Rijndael" IEEE Transactions On Computers, VOL. 51, NO. 12, December 2002

[8] Federal Information Processing Standards Publication 197 " ADVANCED ENCRYPTION STANDARD (AES)" November 26, 2001

[9] Paolo Maistri and Re´gis Leveugle, Member, IEEE "Double-Data-Rate Computation as a Countermeasure against Fault Analysis" IEEE TRANSACTIONS ON COMPUTERS, VOL. 57, NO. 11, NOVEMBER 2008

[10] Ramesh Karri, Kaijie Wu, Piyush Mishra, and Yongkook Kim "Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers" IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 21, NO. 12, DECEMBER 2002

[11] Ingrid Verbauwhede, Senior Member, IEEE, Patrick Schaumont, Student Member, IEEE, and Henry Kuo "Design and Performance Testing of a 2.29-GB/s Rijndael Processor" IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 38, NO. 3, MARCH 2003

[12] Alireza Hodjat, Student Member, IEEE, and Ingrid Verbauwhede, Senior Member, IEEE "Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s AES Processors" IEEE TRANSACTIONS ON COMPUTERS, VOL. 55, NO. 4, APRIL 2006