Performance Evaluation of AES using Hardware and Software Codesign

Vilas V Deotare[1], Dinesh V Padole[2] Ashok S. Wakode[3]
Research Scholar,Professor, GHRCE, Nagpur, India
vilasdeotare@gmail.com[1], dvpadole@gmail.com[2],samratasoka@yahoo.com[3]

Abstract – Implementation of Advanced Encryption Standard (AES) algorithm is more intensive discussion from its starting publication especially in terms of performance. However the studies of implementation of AES using hardware or software, the performance and cost are high to low respectively. The propose method is implemented using hardware and software Co design. This paper presents AES implementation on different platform like ARM7, micro- blaze, FPGA and compares the results with these performances. The target hardware used in this paper is Spartan 3s400PQ208-4 FPGA from Xilinx. The performance is simulated and validated.

General Terms

Advanced Encryption Standard, Advance RISC machine, Xilinx Platform Studio, Soft core processor.

Keywords

Time evaluation, Substitution, Encryption, Decryption Plain text, cipher text, key operating frequency, FPGA, VHDL

*****

I. INTRODUCTION

The importance of cryptography applied to security in electronic data transactions has acquired an essential relevance during the last few years. Each day millions of users generate and interchange large volumes of information in various fields, such as financial and legal files, medical reports, and bank services via Internet, telephone conversations, and e-commerce transactions. These and other examples of applications deserve a special treatment from the security point of view, not only in the transport of such information but also in its storage. This implementation will be useful in wireless security like military communication and mobile telephony where there is a greater emphasis on the speed of communication. Low data rate communication like Zigbee and industrial protocols like WirelessHART and ISA100 protocols are also using this AES security.

In cryptography, the AES, also known as Rijndael, is a block cipher adopted as an encryption standard by the US government, which specifies an encryption algorithm capable of protecting sensitive information [1,2]. 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 hardware implementation of the Rijndael algorithm can provide either high performance or low cost for specific applications. At backbone communication channels or heavily loaded servers it is not possible to lose processing speed, which drops the efficiency of the overall system while running cryptography algorithms in software.

On the other side, a low cost and small design can be used in smart card applications, which allows a wide range of equipment to operate securely. This paper is organized as follows: description of Rijndael cipher, decipher, design preliminaries, implementation on different platform and comparing result. These results are summarized in table form and finally concluded the result. Future scope is mentioned for further development.

II. AES ALGORITHM

AES comes in three flavors, 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] .

These rounds are governed by the following four stages:

- Substitute Bytes
- 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

Again the tenth round Inverse Mix columns stage is not included

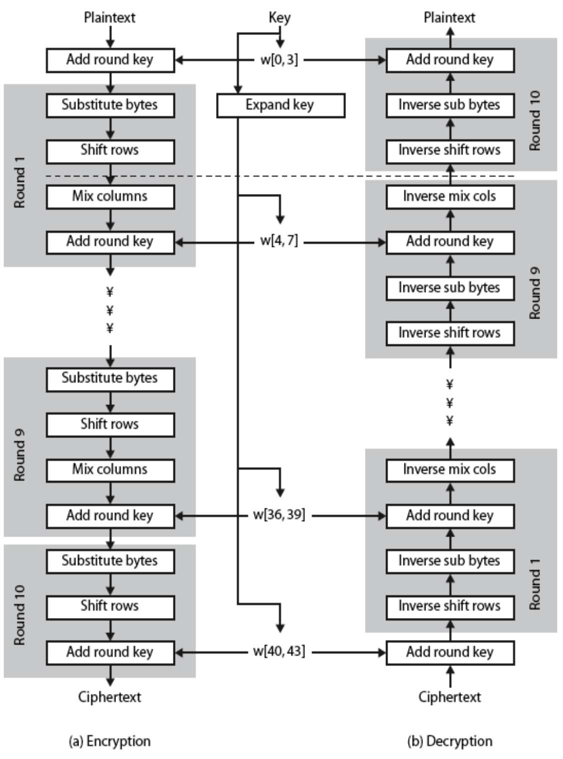


Fig.1-AES Flow

Figure 1 explains the how AES encryption and decryption goes through different stages as explained in above paragraph.

