RESEARCH ARTICLE OPEN ACCESS

FPGA Implementation of Efficient Advance Encryption Standard (AES) With Key Expansion Technique

Ramesh S[1]

[1]Department of Electronics and Communication Engineering, Sankar Polytechnic College, Tirunelveli.

Abstract:

The paper presents an efficient reconfigurable hardware implementation of Advance Encryption Standard (AES) algorithm on Field Programmable Gate Array (FPGA); using High Level Language (HLL) approach with less hardware resources. The FPGA platform used for AES implementation is Xilinx Atlys Virtex-6. Time-to-market is one of the key factors for any design in FPGA and digital system designing industry. This time can be reduced considerably with HLL approach. The presented algorithm is designed on a HLL tool, namely Xilinx system generator. It is very user friendly despite giving detailed control in designing the required system design. For actual testing and hardware implementation of the algorithm, the HLL-tool generates a bit file that can be directly burnt on the FPGA. To get the implementation of design on hardware, the presented work uses a similar approach to directly map the System Generator described design on FPGA. The presented work emphasizes on optimization for less hardware utilization. The presented design uses approximately just one thousand slices and about half a century of BRAMs.

Keywords — **AES; System Generator; FPGA; reconfigurable computing; HLL.**

I. Introduction

An important aspect to be considered with the evolution of internet in the current information age is secrecy and privacy. Cryptography provides confidentiality and reliability to data during communication. It is used in different application which includes e-commerce, wireless communications, cellular networks, online banking, computerized networks etc. Cryptography is related to the study of secret writing i.e. conversion of plaintext into cipher-text [1], so that the information can only be retrieved by the desired entity over an unsecured channel. The cipher text cannot be transform into intelligible form (plaintext) unless receiver has a cipher key. Since a few decades, digital hardware design technology has become more similar to software design and has evolved tremendously with the introduction of reconfigurable platforms like FPGA [2]. The reconfigurable platform provides perfect customization of the hardware with time and cost effective solutions unlike Application Specific Integrated Circuit (ASIC) [3]. ASIC belongs to configurable platform but it configures permanently and provides high performance for a specific application. However, ASICs do not provide hardware reconfiguration flexibility. Whereas, software provides reprogrammable flexibility for different applications but lacks in performance and efficiency as compared to ASICs. The reconfigurable platform like FPGAfills the gap to achieve a balance between hardware and software in terms of performance and flexibility. FPGA provides improved performance than software implementation; and it can also be reconfigured.

It executes the hardware design efficiently over software by minimizing the time required to process the algorithm. Due to the merits described, FPGAs can be considered to implement the cryptographic algorithms [4]. The presented work shows efficient implementation of AES algorithm using High Level

Language (HLL) approach i.e. Xilinx System Generator [5] on FPGA. The proposed FPGA platform for the implementation of this work is Xilinx Atlys Vertix-6 [6]. The reconfigurable platform using system generator provides the better way in designing of hardware. System generator has environment similar to Simulink in which Xilinx blocks are used in the architecture of hardware. It generates the file for synthesis and simulation; and also provides access to FPGA blocks used in the design. The paper is organized as follows. Section 2 contains the explanation of Advance Encryption Standard, Section 3 gives description of proposed AES implementation using Xilinx System Generator and Section 4 is on the synthesis and results obtained from the presented work. Finally, the conclusion from the work and results obtained is presented in Section 5.

II. Proposed Method:

2.1 Design considerations

The Rijindael algorithm is selected for Advanced Encryption Standard over Data encryption standard and was published by NIST-National Institute of Standards and Technology as FIPS PUB 197, in November 2001. The AES handles 128 bit block of data with variable length of the key size 128, 192, 256 bits. The number of rounds depends on the selection of key size i.e. 10, 12 or 14 rounds for key size 128, 192 and 256 bits, respectively.

