An efficient FPGA implementation of the AES Algorithm With Reduced Latency

R. Seelanand Kumar [1] V. Viswanadha [2]
[1]M-Tech (VLSI) student [2]Assoc. Professor
[1, 2] Siddharatha Institute of Engineering and Technology, India

Abstract— a proposed FPGA-based implementation of the Advanced Encryption Standard (AES) algorithm is presented in this paper. This implementation is compared with other works to show the efficiency. The design uses an iterative looping approach with block and key size of 128 bits, lookup table implementation of S-box. This gives low complexity architecture and easily achieves low latency as well as high throughput. Simulation results, performance results are presented and compared with previous reported designs.

I. INTRODUCTION

For a long time, the Data Encryption Standard (DES) was considered as a standard for the symmetric key encryption. DES has a key length of 56 bits. However, this key length is currently considered small and can easily be broken. For this reason, the National Institute of Standards and Technology (NIST) opened a formal call for algorithms in September 1997. A group of fifteen AES candidate algorithms were announced in August 1998. Next, all algorithms were subject to assessment process performed by various groups of cryptographic researchers all over the world. In August 2000, NIST selected five algorithms: Mars, RC6, Rijndael, Serpent and Two fish as the final competitors. These algorithms were subject to further analysis prior to the selection of the best algorithm for the AES. Finally, on October 2, 2000, NIST announced that the Rijndael algorithm was the winner. Rijndael can be specified with key and block sizes in any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits. Therefore, the problem of breaking the key becomes more difficult [1]. In cryptography, the AES is also known as Rijndael [2]. AES has a fixed block size of 128 bits and a key size of 128, 192 or 256 bits.

The AES algorithm can be efficiently implemented by hardware and software. Software implementations cost the smallest resources, but they offer a limited physical security and the slowest process. Besides, growing requirements for high speed, high volume secure communications combined with physical security, hardware implementation of cryptography takes place.

An FPGA implementation is an intermediate solution between general purpose processors (GPPs) and application specific integrated circuits (ASICs). It has advantages over both GPPs and ASICs. It provides a faster hardware solution than a GPP. Also, it has a wider applicability than ASICs since its configuring software makes use of the broad range of functionality supported by the reconfigurable device [3].

This paper deals with an FPGA implementation of an AES encryptor/decryptor using an iterative looping approach with block and key size of 128 bits. Besides, our design uses the lookup- table implementation of S-box. This method gives very low complexity architecture and is easily operated to achieve low latency as well as high throughput.

Organization of the rest of this paper is as follows. Section 2 provides a brief overview of AES algorithm. Design of AES based on FPGA implementation is presented in section 3. Section 4 gives simulation results followed by the comparisons with other works in section 5. Finally, section 6 gives the conclusion of this work.

II. DESCRIPTION OF AES ALGORITHM

The AES algorithm is a symmetric block cipher that can encrypt and decrypt 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 plain-text.

A. AES encryption

The AES algorithm operates on a 128-bit block of data and executed Nr - 1 loop times. A loop is called a round and the number of iterations of a loop, Nr, can be 10**,** 12, or 14 depending on the key length. The key length is 128, 192 or 256 bits in length respectively. The first and last rounds differ from other rounds in that there is an additional AddRoundKey transformation at the beginning of the first round and no MixColumns transformation is performed in the last round. In this paper, we use the key length of 128 bits (AES-128) as a model for general explanation. An outline of AES encryption is given in Fig. 1.

1) SubBytes Transformation

The SubBytes transformation is a non-linear byte substitution, operating on each of the state bytes independently. The SubBytes transformation is done using a once pre-calculated substitution table called S-box. That S-box table contains 256 numbers (from 0 to 255) and their corresponding resulting values. More details of the method of calculating the S-box table refers to [4]. In this design, we use a look-up table as shown in Table I. This is a more efficient method than directly implementing the multiplicative inverse operation followed by affine transformation.

This approach avoids complexity of hardware implementation and has the significant advantage of performing the S-box computation in a single clock cycle, thus reducing the latency.