A. Design Preliminaries

The input and output for the AES algorithm each consist of sequences of 128 bits.[1] These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The input, output and cipher key bit sequences are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes. The different transformations operate on the intermediate result, called the state, which is the intermediate cipher result. The state can be pictured as a rectangular array of bytes. This array has four rows; the number of columns is denoted by Nb and is equal to the block length divided by 32. The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key is denoted by Nk and is equal to the key length divided by 32. The number of rounds is denoted by Nr and depends on the values Nb and Nk. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8. For both its cipher and inverse cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: byte substitution using a substitution table (S-box), shifting rows of the state array by different offsets, mixing the data within each column of the state array, and adding a round key to the state.

B)Encryption Algorithm:

```
Encryption(plaintext[128 bit], ciphertext[128bit], keyexp[][128bit])
begin
        128bit state
        state = plaintext
        AddRoundKey(state, keyexp[0][128bit])
for round = 1 step 1 to 9
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state,keyexp[round][128bit]])
end for
        SubBytes(state)
        ShiftRows(state)
        AddRoundKey (state,keyexp[10][128bit])
        ciphertext = state
end
```

C) Decryption Algorithm:

```
Decryption(ciphertext[128bit],plaintext[128bit], keyexp[][128bit])
  begin
        128bit state
         state = ciphertext
        AddRoundKey (state, keyexp[10][128bit])
   for round = 9 step -1 downto 1
        InvShiftRows(state)
        InvSubBytes(state)
        AddRoundKey(state,
        keyexp[round][128bit])
        InvMixColumns(state)
 end for
        InvShiftRows(state)
         InvSubBytes(state)
        AddRoundKey(state, keyexp[0][128bit])
        plaintext = state
 end
```

D) Total Overview of Work

1) AES on ARM7:

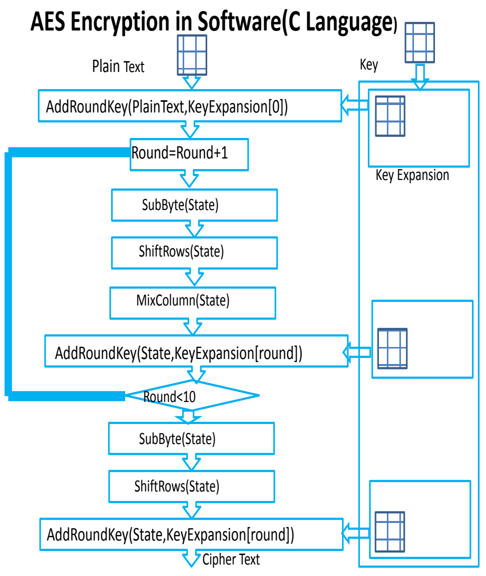

AES Encryption in Software(C Language)
Plain Text
Key
AddRoundKey(PlainText,KeyExpansion[0])
Key Expansion
Round=Round+1
SubByte(State)
ShiftRows(State)
MixColumn(State)
AddRoundKey(State,KeyExpansion[round])
Round<10
SubByte(State)
ShiftRows(State)
AddRoundKey(State,KeyExpansion[round])
Cipher Text

Fig 2. AES Encryption in Software(C Language)

In this figure,plain text matrix is given to add round key stage where plain text are xored with expanded key which comes from key expansion function.This output is called state. This state is looped 9 times and at 10 th round it skips mix column and finally cipher text are generated.

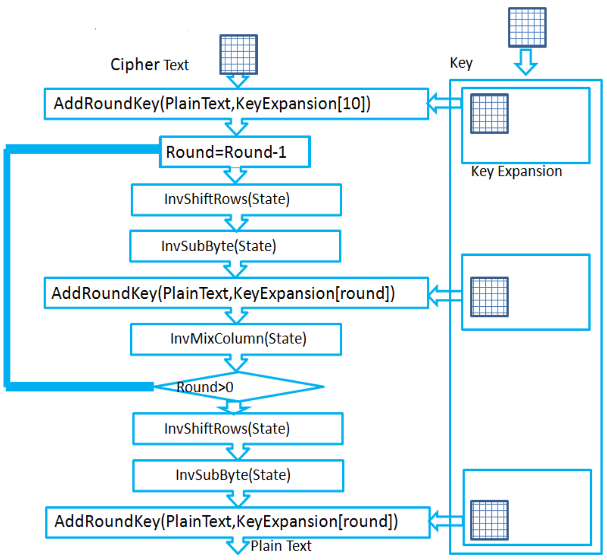


Fig 3. AES Decryption in Software(C Language)

In Figure 3 reverse process of encryption is explained to get original plain text keeping same cipher key for process.

