A High Speed and Low Power Image Encryption with 128-Bit AES Algorithm

G. H. Karimian, B. Rashidi, and A. Farmani

Abstract—This paper presents implementation of a high-speed and low power encryption algorithm with high throughput for encrypting the image. Therefore, we select a highly secured symmetric key encryption algorithm AES (Advanced Encryption Standard), in order to increase the speed and throughput using pipeline technique in four stages, control unit based on logic gates, optimal design of multiplier blocks in mixcolumn phase and simultaneous production keys and rounds. Also for reducing power consumption using resource sharing, pipelining and signal gating. Such procedure makes AES suitable for fast image encryption. Implementation of a 128-bit AES on FPGA of Altera company has been done and the results are as follow: maximum frequency 475 MHz, power achieved is 301mw in clock frequency 100MHz. Power is analyzed using Xilinx Xpower analyzer. The time of encrypting in tested image with 32*32 size is 1.25ms.

Index Terms—Encryption standard (AES), pipelining, signal gating, image encryption, decryption

I. INTRODUCTION

Information is significant in every aspect of human life. Like any other property, it needs protection. There are different cryptographic algorithms available to secure information. However, most of them are computationally intensive, either deals with huge numbers and complex mathematics or involves several iterations. Advanced Encryption Standard (AES) is a cryptography algorithm proved to have the best quality among 15 candidates by National Institute of Standards and Technology (NIST). AES has high security with relatively little memory and CPU resource requirements. It is easier to apply cryptographic solutions on computer based communication systems than on conventional systems like telephone, fax and radios. It is not feasible to dedicate a general computer for each of such systems. Instead, a cheap and portable embedded system can be developed to ensure the communication security. Microcontroller, DSP, or ASIC are used in the construction of embedded systems. Microcontroller based embedded systems have lowest cost, which is one of the basic criteria of an embedded system design. Variety of microcontrollers available, each have different processor and peripheral devices inside them. ARM7TDMI is a popular embedded processor that has a lion's share of the market. It is reliable, that has low cost, low power consumption and small physical size [1]. AES is implemented in different ways, many of the

implementations are freely available [2][3]. However, these implementations do not run fast enough for real-time applications, like voice encryption. In such applications, the encryption has to be done in timely manner. Otherwise, it affects quality of service of the communication, in a way that it cannot be tolerated by the users. In this case, most developers go for a DSP or ASIC, which can run the available implementations faster so that it can meet the required speed. To encrypt the image in [4] add one key stream generator (A5/1, W7) to AES to ensure improving the encryption performance; mainly for images characterised by reduced entropy which has increased the AES security for the image encryption. In [5] used AES 32-bit for encryption of image. AES encryption is an efficient scheme for both hardware and software implementation, and FPGA is used for AES implementation. In most approaches, a RAM/ROM-based lookup table (LUT) is used, such as SubByte [6][7], and MixColumn [6] which operates on a 4-byte column and corresponds to multiplications and additions in GF (2^8). Addroundkey is simply performed by xoring each state with each key. In [7] MixColumn transformation is based on a chain of xor units. In [8] an architecture is used which speeds up the AES algorithm with no feedback by duplicating hardware for implementing each round unit. These approaches are based on pipelining, subpipelining and loop-unrolling. In [9] ShiftRow unit is implemented based on a 4-bit counter and two memories (ROMa, ROMb). In [7][12] the inner and outer pipelining and loop-unrolling has made it possible to achieve the throughput of 30 to 70 Gbps using 0.18 μ m CMOS technology. In [10] the implementation of S-BOX is based on Finite Field. In [11] use of only one S-BOX instead of four has made the hardware and area to be reduced but also the speed to be decreased by 4 times. The rest of the paper is structured as follow. Section2 gives a brief summary of AES algorithm and presents the system architecture adopted in our implementation. Comparison of our implementation with those done is given at section3. Finally section4 provides the conclusion of this paper.

II. AES ALGORITHM AND PROPOSED IMPLEMENTATION

The AES algorithm is a symmetric block cipher that processes data blocks of 128-bits using a cipher key of length 128, 192 or 256 bits each data block consist of a 4*4 array of bytes called the state, on which the basic operations of the AES algorithm are performed. The AES encryption procedure is shown in Fig.1.