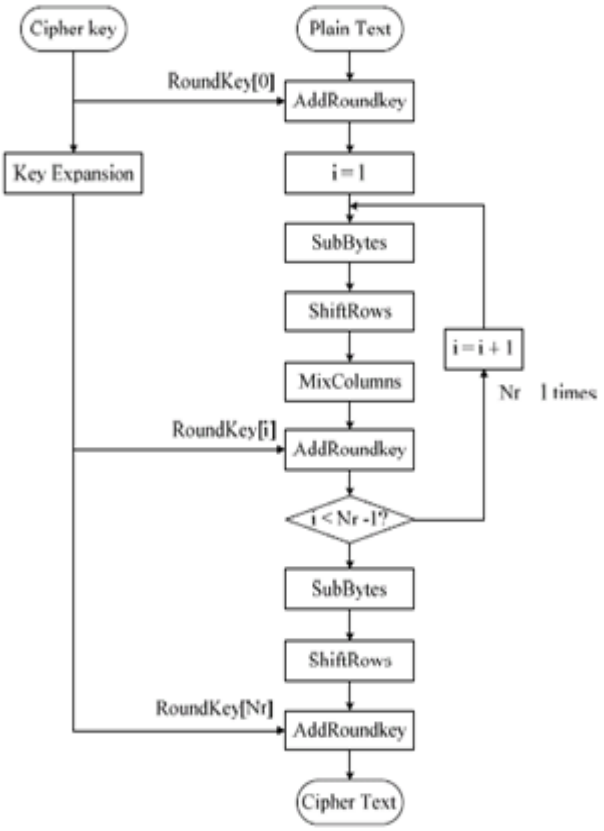

Fig. 1: AES encryption structure

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	1	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	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	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	6	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	8
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	3	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

Table. 1: S-box

2) ShiftRows Transformation

In ShiftRows transformation, the rows of the state are cyclically left shifted over different offsets. Row 0 is not shifted; row 1 is shifted one byte to the left; row 2 is shifted two bytes to the left and row 3 is shifted three bytes to the left.

3) MixColumns Transformation

In MixColumns transformation, the columns of the state are considered as polynomials over GF (28) and multiplied by modulo x4 + 1 with a fixed polynomial c(x), given by: c(x)={03}x3 + {01}x2 + {01}x + {02}.

4) AddRoundKey Transformation

In the AddRoundKey transformation, a Round Key is added to the State - resulted from the operation of the MixColumns transformation - by a simple bitwise XOR operation. The RoundKey of each round is derived from the main key using the KeyExpansion algorithm [1]. The encryption/decryption algorithm needs eleven 128-bit RoundKey, which are denoted RoundKey [0] RoundKey [10] (the first RoundKey [0] is the main key).

B. AES decryption

Decryption is a reverse of encryption which inverse round transformations to computes out the original plaintext of an encrypted cipher-text in reverse order. The round transformation of decryption uses the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes successively.

1) AddRoundKey

AddRoundKey is its own inverse function because the XOR function is its own inverse. The round keys have to be selected in reverse order. The description of the other transformations will be given as follows.

2) InvShiftRows Transformation

InvShiftRows exactly functions the same as ShiftRows, only in the opposite direction. The first row is not shifted, while the second, third and fourth rows are shifted right by one, two and three bytes respectively.

3) InvSubBytes transformation

The InvSubBytes transformation is done using a once-pre-calculated substitution table called InvS-box. That InvS-box table contains 256 numbers (from 0 to 255) and their corresponding values. InvS-box is presented in Table II.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	9	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	8	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	0	8c	bc	d3	0a	f7	e4	58	5	b8	b3	45	6
7	d0	2c	1e	8f	ca	3f	0f	2	c1	af	bd	3	1	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	1a
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	7	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	4	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Table. 2: InvS-box

4) InvMixColumns Transformation

In the InvMixColumns transformation, the polynomials of degree less than 4 over GF(28), which coefficients are the elements in the columns of the state, are multiplied modulo ($x4 + 1$) by a fixed polynomial $d(x) = \{0B\}x3 + \{0D\}x2 + \{09\}x + \{0E\}$, where $\{0B\}$, $\{0D\}$; $\{09\}$, $\{0E\}$ denote hexadecimal values.

In the next section, a description of the proposed design based on FPGA implementation of AES encryption/decryption function is detailed.

III. FPGA IMPLEMENTATION OF AES ALGORITHM

Fig.2 shows the detailed design of AES core based on FPGA implementation, where the control signals are described in Table III.