To create a demo of AES two human interface devices, keyboard and hyper terminal is used. HyperTerminal can be used as embedded console to show the details of demo of AES. When system is powered ON, it will gives the information and instructions to user to put the data for Plain text as well as cipher text. All the data input to the AES demo is in hex format with two ASCII separated by comma. After feeding 16 byte ENTER is pressed .HyperTerminal is configured for baud rate 115200, 8, N, 1.On ARM7 UART 2 is configured for 115200, 8, N, 1. Internal Timer of ARM7 is used for performance measurement. This timer works on 250 KHz. So numbers of counts are noted down for Encryption and Decryption. So time taken by Encryption calculated as: Each count of this timer takes 4 microseconds. So Time taken for Encryption or Decryption is Timer count * 4 .This time is in microsecond.
On HyperTerminal Encrypted data is shown for data verification. Same data is given for Decryption so that original data can be retrieved from decryption engine. We got original data

E) AES On MicroBlaze:

On FPGA in microblaze XPS UARTLITE is synthesized for hyper terminal interface .XPS UART Lite is configured as 9600 baud rate with 8 bit data without parity. Same configuration is done on PC Hyperterminal.XPS Timer without interrupt is synthesized for Time measurement.25MHz clock is given to timer. When AES block is started at that time this timer is started. When Encryption is completed, Timer counts are captured. This timer takes 1 microsecond for each 25 count. So captured count is divided by 25 so that we get the time required for encryption or decryption in microsecond.

F) AES On custom IP in FPGA:

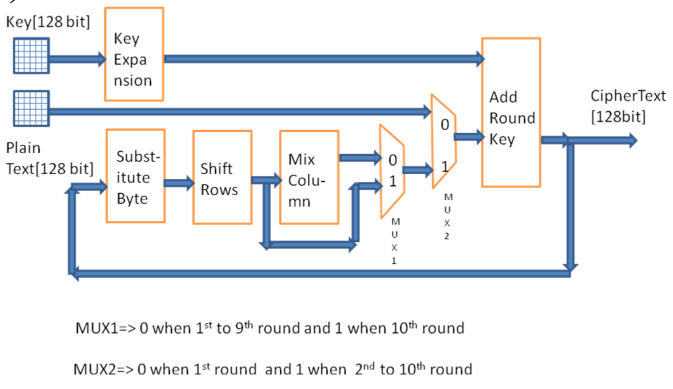


Fig.4 AES Encryption in Hardware (VHDL)

In above figure,Plain text and key in 128 bit form are provided .Key expansion block expands key to provide expanded key for each round.AES encryption flow is controlled by 2:1 MUX.

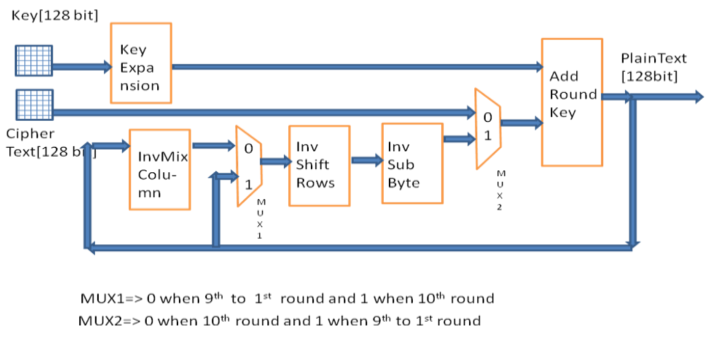


Fig.5 AES Decryption in Hardware (VHDL)

Figure 5 explains decryption flow controlled by two 2:1 Mux.These muxes decide the path of execution.

