ISSN(Print):2320–3765
ISSN (Online): 2278 – 8875

Evaluation Of Microblaze and Implementation Of AES Algorithm using Spartan-3E

M.Sambasiva Reddy[1], Mr. Y. Amar Babu[2]

PG student Dept. of ECE, LBRCE college of Engineering,Mylavaram,India[1]

Associate Professor, Dept. Of ECE, LBRCE college of Engineering, Mylavaram,India[2]

ABSTRACT: Security is the most important part in data communication system, where more randomization in secret keys increases the security as well as complexity of the cryptography algorithms. As a result in recent sates these algorithms are compensating with enormous memory spaces and large execution time on hardware platform scenario due to its reconfiguration nature low price and marketing speed. In this paper we demonstrate the design and implementation of a 128-bit Advanced Encryption Standard(AES) both symmetric encryption and decryption algorithm by developing suitable hardware and software design on Xilinx Spartan-3E(XC3S500E-FG320)device. The system is optimized in terms of execution speed and hardware utilization. We have several real time applications based on AES algorithm like Data integrity, Data secrecy and authentication and various other domains, Digital signature. The future scope of this project is to run an application using this AES Implementation.

Keywords:AES, Encryption, Decryption, Cryptography, Implementation, Spartan-3E

I.INTRODUCTION

Nowadays cryptography has a main role in Embedded systems design. In many applications, the data requires a secured connection which is usually achieved by cryptography. Cryptography is divided in two categories first is symmetric key cryptography (sender and receiver shares the same key) and the second one is asymmetric key cryptography (sender and receiver shares different key).

Micro blaze is a 32-bit RISC softcore processor with HARVARD architecture, it has 32, 32 bit general purpose resistors. It is a 32 bit Instruction word with three operands and two addressing modes .Microblaze has a separate instruction and data busses of 32 bit which conform to IBM’s OPB(ON CHIP PERIPHERAL BUS).Microblaze has 2 types of pipelining. Some applications are implemented in software ,some require hardware implementation, but some require both the combination of software and hardware i.e.. parallel implementation of both software and hardware which increases system performance. EDK(Embedded development kit) which provides both the parallel processing of software and hardware implementation.

The National Institute of Standards and Technology(NIST), solicited proposals for the Advanced Encryption Standard(AES). The AES is a Federal Information Processing Standard(FIPS), which is a cryptographic algorithm that is used to protect electronic data. 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.

The proposed architecture was synthesized using Xilinx ISE 13.3 and was implemented on XC3S500e Spartan 3E FPGA Board . The necessary software for this design is written using the feature-rich C/C++ code editor and compilation environment provided within the SDK (Xilinx Software Development Kit). The SDK provides an environment for creating software platforms and applications targeted for Xilinx embedded processor (MicroBlaze). SDK works with hardware designs created with Xilinx Platform Studio (XPS).

II. AES ALGORITHM

AES comes in three favors, namely AES - 128, AES - 192, and AES-256, with the number in each case representing the size (in bits) of the key used. All the modes are done in 10, 12 or 14 round depends on the size of the block and the key length chosen. AES merely allows a 128 bit data length that can be divided into four basic operation blocks. These blocks operate on array of bytes and organized as a 4*4 matrix that is called the state. The algorithm begins with an Add round key stage followed by nine rounds of four stages and a tenth round of three stages which applies for both encryption and decryption algorithm [1] [2] [4].
These rounds are governed by the following four stages:

- Substitute Bytes

ISSN(Print):2320–3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 7, July 2013

- Shift rows
- Mix columns
- Add round key

The tenth round Mix columns stage is not included. The first nine rounds of the decryption algorithm are governed by the following four stages:

- Inverse Shift rows
- Inverse Substitute Bytes
- Add round key
- Inverse Mix columns