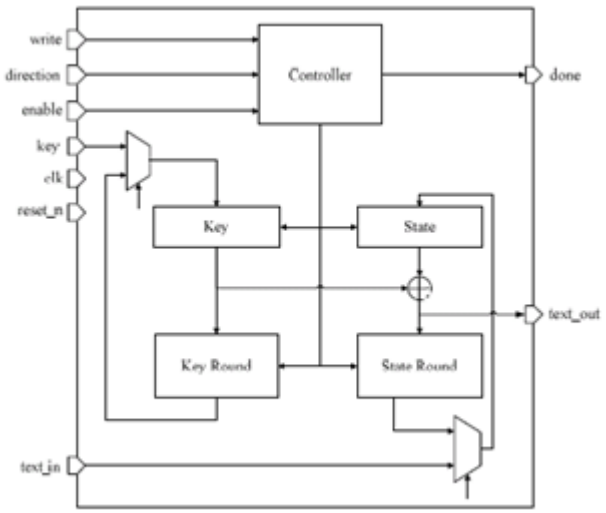

Fig. 2: Architecture of AES core

Pin name	I/O port	Pin number (bit)	Pin description
clk	I	1	Chip clock
reset_n	I	1	Clear all signal and data
write	I	1	1: Write key and text_in
direction	I	1	1: Encryption 0: Decryption
enable	I	1	1: Enable AES core 0: Disable AES core
key	I	128	Key data
text_in	I	128	Plaintext/Ciphertext data
done	O	1	1:Encryption/Decryption is completed
text_out	O	128	Ciphertext/ Plaintext data

Table. 3: Control Signals of AES cores

The total design has 390 pins. It requires the text_in, text_out and key which have a 128 bits length. And the control signals using to control the proper operations of the core are clk, reset_n, and write, direction, done and enable pins.

The Key block loads keys and combines with Key Round block to perform Key Expansion transformation, and generates proper Roundkeys under the control signals from the Controller block. Controller block takes write signal, direction signal, and enable signal from outside and generates all the control signals for the whole system.

The plain text (text_in) and key is loaded only when the write signal makes a low-high-low transition (basically a pulse). The process is going to complete when the done signal is pulsed after some clock cycles from the write signal goes low. The "done" signal actives only in one clock cycle.

Each round key as well as round is completed in one clock cycle. However, the round key is finished before the round is calculated by one clock cycle. Hence, combining with one clock cycle for registering the input, a total clock cycle need for processing 128-bit data is 13 clocks in encryption mode. In decryption, eleven round keys must be completed before the first round is calculated. Because the last round key is used in the first round process, it takes 25 clock cycles to complete.

With using the above iterative looping approach, a minimal number of clock cycles required performing encryption/decryption for each data block of 128-bit.

IV. SIMULATION RESULTS

The design has been coded by Verilog HDL. All the results are synthesized and simulated basing on the Quatus 9.0, the Model Sim – Altera 6.4a and EP20K400CB652C7 device.

The results of simulating the encryption/decryption algorithm from the ModelSim simulator are shown in Fig.3 and Fig.4.

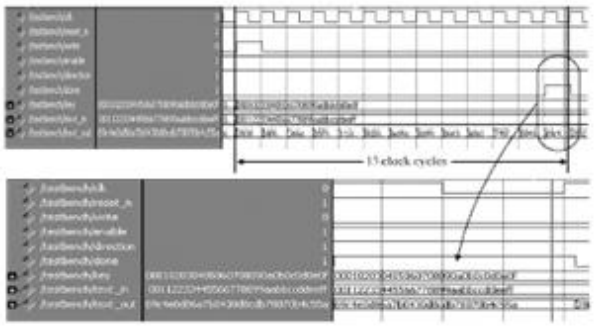

Fig. 3: Timing simulation of AES encryption algorithm

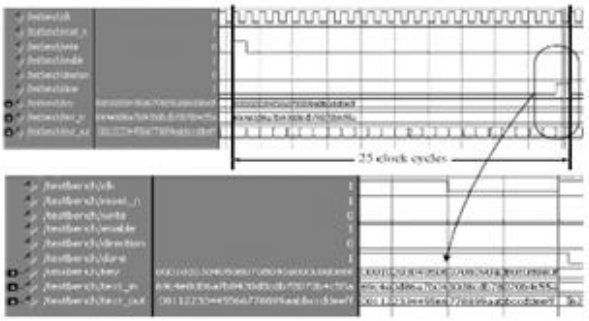

Fig. 4: Timing simulation of AES decryption algorithm

They are showing a low latency. Hence, the practical results are in accordance to theoretical predictions and satisfy the encryption and decryption methodology. To test the system, a test bench is used. The test bench applies encryption/decryption input pulse to trigger the system. The output result of the encryption was found accurately after 13 clock cycles from the starting of encryption process. So the latency of encryption is only 13 clock cycles. Similarly, the latency of decryption is 25 clock cycles.