Custom IP implementation and FPGA simulation, VHDL language is used .VHDL is used as the hardware description language because of the flexibility to exchange among environments. The code is pure VHDL that could easily be implemented on other devices, without changing the design The software used for this work is Xilinx - Project Navigator, ISE 10.1 suite. This is used for writing, debugging and optimizing efforts, and also for fitting, simulating and checking the performance results using the simulation tools available on ISim Xilinx software.
For platform building and custom IP synthesis, XPS 10.1 is used for AES evaluation on microblaze with custom IP.

F) Hardware and Software Implementation

About Hardware Software Co-design hardware and software has advantages and disadvantages. Hardware works parallel so speed is obviously major advantage.
In case of software, every action is taken by microcontroller, so it runs in that sense sequentially.
Software development is fast and it seems very simple when we are handling complex project but hardware development

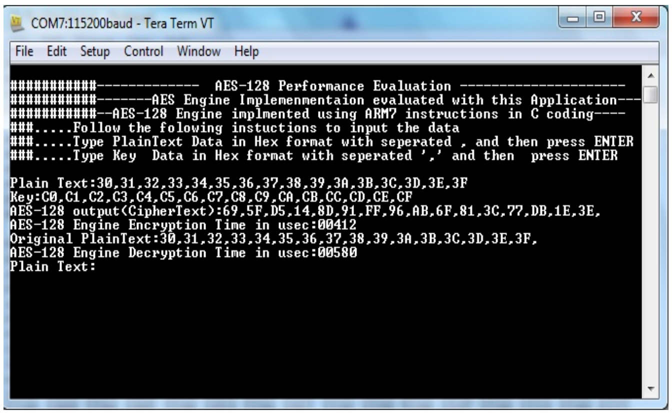
becomes hard in case of volumetric project. So in such scenario, both advantages can be combined to achieve optimize performance. Considering theses aspect, we have chosen hardware software co design strategy for AES performance evaluation. Some of part are implemented in software on microblzae platform and some of the part is implemented in hardware as custom IP. For Software implementation, C language is preferred and VHDL is preferred for hardware development. This co design is implemented using Xilinx platform studio.

III. HW/SW CODESIGN IMPLEMENTATION ON SPARTAN:

AES is synthesized and added to microblzae as peripheral. This is interfaced to microblzae as user register logic. These register are accessed as memory of 32 bit .Total 14, 32-bit register are accessed in this demonstration. 4 32-bit register are allocated for plain text data. Plain text is in 16 byte. Each data byte are taken from user in ASCII format .It is converted in Hex format .4 byte are given to a single register. Union structure is used for byte to 32-bit register conversion. Plain text data takes 4 register and Key data takes again 4 register. Resultant of AES operation is stored into 4 register for further display or processing. Two register are allocated for control of the AES engine and status of the AES engine. AES controls are given to AES engine and status is checked for operation completion. This flag is used for time calculation.

A) Implementation on ARM7 result:

C language is used for AES implementation on ARM7 and Microblzae. Reason is C portability. This can be used easily to any platform with less change in platform.
AES encryption and decryption are implemented as per standard. User data can be taken through standard IO.
C language is faster for implementation as well as give assembly like performance if it used pre-cautiously. It boosts our development period.


```
COM7:115200baud - Tera Term VT
File  Edit  Setup  Control  Window  Help

###########-------------  AES-128 Performance Evaluation ---------------------
###########-------AES Engine Implemenmentaion evaluated with this Application---
###########--AES-128 Engine implmented using ARM7 instructions in C coding----
###.....Follow the folowing instuctions to input the data
###.....Type PlainText Data in Hex format with seperated , and then press ENTER
###.....Type Key  Data in Hex format with seperated ',' and then  press ENTER

Plain Text:30,31,32,33,34,35,36,37,38,39,3A,3B,3C,3D,3E,3F
Key:C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,CA,CB,CC,CD,CE,CF
AES-128 output(CipherText):69,5F,D5,14,8D,91,FF,96,AB,6F,81,3C,77,DB,1E,3E,
AES-128 Engine Encryption Time in usec:00412
Original PlainText:30,31,32,33,34,35,36,37,38,39,3A,3B,3C,3D,3E,3F,
AES-128 Engine Decryption Time in usec:00580
Plain Text:
```

Fig.6: TeraTerm used for input and showing result

Figure 6 shows user input , plain text and key and shows encryption alogortm output in hex form.Alghorithm performance in time are shown and it is in 412 microsec.It shows decryption performance i.e.580 microsecond.