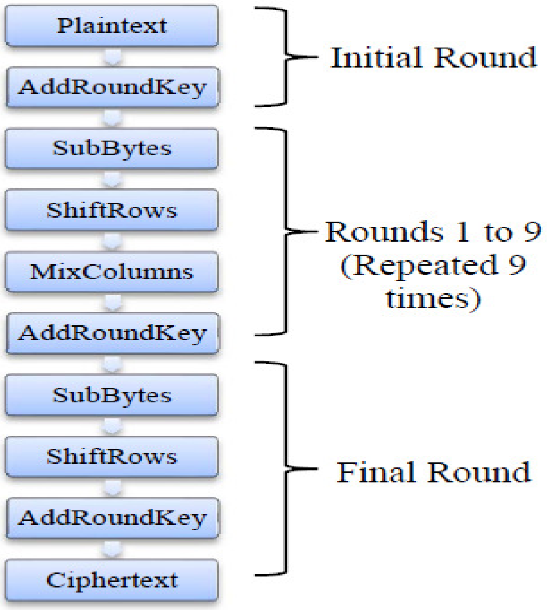


Figure 1. AES rounds

2.2 Implementation

The proposed AES encryption function is designed and implemented using the Xilinx System Generator for MATLAB. The figure 2 shows the outline of the structure. The implementation uses a pipelined architecture, as shown in figure 4; which is most commonly used reconfigurable architectures for implementation of encryption functions. Xilinx System Generator for MATLABprovides flexibility in design and scalability in FPGA chipselection.

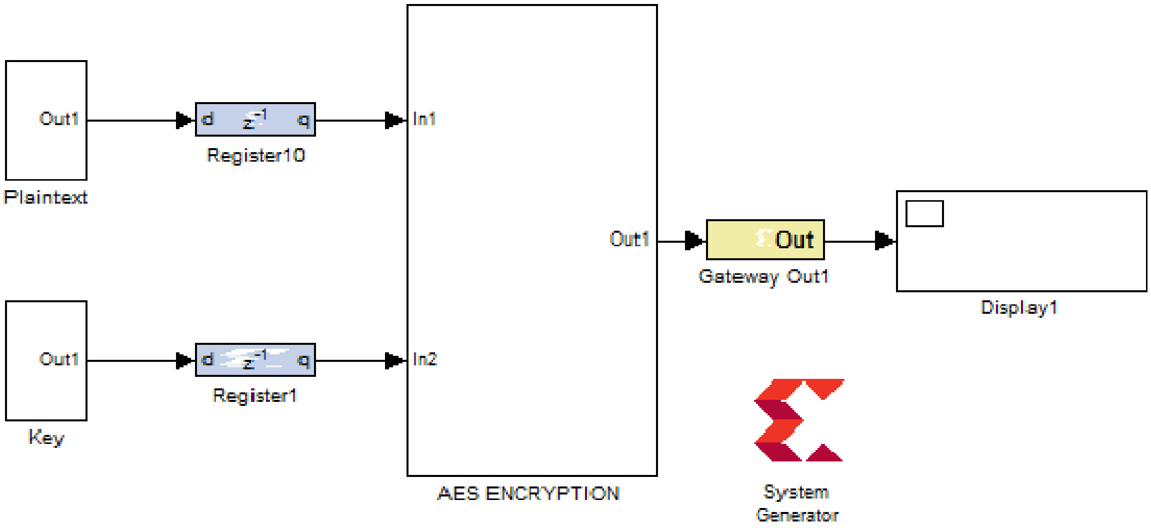


Figure 2. Proposed AES encryption function block

It is a pipeline architecture of AES-128 encryption function which consist of 10 rounds. Each round isimplemented separately enclosed in a subsystem, comprising of five transformations i.e. Sub Bytes, Shift Rows, Mix Columns, Add Round Key and Key Generation shown in figure 3; where Mix Columns is eliminated in the final round. The initial round is just Add Round Key transformation in which input state is XOR-ed with the initial round key. Registers are placed in the algorithm at the end of each round for better performance. The Plaintext and Key are defined in separate subsystems each as shown in figure 2; in which each column of the state can be defined separately as shown in Figure 3. This allows for easy handling of inputs.