V. COMPARISIONS

In this section, the results obtained by our design, and comparison between our results and other equivalent implementations is given and discussed.

Our design for AES 128-bit encryption/decryption algorithm was synthesized, implemented by Altera tools. Table IV summarizes the hardware resources required by main building blocks and gives detailed comparisons with the other designs [5], [6]. Considering the comparison in

table IV, our design is found to be more efficient in terms of latency, throughput and area. Therefore it allows us to process data in communication applications requiring a high security communication with low latency, high throughput and small area. Besides, the design is compared with another implementation using Xilinx chip [6] which uses the similar architecture with our design, but it requires a higher latency. Because Altera and Xilinx have the different chip architectures, comparison between us and [6] cannot be done in the other criteria of memory, throughput and area.

The design is tested with the sample vectors provided by FIPS 197 [2]. The algorithm achieves a low latency and the throughput reaches the value of 1054Mbit/sec for encryption and 615 Mbit/sec for decryption.

Designs		Our design	[5]	[5]	[6]
FPGA Vendor		Altera	Altera	Altera	Xilinx
FPGA Chip		APEX20KC	APEX20KC (Mixed Version)	APEX20K (Mixed Version)	XCV600
Memory	Encryption	40960	16384	16384	-
	Decryption	42368	16384	16384	-
Latency (Clock cycles)	Encryption	13	50	50	51
	Decryption	25	50	50	51
Throughput (Mbps)	Encryption	1188	240	187	-
	Decryption	442	184	128	-
Area (LCs)	Encryption	895	2231	1998	-
	Decryption	1098	2548	1999	-

Table. 4: Comparison in FPGA implementation of the AES Algorithm

VI. CONCLUSIONS

The Advanced Encryption Standard 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. An efficient FPGA implementation of 128 bit block and 128 bit key AES algorithm has been presented in this paper. The design is implemented on Altera using APEX20KC FPGA which is based on high performance architecture. The proposed design is implemented based on the iterative approach for cryptographic algorithms. Our architecture is found to be better in terms of latency, throughput as well as area. The design is tested with the sample vectors provided by FIPS 197 [2]. The algorithm achieves a low latency and the throughput reaches the value of 1054Mbit/sec for encryption and 615 Mbit/sec for decryption.

REFERENCE

[1] Hoang Trang, University of Technology, Vietnam National University HoChiMinh City, HoChiMinh City, Vietnam, 2012.
[2] Nguyen Van Loi, IC Design Research & Education Center (ICDREC), Vietnam National University HoChiMinh City, HoChiMinh City, Vietnam, 2012.
[3] Daemen J., and Rijmen V, "The Design of Rijndael: AES-the Advanced Encryption Standard", Springer-Verlag, 2002.
[4] FIPS 197, "Advanced Encryption Standard (AES)", November 26, 2001.
[5] Tessier, R., and Burleson, W., "Reconfigurable computing for digital signal processing: a survey", J.VLSI Signal Process, 2001, 28, (1-2), pp.7-27.
[6] Ahmad, N.; Hasan, R.; Jubadi, W.M; "Design of AES S-Box using combinational logic optimization", IEEE Symposium on Industrial Electronics & Applications (ISIEA), pp. 696-699, 2010.
[7] Alex Panato, Marcelo Barcelos, Ricardo Reis, "An IP of an Advanced Encryption Standard for Altera Devices", SBCCI 2002, pp. 197-202, Porto Alegre, Brazil, 9 and 14 September 2002.
[8] Mr. Atul M. Borkar, Dr. R. V. Kshirsagar and Mrs M. V. Vyawahare, "FPGA Implementation of AES Algorithm", International Conference on Electronics Computer Technology (ICECT), pp. 401-405, 2011 3rd.