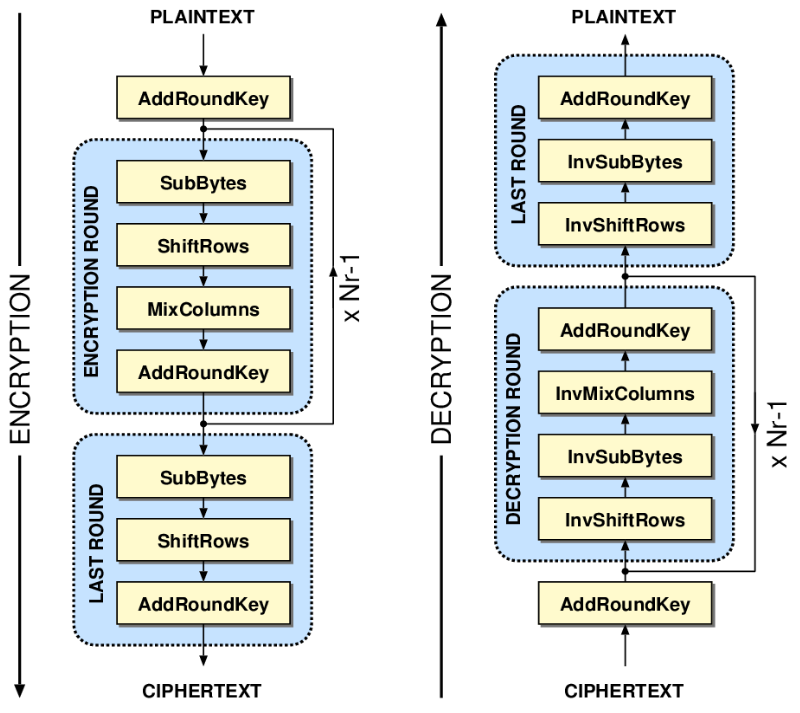

Figure 1: Design flow of AES Algorithm (a) Encryption Process (b) Decryption Process

Again the tenth round Inverse Mix columns stage is not included. The Overall flow of the encryption and ecryption algorithm of the AES algorithm is show in Figure1.

Substitute Bytes: Is a non linear byte substitution, using a substation table (S-box) each byte from the input state is replaced by another byte. The substitution is invertible and is constructed by the composition of two transformations as described below [2] [8]. The substitute bytes operation is as shown in Figure 3.

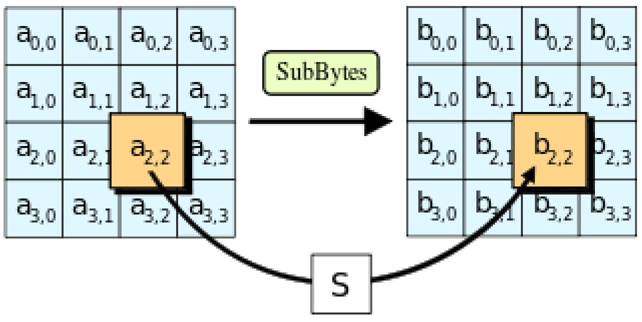
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S

Figure 2: Affine transformation of Substitute Bytes

ISSN(Print):2320–3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 7, July 2013

1. The state bytes are constructed by multiplicative inverse in the finite field GF (28) that is used in the AES. This field is derived using the following irreducible polynomial of degree 8:

m(x) = x8+x4+x3+x+1. (1)

2. Affine transformation is applied on the result of above statement. The fixed matrix and the fixed vector over GF (2) in the affine transformation are shown in the Figure 2.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 2: Affine transformation of Substitute Bytes

Inverse Substitute Bytes: Is the reverse operation of the Substitute Bytes transformation, in which the inverse S-box is applied to each byte of the State. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF (28).

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 as shown in Figure 4. Inverse Shift rows: Is the inverse of the shift rows, 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 right, respectively.