The AES decryption procedure is shown in Fig.2. AES algorithm contain of two parts:

Manuscript received April 20, 2012; revised May 16, 2012.

The authors are with the Department of Electrical and Computer Engineering, University of Tabriz, IRAN (e-mail: karimian@tabrizu.ac.ir, bahram88@ms.tabrizu.ac.ir, ali_farmani88@ms.tabrizu.ac.ir)

Fig. 1. 128 bit encryption AES algorithm.

Fig. 2. 128 bit decryption AES algorithm

- Round function

This part consisting of different transformations: subBytes, shiftrows, mixColumns and addroundkey the four transformation are described briefly as follows[19]:

A. SubByte

Every byte in the state is replaced by another, using the Rijndael S-Box. It is a non-linear substitution that operates independently on each byte of the state using a substitution table (S-Box). The S-Box is invertible and is constructed by composition of two transformations. Namely, multiplicative inverse in finite field $GF(2^8)$ followed by affine transformation [14]. Calculating S-Box entries is computationally expensive, and its values are independent of the input. For most applications, S-Box values are pre-calculated and stored in a 16*16 byte (256 byte) memory. Each individual byte of state is mapped into a new byte in the following way: The left most 4 bits used as a row value and the right most 4 bits are used as a column value. These row

and column values serve as indexes into the S-Box to select a unique 8-bit output value as shown in the Fig.3. In proposed implementation, S-Box is based LUT as a way of increasing the speed. This implementation is shown in Fig.3.

Fig. 3. S-BOX transformation(LUT)

B. ShiftRow

Every row in the state is shifted a certain amount to the left. In this operation, each row of the state is cyclically shifted to the left, depending on the row index. The first row is not shifted, the second shifted 1 byte position, the third 2 byte and the fourth 3 byte position. A graphical representation of shiftrows and inverse shiftrows is shown Fig.4 and Fig.5.

Fig. 4. ShiftRows transformation

Fig. 5. Inverse ShiftRows transformation

Fig. 6. ShiftRows/inverse shiftrows

In proposed implementation, 16*8-bit registers have been used, and in the Verilog program, each output byte is placed in the position as it has to be after shift operation, making a 128-bit register which can also be used as one of the pipeline registers. Fig.6 shows the implemented shiftrow.

C. MixColumns

The data within each column of state are mixed. It operates on the state column wise, treating each column as a four term polynomial over $GF(2^8)$. The column polynomial is multiplied module x^4+1 with fixed polynomial, $p(x)$ giving by

$$p(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

The transformation can be defined by the following matrix multiplication on state (Fig. 7):

Fig. 7. Mixcolumns transformation

Multiplication of a value by x (i.e., by $\{02\}$ and $\{03\}$) can be implemented as a 1-bit left shift followed by a conditional bitwise xor with $(0001\ 1011)$ if the leftmost bit of original value is 1. ($\wedge$ stands for Exclusive OR) for example, we have: $\{02\} * \{87\} = (00001110) \wedge (00011011) = (0001\ 0101)$ and $\{03\} * \{6E\} = \{6E\} \wedge (\{02\} * \{6E\}) = (01101110) \wedge (11011100) = (10110010)$.

Proposed implementation is based on multiplication by 2 and 3 (multi2, multi3) and xor operation and these two multiplications have been written as a function. In Fig. 8 shows our mixcolumn implementation. The equation of multiplication of each row by each column has been fully pre-calculated; therefore, the operations are only based on shift and xor, and this has resulted in an increase in speed of this transformation. implementation of mixcolumn is shown in Fig. 8 InvMixcolumns is the inverse of the mixcolumns transformation. Proposed implementation of inverse mixcolumn shown in Fig. 9 InvMixcolumns operates on the State column-by-column. The InvMixcolumn can be written as matrix multiplication shown in below:

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{pmatrix} = \begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{pmatrix}$$

Fig. 8. Proposed implementation mixcolumn base of multi3 and multi2

Fig. 9. Proposed implementation inverse mixcolumn base of multi2

D. AddRoundkey

A round key is added to a state. In this operation round key is applied to the state by a simple bit wise XOR. The round key is extracted from the cipher key by means of key schedule. The operation is viewed as a column wise operation between the 4byte of a state column and word of the round key, it can also be viewed as a byte-level operation.

