International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.02, Issue.11,
December-2014,
Pages:1124-1128

www.ijvdcs.org

High Performance Hardware Implementation of AES using Minimal Resources

DARAM MALAKONDA[1], KANDULA RAVI KUMAR[2], ANANDA BABU BATTU[3]

[1]PG Scholar, Dept of ECE, Rao & Naidu Engineering College, Ongole, AP, India, Email: darammalakonda@gmail.com.
[2]Professor, Dept of ECE, Rao & Naidu Engineering College, Ongole, AP, India, Email: ravi5_kumar@yahoo.com.
[3]Assoc Prof, Dept of ECE, Rao & Naidu Engineering College, Ongole, AP, India, Email: anand.rnec@gmail.com.

Abstract: Increasing need of data protection in computer networks led to the development of several cryptographic algorithms hence sending data securely over a transmission link is critically important in many applications. Hardware implementation of cryptographic algorithms are physically secure than software implementations since outside attackers cannot modify them. In order to achieve higher performance in today's heavily loaded communication networks, hardware implementation is a wise choice in terms of better speed and reliability. This paper presents the hardware implementation of Advanced Encryption Standard (AES) algorithm using Xilinx–virtex5 Field Programmable Gate Array (FPGA). In order to achieve higher speed and lesser area, Sub Byte operation, Inverse Sub Byte operation, Mix Column operation and Inverse Mix Column operations are designed as Look Up Tables (LUTs) and Read Only Memories (ROMs). This approach gives a throughput of 3.74Gbps utilizing only 1% of total slices in xc5vlx110t-3- ff1136 target device.

Keywords: AES, Rijndael, Cryptography, FPGA, Verilog, Encryption, Decryption.

I. INTRODUCTION

Cryptography allows people to carry over the confidence found in the physical world to the electronic world. The importance of cryptography is constantly increasing since the amount of sensitive data being transmitted over an open environment is also increasing day by day. The more information that is transmitted in computer-readable form, the more vulnerable we become to automated spying. Cryptography is not only important in defense applications but also important in real world applications such as E-commerce, E-mail etc.Encryption is usually done just before sending data. To utilize the channel resources completely encryption algorithm must have a speed at least equivalent to data transmission speed. Achieving high throughput for encryption algorithm for a communication channel of high data rate is a challenging task. The hardware (FPGAs and Application Specific Integrated Circuits-ASICs) implementation of such algorithm which meets these requirements is done in the present work. FPGAs are chosen considering several advantages over the other counterpart [2].The AES was published by National Institute of Standards and Technology (NIST) in 2001. Later Rijndael algorithm was selected as AES algorithm. Rijndael algorithm can have key length of 128, 192 and 256 bits while block size must be 128 bit [3].

There are many architecture proposals for AES Rijndael algorithm [4, 5], but many of them are poor in terms of area and speed. This paper proposes a different approach to increase speed by utilizing lesser resources available in FPGA. This paper is structured as follows: Section II Advanced Encryption Standard and Section III Parallel-Mix Columns. Section IV AES Modes of Operation The result and conclusion are described in Section V and VI respectively.

II. ADVANCED ENCRYPTION STANDARD

AES is a symmetric encryption algorithm, and it takes a 128-bit data block as input and performs several rounds of transformations to generate output cipher text. Each 128-bit data block is processed in a 4-by-4 array of bytes, called the state. The round key size can be 128, 192 or 256 bits. The number of rounds repeated in the AES, Nr, is defined by the length of the round key, which is 10, 12 or 14 for key lengths of 128, 192 or 256 bits, respectively.Fig.1. Block diagram of AES encryption for encryption, there are four basic transformations applied as follows:

A. Sub Bytes

The Sub Bytes operation is a nonlinear byte substitution. Each byte from the input state is replaced by another byte according to the substitution box (called the S-box). The S-box is computed based on a multiplicative inverse in the finite field GF (28) and a bitwise affine transformation.

B. Shift rows

In the Shift Rows transformation, the first row of the state array remains unchanged. The bytes in the second, third, and forth rows are cyclically shifted by one, two, and three bytes to the left, respectively.

C. Mix columns

During the Mix Columns process, each column of the state array is considered as a polynomial over GF (28). After multiplying modulo x4+1with a fixed polynomial a(x), given by a(x) = {03} x3+ {01} x2+ {01} x+ {02} the result is the corresponding column of the output state.