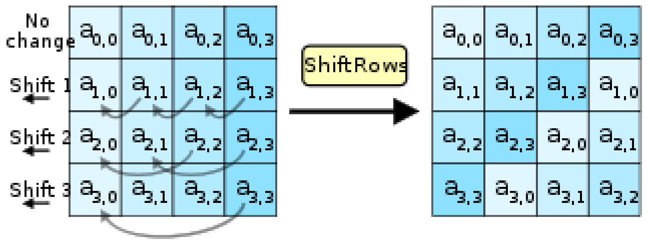
No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Figure 4: Shift Rows Operation

Mix columns:In the Mix Columns transformation ,every column of the state array is considered as polynomial over GF (28). After multiplying modulo x4+1 with a fixed polynomial a(x) [2], the operation of MixColumn is as shown in Figure 5.

a(x)=03*x3+01*x2+ 01*x + 02 (2)

The result is the corresponding column of the output state is as shown in Figure 5. As it can be noticed this operation requires multiplication by „two" and „three" that is relatively simple shift operation in hardware.

ISSN(Print):2320–3765
ISSN (Online): 2278 – 8875

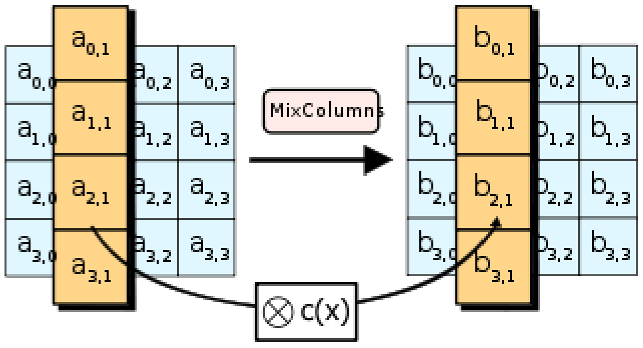

Figure 5: MixColumn Operation

Inverse Mix columns: In the Inverse Mix Columns transformation, every column of the state array is considered a polynomial over GF (28). After multiplying modulo x4+1 with a fixed polynomial b(x),

$$b(x)=0B*x3 + 0D*x2 + 09*x + 0E \qquad (3)$$

The result is the corresponding column of the output state. As it not so straightforward hardware implementation as Mix column, so if we compare both, InvMixCol requires more logic resources for implementation.

Add Round Key: The Add Round Key operation is as shown in Figure 6, which is a simple XOR operation between the State and the Round Key. The Round Key is derived from the Cipher key by means of key schedule process. The State and Round Key are of the same size and to obtain the next State an XOR operation is done per element:

$$b\ (i, j) = a\ (i, j) \oplus k\ (i, j) \qquad (4)$$

Where a is the current State, b the next State and k the round key.

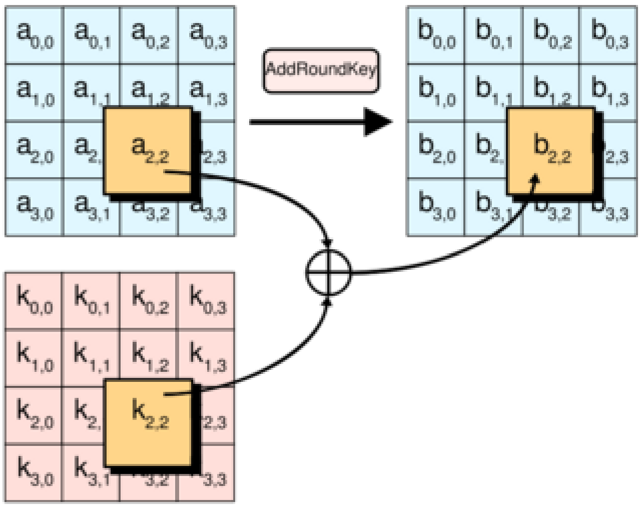

Figure 6: AddRound Key Operation

There are three steps, in each Key schedule round [6]. Keyrotate: The function Keyrotate takes a four-byte word and rotates one byte to the left. Keysubytes: The Keysubytes operation takes four-byte input word by substituting each byte in the input to another byte according to the S-Box. KeyRcon: The first byte of a word is XORed with the round constant. Each value of the Rcon table is a member of the Rinjdael finite field. Add round key is same for the both encryption and decryption.