1) Key Expansion

In Fig. 10 shown AES key expansion. That explanation according to Fig. 10 in next paragraph.

Fig. 10. AES key expansion

The function g consist of the following sub functions:

1. Rotword performance a one-byte circular left shift on a word. this means that an input word $[b_0, b_1, b_2, b_3]$ is transformed into $[b_1, b_2, b_3, b_0]$.
2. subword performs a byte substitution on each byte of its input word using the S-box.
3. The result of steps 1 and 2 is xored with a round constant shown in Table I.

TABLE I: THE VALUE $RC[j]$ IN HEXADECIMAL

j	1	2	3	4	5	6	7	8	9	10
$RC[j]$	01	02	04	08	10	20	40	80	1B	36

Proposed implementation of the key expansion non-pipelining is shown in Fig. 11 and Fig. 12

Fig. 11. Rotword (rotate)

Fig. 12. Key expansion non-pipelining

Inverse key expansion non-pipelining is shown in Fig.13 also We see Inverse R-con shown in Table II:

TABLE II. INVERSE R-CON

j	1	2	3	4	5	6	7	8	9	10
RC[J]	36	18	80	40	20	10	08	04	02	01

Fig. 13. Inverse key expansion non-pipelining

For implement key expansion, pipelining technique has been used and its control unit has been implemented using logic gates. These two factors lead to an increase in speed and throughput of the unit, and it is controlled in the way that with each state, a key is generated; this means that the steps of data shifting in key expansion and round are done simultaneously. Finally, each key is xored with its corresponding round. The use of 4 stage pipelining, and control unit based on logic gates, design of MixColumn unit based on multiplications by 2 and 3, hardware implementation of multiplication of each row by each column, and simultaneous generation of each key and each round have made this implementation to be high in terms of speed and throughput. In the control unit of AES algorithm, the final result generated in the end of each round and key expansion should be produced simultaneously, in order to prevent happening a cycle difference and subsequently no wrong numbers is produced at this stage. Therefore, the control unit of this design is such that control signals of 4 performed operations in each round and also their multiplexers are simultaneous and done step-by-step with the performed operations in each phase of key production in key expansion, in order to ensure the synchronization and speed increase and that the key production takes place just in the last phase. The number is just produced in the last phase of round and to achieve this, control signals have been ordered and arranged accordingly. In other words, because of using 4 stage pipelining both in key expansion unit and in each round, and provided that their control signals are defined correctly, a perfect harmony will be created between these stages. More over, this unit is implemented by logic gates which again cause the production speed of each round and key to be increases. And for reduce power consumption using of pipelining and signal gating or enable/select signal that pipelining shortens the depth of combinatorial logic by inserting pipeline registers also pipelining is very effective for data path elements such as parity trees and multipliers. Enable/select signal prevents the propagation of their switching activity. Therefore power consumption is reduced. Fig.14 and Fig.15 shows the proposed implemented encryption and decryption algorithm.

Fig. 14. Proposed implementation of AES encryption

Fig. 15. Proposed implementation of AES decryption

Block of k -to- w is register that get 128bit input and output include w_1, w_2, w_3, w_4 that each are 32 bits. The original image can be regenerated using the encrypted image and the final key produced at the last stage of encryption by the image decryption circuit which is implemented too. In this implementation, the image used is of 32*32 size; the Hex codes of the image is given to the designed AES encrypting, and encrypted data of the original image, that is the encrypted image, are obtained. The time needed to generate the encrypted image is 1.25ms which is very shorter of [5]. Fig.16 shows the original image and the encrypted image obtained by this implementation. The histogram of the original and encrypted images shown in Fig.17 We can see that the histogram of the ciphered image is fairly uniform and is significantly different from that of the original image. Therefore, it does not provide any indication to employ any statistical attack on the image under consideration.