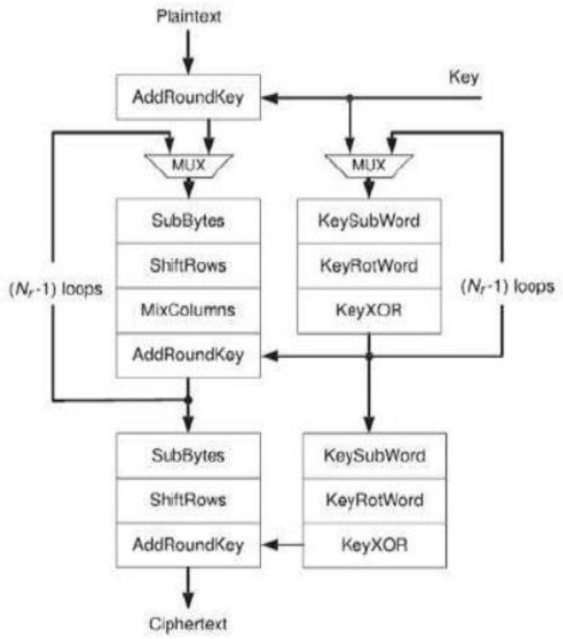

Fig1. Block diagram of AES encryption.

D. Addroundkey

A round key is added to the state array using a bitwise exclusive-or (XOR) operation. Round keys are calculated in the key expansion process. If Round keys are calculated on the fly for each data block, it is called AES with online key expansion. On the other hand, for most applications, the encryption keys do not change as frequently as data. As a result, round keys can be calculated before the encryption process, and kept constant for a period of time in local memory or registers. This is called AES with offline key expansion. In this paper, both the online and offline key expansion AES algorithms are examined. Similarly, there are three steps in each key expansion round.

E. Key sub word

The Key Sub Word operation takes a four byte input word and produces an output word by substituting each byte in the input to another byte According to the S-box.

F. Key rot word

The function Key Rot Word takes a word [a3; a2; a1; a0], performs a cyclic permutation, and returns the word [a2; a1; a0; a3] as output.

G. Key xor

Every word w[i] is equal to the XOR of the previous word, w [i-1], and the word Nk positions earlier, w [i- Nk]. Nk equals 4, 6 or 8 for the key lengths of 128, 192 or 256 bits, respectively. The decryption algorithm applies the inverse transformations in the same manner as the encipherment. As a result, we only consider the encryption algorithm in this work for simplicity, since the decipherment yields very similar results.

III. PARALLEL-MIX COLUMNS

Besides loop unrolling, another way to increase the through put of the OTOP model is to reduce the main loop's latency in the AES algorithm. In a single loop, the execution delay of Mix Columns- 16 results in 60 percent of the total latency. Each Mix Columns-16 operates on a four-column data block, and the operation on each column is in dependent. Therefore, each Mix Columns-16 processor can be replaced by four Mix Columns- 4s.Each MixColumns-4 actor computes only one column rather than a whole data block. As a result, the through put of the Parallel-Mix Columns implementation is increased to 2,180 cycles per block, equaling 136.25 cycles per byte. The data flow diagram and mapping of the Parallel-Mix Columns model are shown in Figs.2a and 2b. Each core on our targeted computational plat form can only support two statically configured input ports. Three cores, each called Merge Core, are