ISSN(Print):2320–3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 7, July 2013

III. IMPLEMENTATION

A. AES Encryption :

To implement the AES algorithm using VHDL coding in the Xilinx 12.4 [3], we proceed with the encryption and decryption with 128 bits. The AES Encryption block is as shown in Figure 7, the encryption parameters are the input plaintext, the key of size key size and the output cipher text. First, we have to map the 16 byte input plaintext in the correct order to the 4*4 byte state, calculate the number of rounds based on the key Size and expand the key using our key schedule. At round one plaintext and key is XORed, the remaining nine rounds which has to apply all four operations: Substitute Bytes, Shift rows, Mix columns, Add round key.

The tenth round Mix columns stage is not included and round key was generated during each iteration [7]. Here simple XOR of each byte of the key with the respective byte of the state is done to get cipher text of the Encryption algorithm.

B. AES Decryption:

For AES Decryption, the same encryption process occurs simply in reverse order. The decryption block is as shown in Figure 8, the encryption parameters are the input cipher text, the key and the output plaintext should be same as encryption input. In decryption the key schedule remains the same; the only operations we need to implement are the Inverse subBytes, shiftRows and mixColumns, while addRoundKey stays the same.

EDK IMPLEMENTATION OF AES ALGORITHM:

In this chapter we had already seen the implementation of AES algorithm using SOFTWARE and HARDWARE individually. In those implementations some disadvantages are there. Disadvantages of software implementation overcome by hardware implementation and disadvantages of hardware implementation can be overcome by using EDK implementation. In EDK implementation we use both software and hardware combinely.

Embedded design in an FPGA:

For this design process we can use MICROBLAZE which is a softcore processor and we use SPARTAN 3E starter kit as FPGA kit for the synthesis purpose.

The Embedded Development Kit (EDK):

The Embedded Development Kit (EDK) is a suite of tools *and* IP that you can use to design a complete embedded processor system for implementation in a Xilinx FPGA device.

Xilinx Platform Studio (XPS):

Xilinx Platform Studio (XPS) is the development environment used for designing the *hardware* portion of your embedded processor system. You can run XPS in batch mode or using the GUI, XPS is now integrated with PlanAhead tools, making development easier.

Software Development Kit (SDK):

The Software Development Kit (SDK) is an integrated development environment, complementary to XPS, that is used for C/C++ embedded software application creation and verification. SDK is built on the Eclipse open-source framework and might appear familiar to you or members of your design team. For more information about the Eclipse development environment Typically, the ISE Design Tools development software is used to add an Embedded Processor source, which is then created in XPS using the Base System Builder.

Selecting a Board Type:

The BSB requires the selection of an available development board, or a custom board. Supported development boards can be selected in Project Navigator, or if starting in XPS, in the BSB introduction screens.

XPS Tools:

XPS includes the underlying tools you need to develop the hardware components of an embedded processor system

Platgen:

XPS includes the Hardware Platform Generation tool, Platgen, which generates HDL and lower level netlists for the embedded processor system. When you implement the FPGA design in the PlanAhead™ design tool, Platgen automatically runs if necessary.

Converting the Hardware Platform to a Bitstream:

For a design to work in an FPGA, it needs to be converted to a bitstream. This conversion is the following three-step process:

1. XPS generates a netlist that is representative of your embedded hardware platform.
2. The design is implemented (mapped into FPGA logic) in the PlanAhea tool.
3. The implemented design is converted to the bitstream that can then be downloaded to the FPGA.