B) Implementation on MicroBlaze and Custom IP

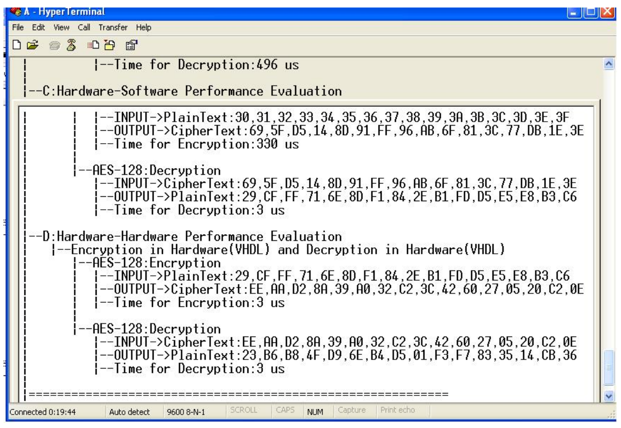

```
A - HyperTerminal
File  Edit  View  Call  Transfer  Help
 |          |--Time for Decryption:496 us
 |
 |--C:Hardware-Software Performance Evaluation
 |      |   |--INPUT->PlainText:30,31,32,33,34,35,36,37,38,39,3A,3B,3C,3D,3E,3F
 |      |   |--OUTPUT->CipherText:69,5F,D5,14,8D,91,FF,96,AB,6F,81,3C,77,DB,1E,3E
 |      |   |--Time for Encryption:330 us
 |      |
 |      |--AES-128:Decryption
 |          |--INPUT->CipherText:69,5F,D5,14,8D,91,FF,96,AB,6F,81,3C,77,DB,1E,3E
 |          |--OUTPUT->PlainText:29,CF,FF,71,6E,8D,F1,84,2E,B1,FD,D5,E5,E8,B3,C6
 |          |--Time for Decryption:3 us
 |
 |--D:Hardware-Hardware Performance Evaluation
 |   |--Encryption in Hardware(VHDL) and Decryption in Hardware(VHDL)
 |      |--AES-128:Encryption
 |      |   |--INPUT->PlainText:29,CF,FF,71,6E,8D,F1,84,2E,B1,FD,D5,E5,E8,B3,C6
 |      |   |--OUTPUT->CipherText:EE,AA,D2,8A,39,A0,32,C2,3C,42,60,27,05,20,C2,0E
 |      |   |--Time for Encryption:3 us
 |      |
 |      |--AES-128:Decryption
 |          |--INPUT->CipherText:EE,AA,D2,8A,39,A0,32,C2,3C,42,60,27,05,20,C2,0E
 |          |--OUTPUT->PlainText:23,B6,B8,4F,D9,6E,B4,D5,01,F3,F7,83,35,14,CB,36
 |          |--Time for Decryption:3 us
 |
 |==============================================================
Connected 0:19:44   Auto detect   9600 8-N-1   SCROLL   CAPS   NUM   Capture   Print echo
```

Fig.7: Showing data input and AES result on Hyperterminal

In this figure 7 Hardware and software performance are shown.AES encryption implemented in software and hardware are shown in figure.Time taken by both algorithm implementation are shown in microsecond.