Fig. 16. Original image and the encrypted image

Fig. 17. Histogram of the original and encrypted image.

III. COMPARISONS

This design is accomplished via Verilog HDL hardware description language by QuartusII9.0 software simulated with MATLAB, and finally implemented on FPGA in Stratix-II family. power is analyzed to Xilinx Xpower. This design has a high speed, high throughput and low power consumption. It is really suitable for highly secured image encryption; and also the time of its converting is low. Table III shows the comparison between frequency, throughput, numbers of register and devices and the type of device that has been used in different articles and in Table IV the characteristic of our encryption image has been shown.

TABLE III. COMPARE IMPLEMENTATION OF DIFFERENT AES ALGORITHM

implementation	Device	Frequency (Mhz)	Throughput (Mbps)	Nbr of registers
M.zeghid[4]	-----	129	1651	-----
I.thulasimani [19]	Xc2v600bf 957-6	-----	666.7	2943
Parhi[18]	xc2vp30	150.5	221.4	536
Wang[15]	-----	125.38	1604	395
f.burns[16]	-----	132	156	4800
Chang[9]	Spartan-3xc3s200	287	647	148
Cheng[13]	Vertex-2pxc2vp22 73	273	749	104
Elkeelanv[14]	Single core	-----	12.6	1475

Our measurement result of implementation image encryption by AES is shown in Table 4 and power consumption is shown in Table V.

TABLE IV: RESULT OF PROPOSED IMPLEMENTATION AND [5] OF IMAGE ENCRYPTION BY AES

Implementa tion	Device	Frequenc y (Mhz)	Throug hput(M bps)	Nbr of register s	Encryption time(ms)
Proposed method	Stratix II	475	617	808	1.25 (32*32)
Kuo-huang [5]	Vertex-2 pxc2vp2	273	749	104	8243 (120*160)

TABLE V: POWER CONSUMPTION

Clock Frequency	Proposed method	[20]
25MHZ	109mw	885mw
100MHZ	301mw	---

IV. CONCLUSION

In this paper, hardware implementation of AES algorithm is used to encrypt the image. For increase speed applying 4 pipeline stages, designing the control unit based on logical gates, implementation of mixcolumn and inv-mixcolumn by only with mult 2 and mult 3 units and synchronizing the key production phase with each round phase, also for reducing power consumption using techniques include: resource sharing, pipelining and signal gating. This algorithm has been improved in terms of hardware on FPGA and is appropriate for encrypting an image in a short time and high security than other works.

REFERENCES

- [1] N. Sloss, D. Symes, and C. Wright, "ARM System Developer's Guide, Designing and Optimizing System Software," Morgan Kaufmann, 2004.
- [2] B. Gladman, "A specification for Rijndael, the AES Algorithm," Available at <http://fp.gladman.plus.com>, May 2002.
- [3] XYSSL Crypto Library, GNU Lesser General Public License, 2003.
- [4] M. Zeghid, M. Machhout, L. Khriji, A. Baganne, and R. Tourki, "A Modified AES Based Algorithm for Image Encryption," *World Academy of Science, Engineering and Technology*, 2007.
- [5] Y. Cheng, C. C. C. Hsieh, C. W. Huang, and C. J. Chang, "Embedded a Low Area 32-bit AES for Image Encryption/Decryption Application," *Circuits and Systems, ISCAS 2009. IEEE International Symposium on*, pp. 1922–1925, 2009.
- [6] S. S. Wang and W. S. Ni, "An efficient fpga implementation of advanced encryption standard algorithm," *Circuits and Systems, ISCAS '04. Proceedings of the 2004 International Symposium on*, pp. 597–600, vol. 2, 2004.
- [7] A. Hodjat, D. Hwang, B. Lai, and I. Verbauwhede, "A 3.84 gbits/s AES crypto coprocessor with modes of operation in a0.18um cmos technology," *GLSVLSI'05*, pp. 17–19, 2005.
- [8] X. Zhang and K. K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm," *IEEE Transaction Sonvery Large Scale Integration (Vlsi) Systems*, vol. 12, no. 9, 2004.
- [9] C. J. Chang, C. W. Husang, H. Y. Tai, M. Y. Lin, and T. K. Hu, "8-bitAES FPGA Implementation using Block RAM," *The 33 Annual Conference of the IEEE Industrial Electronics Society(IECON)*, pp. 5–8, 2007.
- [10] C. P. Su, T. F. Lin, C. T. Huang, and C. W. Wu, "A High-Throughput Low-Cost AES Processor," *IEEE Communications Magazine December*, 2003.
- [11] N. Yu and H. M. Heys, "Investigation of Compact Hardware Implementation of the Advanced Encryption Standard," *2005 IEEE CCECE/CCGEI*, Saskatoon, May 2005.
- [12] A. Hodjat, and I. Verbauwhede, "Area-Throughput Trade-Off for Fully Pipelined 30to70Gbits/s AES Processors," *IEEE Trans Actionson Computers*, vol. 55, no. 4, 2006.
- [13] C. J. Chang, C. W. Huang, K. H. Chang, Y. C. Chen, and C. C. Hsieh, "High Throughput 32-bit AES Implementation in FPGA," *IEEE Asia Pacific Conference On Circuits And Systems*, December 2008, pp.1806–1809, 2008.
- [14] J. Yenuguvanilanka and O. Elkeelany "Performance Evaluation of Hardware Models of Advanced Encryption Standard (AES) Algorithm," *Southeastcon, 2008 IEEE*, pp.222–225, 2008.