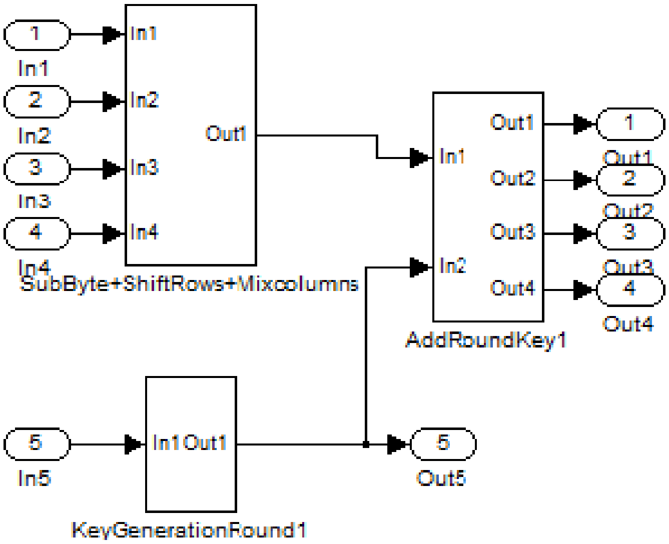


Figure 3. Structure of each round

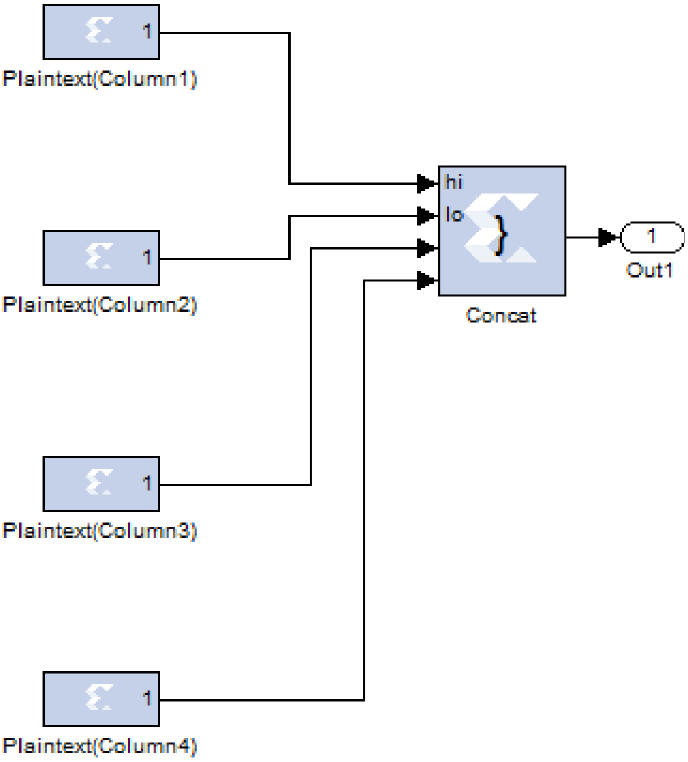


Figure 4. Sub-System of Plain-text

2.1.1 Sub bytes and Shiftrows:

There are two basic methods for generating SubBytes of AES, either by using multiplicative inverse or by using memory. Here we used BRAM method to implement AES SubBytes with multiplexing technique in order to make it more efficient in term of resource utilization. An input of 28-bits arranged in four words of 32-bits is given directly to the SubBytes block. The details of proposed SubBytes architecture are shown in figure 5. The scheme uses memory efficient SubBytes with the help of Counter, Mux Blocks, Dual-Port RAM and Time Division Demultiplex Blocks from Xilinx System Generator. We used Dual-Port RAM in order to store the 256 lookup values. The Counter is used by Mux Block to select from inputs accordingly. The Mux Block selects the desired input on the basis of counter value which is then connectedto Dual Port RAM.The Dual-Port RAM is configured as Block RAM to access 8-bit lookup values corresponding to the 8-bit input addresses. Dual-Port RAM is operated in “no read on write” mode. The input bytes are delivered to the address pins addra [7:0] and addrb[7:0] of BRAM while corresponding lookup values will be taken from the output pins A [7:0] and B [7:0] of the Dual-Port RAM. Input of constant zero is given to the data input and write enable pins of the RAM as they are not required in our architecture. In order to extract 8-bit data from 32-bit input, Bit Basher Block has been used as shown in Fig.8. We are able to extract specific 8-bit data required for Dual-Port RAM’s input addresses. Shifting is achieved by simply rearranging the wires in Round module as marked.