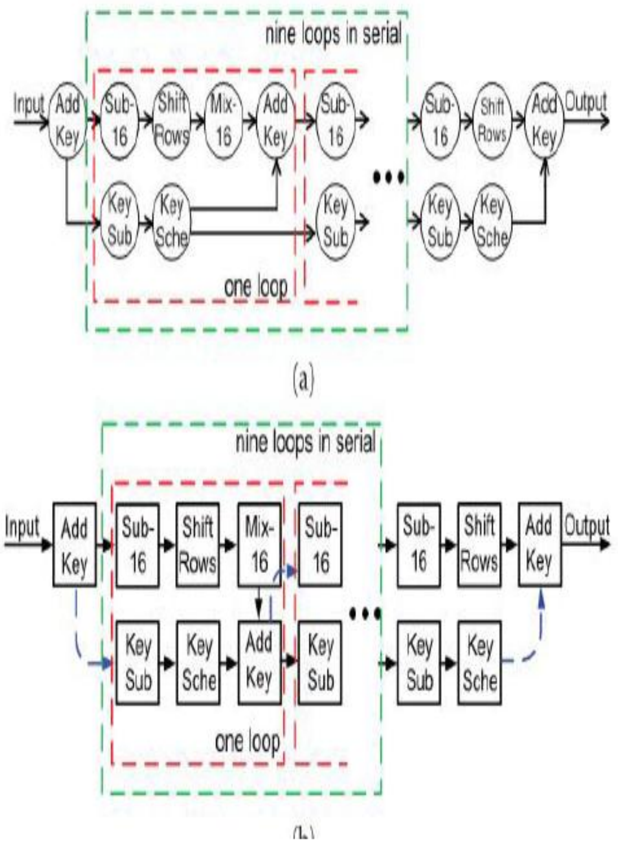

Fig.2. data flow diagram and mapping of the Parallel-Mix Columns model.

A. parallel-Sub Bytes-Mix Columns

In the Parallel-Mix Columns implementation, Sub Bytes-16 requires 132 cycles to encrypt one data block, which contributes the largest execution delay in one loop. In order to increase the throughput further, we parallelize one Sub Bytes- 16 in to four SubBytes-4s. In this implementation, each SubBytes-4 processes 4 bytes rather than 16 bytes in one data block. The effective execution delay of the Sub Bytes process is decreased to 40 cycles per block, only around one fourth as before. Therefore, the throughput of the Parallel Sub Bytes- Mix Columns model is increased to 1,350 cycles per block, equaling 84.375 cycles per byte. The mapping graph of the Parallel-Sub Bytes- Mix-Columns implementation on as requires 22cores.Instead of parallelizing SubBytes-16 in to four SubByte-4s, we can replace it with 16 SubBytes-1s. The effective execution delay of the Sub Bytes process is reduced to 10cycles. As a result, the latency of one-loop decreases to 120cycles.Therefore, the throughput of the cipher is increased to 67.5cycles per byte. However, it requires seven additional cores dedicated to communication (four Merge Cores and three Dispatch Cores), which impair the area and energy efficiency of the implementation.

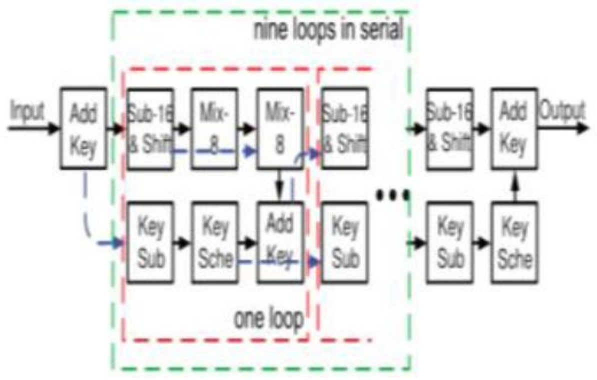

Fig.3. data flow diagram after loop-un rolling.

B. No-Merge-Parallelism

In contrast to the Small model, the No-merge parallelism model exploits as much parallelism as possible without introducing any cores dedicated to communication, including Merge Cores and Dispatch Cores the mapping graph of the No merge- parallelism implementation on As AP. To speed up the implementation, loop unrolling is applied in this model. Each MixColumns-16 is divided in to twoMixColumns-8s, which helps reduce the effective delay of the Mix Columns process. In order to eliminate additional communication processors and simplify the routing, we combine the Sub Bytes and the Shift Rows stages in one core. This implementation requires 59cores, and has a throughput of 152 cycles per block, equaling 9.5 cycles per byte. AES cipher can be partitioned into a number of serial and parallel independent tasks corresponding to different step s in the algorithm. However, the through put of this partitioning is low due to the time-consuming loop operation in the algorithm. In order to enhance the through put, loop-un rolling is applied to break the dependency among loops and allow the cipher to operate on multiple data blocks simultaneously.