- [15] D. Wang and X. Li, "Improved Method to Increase AES system Speed," *The Ninth International Conference on Electronic Measurement and Instruments, ICEMI' 2009*, pp.49-52, 2009.
- [16] F. Burns, J. Murphy, A. Koelmans, and A. Yakovlev, "Efficient advanced encryption standard Implementation using lookup and normal basis," *Published in IET Computers and Digital Techniques*, IET Comput. Digit. Tech, vol. 3, pp. 270–280, and 2009.
- [17] L. Thulasimani and M. Madheswaran "A SINGLE CHIP DESIGN AND IMPLEMENTATION OF AES 128/192/256 ENCRYPTION ALGORITHMS," *International Journal of Engineering Science and Technology*, vol. 2, no. 5, pp.1052-1059, 2010.
- [18] X. Zhan and K. K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm," *IEEE Transaction Son Very Large Scale Integration (Vlsi) Systems*, vol. 12, no. 9, 2004.
- [19] D. Atnafu, "Optimizing AES Implementation for High-Speed Embedded Application," Feb 2008 Addis Ababa.
- [20] R. Banu and T. vladimirova, "fault-tolerant encryption for space applications," *IEEE transactions on aerospace and electronic systems*, vol. 45, no. 1, 2009.

Ghader Karimian, was born in 1975 in Khosrowshar, Iran. He received his B.S. and M.Sc. degrees in Electrical Engineering from Tabriz University and Amirkabir University of Technology in 1998 and 2000, respectively. He received Ph.D. degree in Electrical Engineering from Amirkabir University of Technology in 2006. He is now a assistant professor in the Faculty of Electrical and Computer engineering at University of Tabriz. His

research interests include Robot vision, Target tracking, Image processing, Digital signal processing and Hardware Design.

Bahram Rashidi, was born in 1986 in Boroujerd-Lorestan, Iran. He received his B.SC. degree in Electrical Engineering from the Lorestan university, Iran, in 2009 and he is now M.SC. in the Tabriz university, Iran, respectively. He got place in his M.SC. program. He has published 4 refereed conference papers and also submitted 7 papers to different journal. His research interests include digital signal processing, DSP processors and computer vision. He is also interested in the field of Digital programming include VHDL and VERILOG.

Ali Farmani, was born in 1985 in Tehran, Iran. He received his B.SC. degree in Electrical Engineering from the Shiraz university, Iran, in 2008 and he is now M.SC. in the Tabriz university, Iran, respectively. He has published 2 refereed conference papers. His research interests include digital signal processing, DSP processors and computer vision. He is also interested in the field of Digital programming include VHDL and

VERILOG