International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.03,Issue.35
November-2014,
Pages:6935-6938

www.ijsetr.com

Data Encryption and Transition by AES Algorithm with UART

CHUNCHU SADASHIVA[1], SANDEEP SUNKARI[2]
[1]PG Scholar, Prasad Engineering College, Telangana, India.
[2]Assoc Prof, Prasad Engineering College, Telangana, India.

Abstract: A proposed FPGA-based implementation of the Advanced Encryption Standard (AES) algorithm in UART Module is designed 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.

Keywords: AES, UART, FPGA, encryption, decryption, Rijndael, block cipher.

I. INTRODUCTION

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 Twofish 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, 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.

The UART protocol is a serial communication protocol that takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signaling levels. 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 in a UART module. 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.

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 MixCoulmns 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.

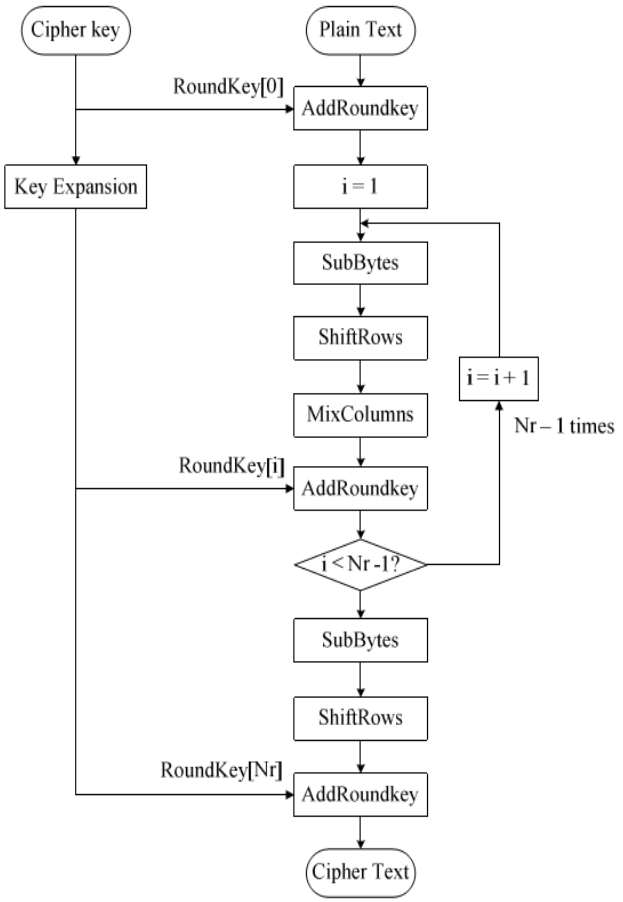

Figure1. AES encryption structure.

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.

TableI.

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

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.

TableII. Inv S-Box Table

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

3. InvSubBytes transformation:

The InvSubBytes transformation is done using a once-precalculated 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.

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.

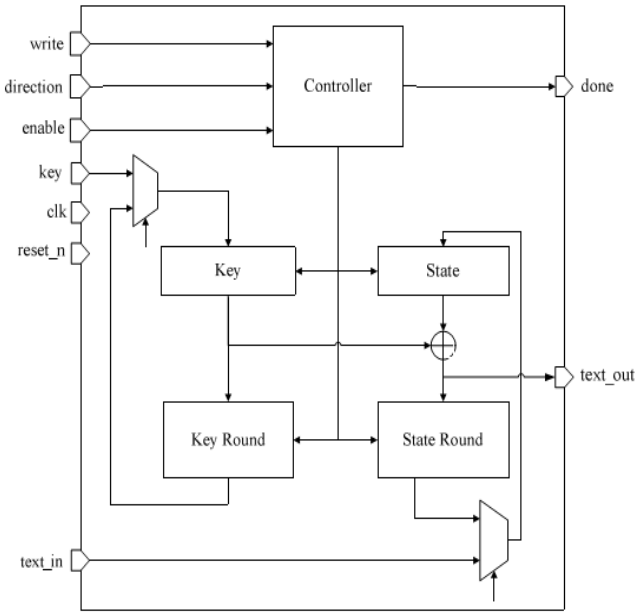

Figure 3: AES Opertional diagram

Table III. Control Signals 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

IV. SIMULATION RESULTS

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

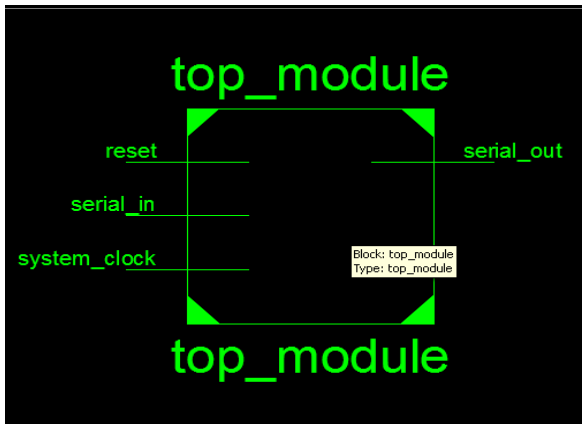

Figure4.Block diagram of AES encryption algorithm

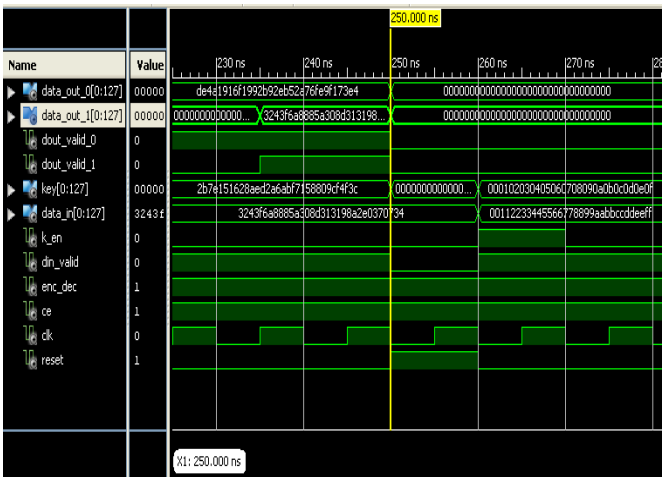

Figure5. Simulation of AES algorithm.

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. CONCLUSIONS

In this paper we have designed the UART Module using AES Algorithm for secure data transformation with the size of 128 bit block. 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. 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 The algorithm achieves a low latency and the throughput reaches the value of 1054Mbit/sec for encryption and 615 Mbit/sec for decryption.

VI. REFERENCES

[1]Daemen J., and Rijmen V, "The Design of Rijndael: AES-the Advanced Encryption Standard", Springer-Verlag, 2002

[2]FIPS 197, "Advanced Encryption Standard (AES)", November 26, 2001.

[3]Tessier, R., and Burleson, W., "Reconfigurable computing for digital signal processing: a survey", J.VLSI Signal Process., 2001, 28, (1-2), pp.7-27.

[4] 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.

[5] 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.

[6] 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.