To improve the through put as much as possible, we un roll the loops in both the AES algorithm and the key expansion process by Nr 1 times, which equals nine in our design. The data flow diagram after loop-un rolling is shown in Fig.3. Shows a preliminary AES cipher implementation based on the dataflow diagram. Each task in the data flow diagram is mapped to one small processor. As shown inFig.4, seven small processors are required for one loop, four for the AES algorithm and three for the key expansion process, respectively. Therefore, the total number of processors used in this enciphers is:

$$N_{processors} = (N_r - 1) \times N_{one-loop} + N_{last-round} \\ = 9 \times 7 + 7 = 70 \qquad (1)$$

The instruction and data memory sages for each processor in the original design, respectively.Eachrocessorinthe70 core AES cipher uses an average of 28 words fin struction memory, which is 22% of all available instruction memory; and an average of 55 words of data memory, which is 43% of all available data memory

IV. AES MODES OF OPERATION

The AES encryption algorithm accepts one data block and the key and produces the encrypted data block. The input and output data blocks are of identical size. The decryption algorithm accepts one encrypted data block and the key to produce the encrypted data block. Several modes of operation have been defined to apply the AES block cipher to encryption of more than one 128 bit block of data. The most commonly used modes with AES are: electronic code book (ECB) mode, cipher block chaining (CBC) mode, output feedback (OFB) mode, cipher feedback (CFB) mode, and counter (CTR) mode. ECB and CTR are known as no feedback modes whereas CBC, CFB, and OFB are known as feedback modes. In addition, ECB and CBC are referred to as block cipher modes as they require the entire data block before the start of the encryption, and OFB, CFB, and CTR are referred to as stream cipher modes as they operate in a stream-like fashion.

A. Output Feedback Mode

In the OFB mode the output of the encryption is fed back into the input to generate a key generate the cipher data, as illustrated in fig 4 SEU propagation during encryption in OFB mode. In Fig. 4, if an SEU occurs during encryption in the OFB mode then all the subsequent blocks will be corrupted starting from the point where the fault has occurred. This is because the key stream required for encryption and decryption is independent of the plain and cipher data and hence the feedback propagates the faults from one block to another until the end of the encryption process. This is demonstrated by introducing an SEU during the encryption of a plain multispectral satellite image. The satellite image, the image has 500£500 pixels and each pixel is of 24 bits, representing 3 spectral bands with 8 bits per band. Thus, the number of 128 bit blocks for this image is 46875. The fault propagation for a single bit error, which was introduced during the encryption of the 20,000th block at the Sub Bytes transformation of the 4th byte in the third round The propagation of a single bit error that was introduced during the encryption of the 40,000th block at the Mix Columns transformation of the 7thbyte in the 6th round In contrast, if a bit is corrupted during transmission, only a single bit in the plain data is affected and the error does not propagate to other parts of the message again for the same reason that the key stream does not depend on the plain or cipher data. So the transmission fault is not propagated. This property is very useful to applications such as satellites where the transmission channels are very noisy. Hence the OFB mode has an advantage over the CBC and CFB modes in that any bit errors that might occur inside cipher data are not propagated to affect the decryption of subsequent blocks.

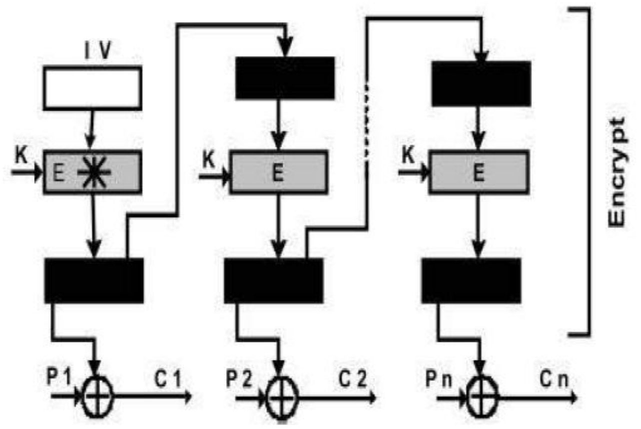

Fig.4.Block diagram of OFB mode.

V. SIMULATION RESULTS

AES Rijndael algorithm is simulated and synthesized using Xilinx 13.1 ISE tool and the targeted FPGA is 5vlx110tff1136-3 which belongs to Virtex-5 family. The design uses only LUTs, ROMs for all the operations of AES encryption and decryption. This approach reduces device utilization and significantly improves the speed compared to other implementation [4, 5, 10]. The key register in the decryption module is synthesized as Block-Ram to reduce the number of slices used. The utilization summary for device 5vlx110tff1136-3 is presented in Table I.