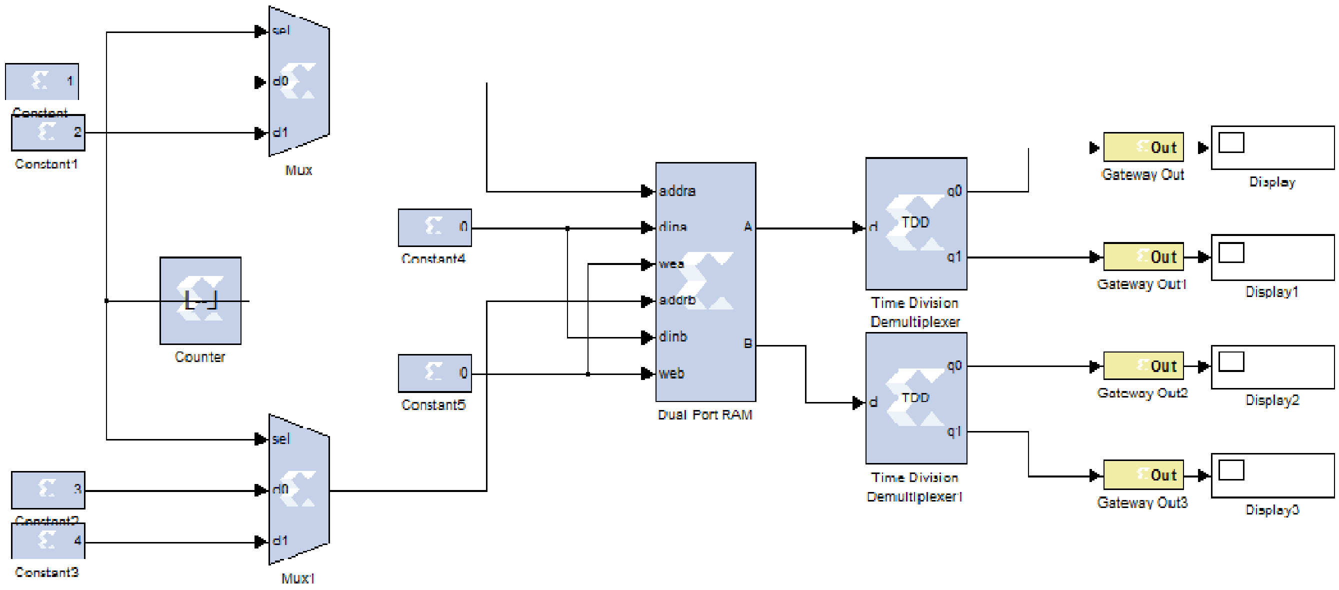


Figure5: System generator sub bytes modules

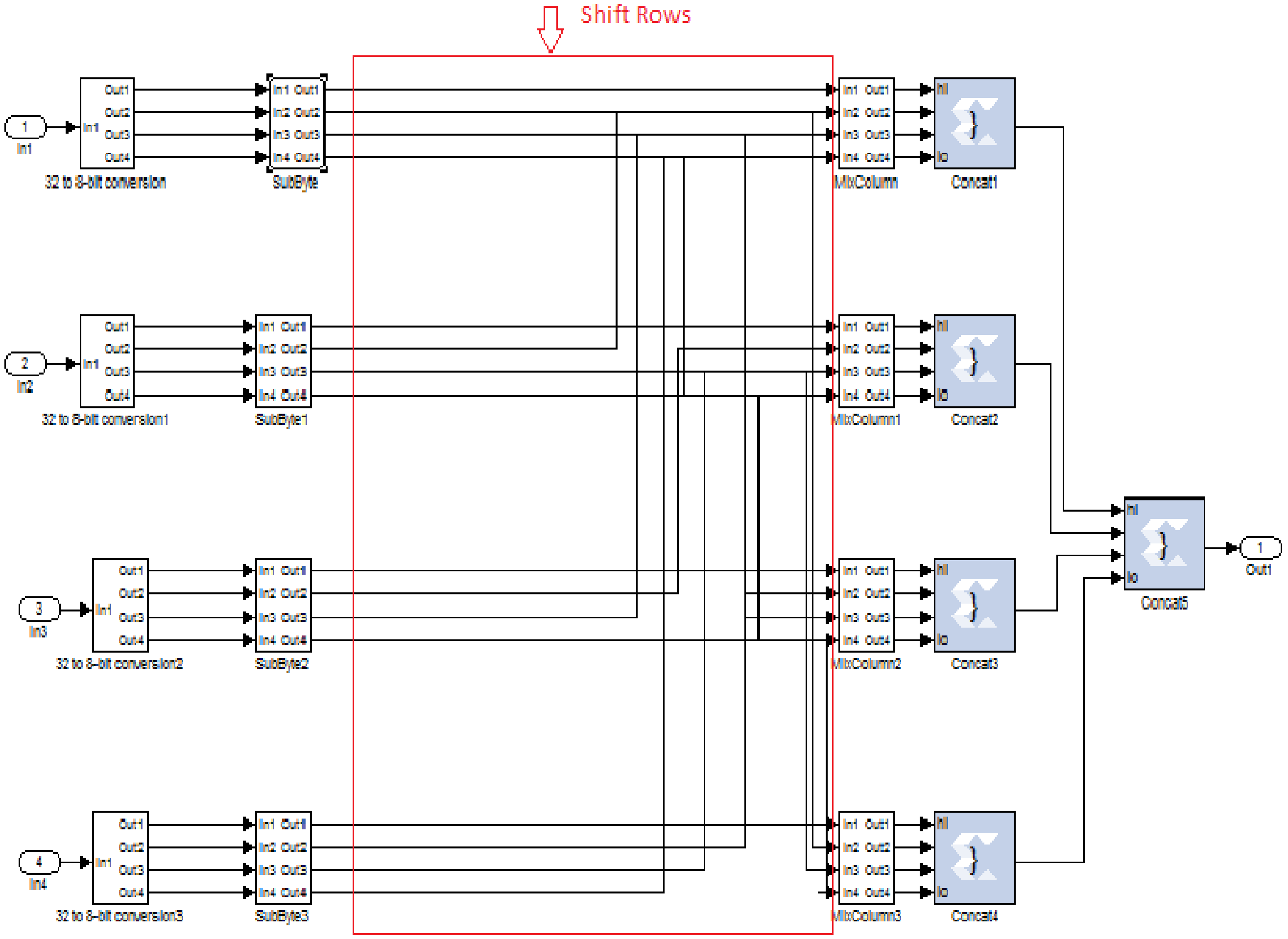


Figure6. System Generator Shift Rows structure

2.1.2 Mix Columns:

In Mix Columns block, input data is multiplied with a constant matrix consisting of three numbers 1, 2 and 3 only. The design laid down to perform this operation uses data to directly pass through connection wire for multiplication with 1. Multipliers of "2" and "3" are designed to carry out the multiplication. The architecture of Mix Columns is shown in figure 7. Rather than using conventional multiplications, the shift and add method has been applied for matrices multiplication; by u sing shift block to minimize the utilization cost. Multiplier of 2 is made using shift block where left shift is applied to data which results in the multiplication by number “2” as shown in figure 10. It should satisfy the condition of irreducible polynomial, that the number should be less than 255. So, if the result exceeds from number 255, then mod 27(in decimal) or mod 1b (in hexa-decimal) is applied on the results of multiplication. It is done so by using bitwise XOR operator in order to get the number within the range of 255 as shown in figure 8.