C) Simulation Result On FPGA:

Simulation Test Vectors For Encryption process:
Plain Text: 00112233445566778899aabbccddeeff.
Key: 000102030405060708090a0b0c0d0e0f.
Cipher Text: 69c4e0d86a7b0430d8cdb78070b4c55a

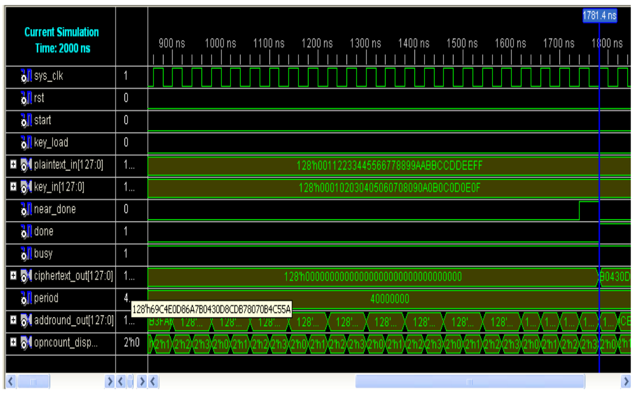


Fig.8: Encryption simulation

In figure 8,all intermediate and final result of encryption are dispalyed.Starting to end process time is shown in vertical blue line.

Simulation Test Vectors For Decryption process:

Cipher Text: 69c4e0d86a7b0430d8cdb78070b4c55a.

Key: 000102030405060708090a0b0c0d0e0f.

Plain Text: 00112233445566778899aabbccddeeff.

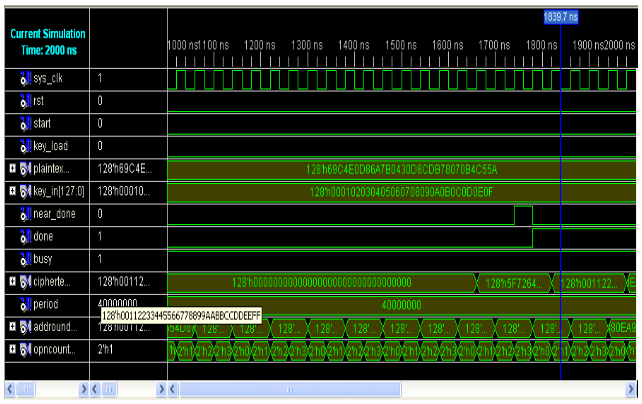

Current Simulation Time: 2000 ns
1839.7 ns
1000 ns 1100 ns 1200 ns 1300 ns 1400 ns 1500 ns 1600 ns 1700 ns 1800 ns 1900 ns 2000 ns
sys_clk 1
rst 0
start 0
key_load 0
plaintex... 128'h69C4E...
128'h69C4E0D86A7B0430D8CDB78070B4C55A
key_in[127:0] 128'h00010...
128'h000102030405060708090A0B0C0D0E0F
near_done 0
done 1
busy 1
cipherte... 128'h00112...
128'h00000000000000000000000000000000
128'h5F7264...
128'h001122...
period 40000000
40000000
128'h00112233445566778899AABBCCDDEEFF
addround... 128'h00112...
opncount... 2'h1

Fig 9: Decryption simulation

Figure 9 shows decryption result along with all stages output of decryption process.At the end of process decryption data i.e.plain text are displayed.

D) Test vectors and Results:

Encryption Process (128 bit):

Plain Text: 303132333435363738393A3B3C3D3E3F.

Key: C0C1C2C3C4C5C6C7C8C9CACBCCCDCF.

Cipher Text: 695FDF148DFF96AB6F813C77DB1E3E.

Decryption Process (128 bit) :

Cipher Text: 695FDF148DFF96AB6F813C77DB1E3E.

Key: C0C1C2C3C4C5C6C7C8C9CACBCCCDCF.

Plain Text: 303132333435363738393A3B3C3D3E3F

Summarized Result:

AES Implementation	AES type	Device Name	Operating Frequency	Performance in Mbps
Software on ARM7	Decryption	MC13226V	24MHz	0.256
Software on Microblaze	Decryption	MicroBlaze on 3s400PQ208-4	25MHz	0.258
Software on ARM7	Encryption	MC13226V	24MHz	0.324
Software on Microblaze	Encryption	MicroBlaze on 3s400PQ208-4	25MHz	0.3878
Microblaze co-design	Encryption	MicroBlaze on 3s400PQ208-4	25MHz	42
Spartan-2[8]	Encryption/ Decryption	2S30	60MHz	70
Simulation for Decryption	Decryption	3s400PQ208-4	25MHz	70.1
Simulation for Encryption	Encryption	3s400PQ208-4	25MHz	71
Cyclone Decryption[9]	Decryption	EP2C35F672C6	51MHz	142
Cyclone Encryption[9]	Encryption	EP2C35F672C6	60MHz	213