Configuring the FPGA: Download the bitstream

– Input file → download.bit

ISSN(Print):2320–3765
ISSN (Online): 2278 – 8875

– This downloads the download.bit file onto the target board using the Xilinx iMPACT tool in batch mode
– XPS uses the etc/download.cmd file for downloading the bitstream
• The download.cmd file contains information such as the type of cable is used and the position of the FPGA in a JTAG chain.

IV.RESULTS

Encryption Process (128 bit) :
Plain Text: 3658001543081a4f0a0c42130b3a0c00.
Key: 000102030405060708090a0b0c0d0e0f.
Cipher Text: 61d03aeddde79bdbda779bcaf0abfffe.
Decryption Process (128 bit) :
Cipher Text: 61d03aeddde79bdbda779bcaf0abfffe.
Key: 13111d7fe3944a17f307a78b4d2b30c5.
Plain Text: 3658001543081a4f0a0c42130b3a0c00.

V.CONCLUSION

We studied about Micro Blaze and evaluated it using small programs. We implemented AES algorithm in all three modules, which are software platform ,hardware platform and mixing of software and hardware platforms i.e using microblaze processor. Implementing AES algorithm using EDK(embedded development kit) is the better solution than the Hardware implementation and Software implementation. The execution speed is more, Power consumption is less and size required for implementation on FPGA is less when compared to Hardware implementation.

COMPARISION:

	Time for Execution (sec)	Size (MB)	Power Consumption (Watts)
Software	32	10	17
Hardware	20	22	1.344
EDK	2.06	1	0.513

FUTURE SCOPE:

In this project we had done the implementation of AES Algorithm.We can extend this implementation to image encryption and decryption using AES algorithm.

REFERENCES

[1]. MicroBlaze overview.
http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/mb_ref_guide.pdf
[2]. MicroBlaze Processor Reference User Guide, Xilinx, Inc., Aug. 2004. Embedded Development Kit (EDK 6.3), UG081, Version 4.0
[3]. EDK Reference manual
http://www.xilinx.com/ise/embedded/edkdocs.htm
[4]. Spartan-3 FPGA Family Data Sheet
http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf
[5]. National Institute of Standards and Technology, "Federal Information Processing Standard Publication 197, the Advanced Encryption Standard (AES)," Nov. 2001.
[6]. William Stalling, Cryptography and Network Security: Principles and Practices, Principles and Practices, 4th ed. Prentice Hall, 2006.
[7]. Animation of AES algorithm
http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng.swf
[8]. Mehran Mozaffari-Kermani and Arash Reyhani-Masoleh "Efficient and High Performance Parallel Hardware Architecture for the AES-GCM" IEEE Transactions On Computers, vol.61,no. 8, August 2012.
[9]. Saambhavi Baskaran and Pachamuthu Rajalakshmi "Hardware Software Co-Design of AES on FPGA" ICACCI '12,ACM August 2012.
[10]. Ashwini M. Deshpande, Mangesh S. Deshpande and Devendra N. Kayatanavar "FPGA Implementation of AES Encryption and Decryption" International Conference on Control, Automation, Communication and Energy conservation -2009

ISSN(Print):2320–3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering
Vol. 2, Issue 7, July 2013

BIOGRAPHY

M.Sambasiva Reddy is a PG student in Lakireddy Bali reddy College of Engineering, Mylavaram. This paper is the work of his project. Email: sambasivareddymula@yahoo.com, Area of Interest is VLSI system design. He completed hi B.tech degree in Bapatla Engineering College, Bapatla. Interested in Research feild.

Mr. Y. Amar Babu had Completed Pre-Ph.D from JNTU, Hyderabad in Jan 2010. He completed his ME in VLSI System Design PSG College of Technology, Coimbatore in Jan 2002. Published several research papers in various journals. Presently working as a Associate Professor in Lakireddy Bali reddy College of Engineering,Mylavaram,A.P.