Table I. SLICE Logic Utilization

Number of Slice Registers	128 out of 69120 0%
Number of Slice LUTs	1106 out of 69120 1%
Number used as Logic	1106 out of 69120 1%

In this proposed design, the encryption unit takes 10 clock cycles to complete the operation. The maximum path delay of the design is 3.420ns resulting in a maximum frequency of operation as 292.403MHz. The throughput of the proposed encryption module is 3.74Gbps which is given by the Equation 1.

$$\text{Throughput} = \frac{\text{Number of block color Frequency}}{\text{Number of Frequency}} \quad (1)$$

The proposed work is also compared with the work done by Kretzschmar et al. [5] on different AES architectures implementation on xilinx virtex-5 FPGA Table II presents the comparison result of proposed work with Kretzschmar et al. [5] and also with other references.

Table II. Comparison [5]

Design	Delay (ns)	Max.frequency (MHz)	Throughput (Gbps)	LUTs	Registers
Proposed design	3.42	292.40	3.74	1%	1%
M. Goswami &S. kannujiya[1]	4.25	235.29	2.73	.7%	1%
W. Wei, C.jie, X.Fei[3]	4.97	201.20	2.57	Not Available	
Kampen *	5.291	189	.016	73%	39%
J.Castillo *	6.711	149	.0375	94%	57%
Four sbox AES *	6.493	154	.241	93%	22%
H.Satyanarayan*	3.690	271	2.166	81%	33%

* Reference given in the paper by Kretzschmar et al. [5]

VI. CONCLUSION

AES-128 algorithm for encryption and decryption is implemented in Virtex-5 FPGA. With the designing of all the operations as LUTs and ROMs, the proposed architecture achieves a throughput of 3.74 Gbps and thereby utilizing only 1% of slices in the targeted FPGA. Since the speed is higher than the already reported systems, hence the proposed design serves as the best high speed encryption algorithm and is thus suitable for various applications. Moreover with less area utilization, the proposed design can be embedded with other larger designs as well.

VII. REFERENCES

[1] Abhijith.P.S, Mallika Srivastava, Aparna Mishra, Manish Goswami, B.R.Singh, “High Performance Hardware Implementation of AES Using Minimal Resources”, 2013 International Conference on Intelligent Systems and Signal Processing (ISSP).
[2] M. Goswami and S. Kannojiya, “High Performance FPGA Implementation of AES Algorithm with 128-Bit Keys,” Proc. IEEE Int. Conf. Advances Computing Comm., vol. 1, Himarpur, India, 2011, pp. 281-286.
[3] FIPS-197, NIST - National Institute of Standards and Technology, “Announcing the Advanced Encryption Standard (AES),” http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.
[4] W. Wei, C. Jie and X. Fei, “An Implementation of AES Algorithm on FPGA,” IEEE 9th Int. Conf. on Fuzzy Systems and Knowledge discover 2012, pp. 1615-1617.
[5] U. Kretzschmar, A. Astarloa, J. Lazaro, U. Bidarte and J. Jimenez, “Robustness analysis of different AES

implementations on SRAM based FPGAs," Int. Conf. on Reconfigurable Computing and FPGAs 2011, pp. 255-260.
[6] J. Daeme and V. Rijmen, "AES proposal: Rijndael," NIST AES Proposal, June 1998.
[7]W. Stallings, "Cryptography and network security principles and practice," Pearson edition 2009, pp. 135-160.
[8]P.V.S. Shastry, A.Agnihotri, D. Kachhwaha, J. Singh and M.S. Sutaone, "A Combinational Logic Implementation of S-Box of AES," IEEE 54th Int. Midwest Symp.on Circuits and Systems (MWSCAS), Aug. 2011, pp. 1-4.
[9] S. Kaur and R. Vig, "Efficient Implementation of AES Algorithm in FPGA Device," Int. Conf. on Computational Intelligence and Multimedia Applications, Dec. 2007, pp. 179 – 187.
[10]H. Trang and N.V. Loi, "An efficient FPGA implementation of the Advanced Encryption Standard algorithm," IEEE Int. Conf. on Computing and Communication Technologies, Research, Innovation and Vision for the Future (RIVF), 2012, pp. 1-4.