Table No.1- Hardware and software performance

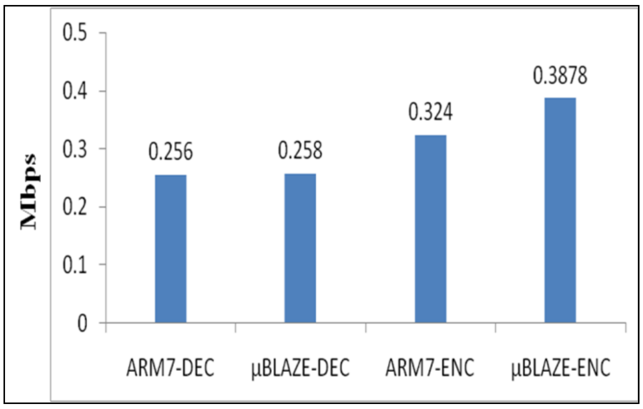


Fig.10 AES Performance in Software

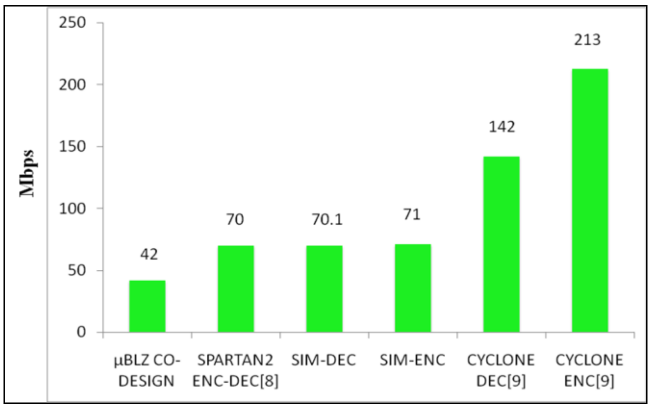


Fig.11 AES Performance in Hardware

Figure 10 shows the software performance in Mbps and Figure 11 shows AES performance in Hardware. These two performance straight forward indicates hardware performance is superior .

IV. CONCLUSION AND FUTURE SCOPE

As ARM7 and Microblaze both are RISC processor, and takes very less power, AES can be used for embedded application. Microblaze is soft core processor so scaling of the design can be possible to enhance speed and reduce power consumption. This processor can work on high frequency, so AES latency can be improved.

AES implemented on software hardware and software hardware co-design. From above graph it is clear that hardware performance is very much high in 40 above Mbps.

Software performance is below 1 Mbps. But Software hardware co design can be used for both benefit of microcontroller and speed advantage.

We have implemented AES engine on ARM 7 and microblaze processor. It can be evaluated for PowerPC and Nios like processor. Image Encryption and Decryption can be next work of this project.

V. REFERENCES:

[1] National Inst. Of Standards and Technology, “Federal Information Processing Standard Publication, the Advanced Encryption Standard (AES),” Nov. 2001

[2] J. Daemen and V. Rijmen,” AES Proposal: Rijndael,” AES Algorithm Submission, Sept. 1999.

[3] Wayne Wolf, “FPGA-Based System Design, Pearson Education,pp. 17-37

[4] Charles H Roth, Jr. Digital Systems Design Using VHDL, Thomson, India Edition 2007.

[5] Spartan-3 FPGA Family Data Sheet http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf

[6] MicroBlaze overview. http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_2/mb_ref_guide.pdf

[7] ARM7 overview. http://www.freescale.com/files/rf_if/doc/ref_manual/MC1322xRM.pdf

[8] Chi-Wu Huang1, Chi-Jeng Chang1, Mao-Yuan Lin1, Hung-Yun Tai2 ,” Compact FPGA implementation of 32-bits AES Algorithm Using Block RAM ” 1-4244-1272-2/07/ 1007 IEEE

[9] Yang Jun ,Ding Jun, Li Na, Guo Yixiong, “FPGA-based design and implementation of reduced AES algorithm” 2010 International Conference on Challenges in Environmental Science and Computer Engineering 2010 IEEE.