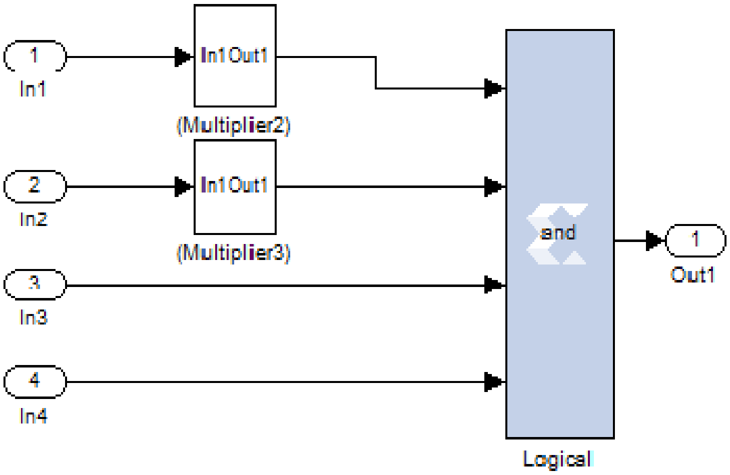


Figure 7. Mix-column structure

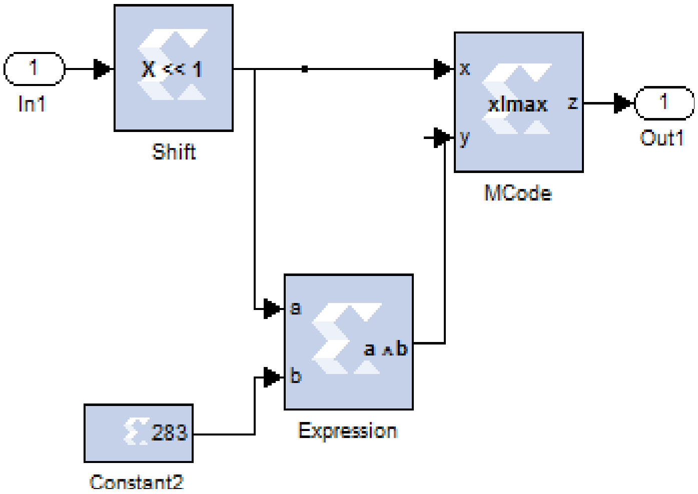


Figure 8. Multiplier2 internal structure

Multiplier of 3 is designed by dividing the number ‘3” into (2+1) where multiplication by number “2” is done by "Multiplier 2" and the resultant number is added with the state matrix by using XOR operation as shown in figure 9.

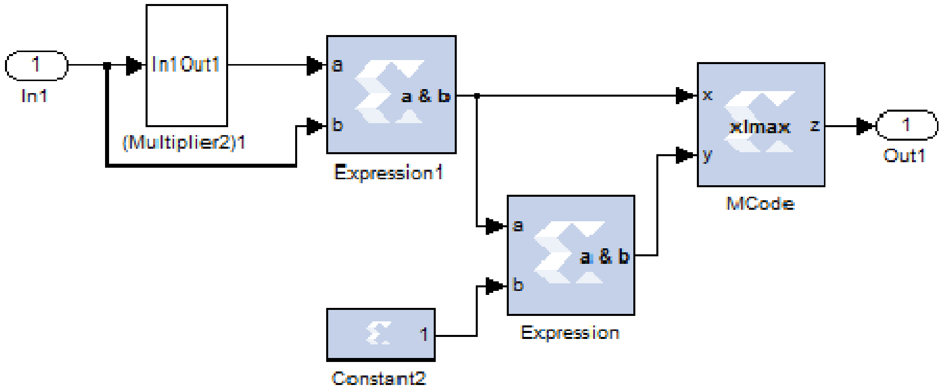


Figure 9. Multiplier 3 internal structure

2.1.3 Round Key Generation

In Round key, firstly 128 to 8 bit conversion is carried out with BitBasher block. The first word (1st Column) of 32-bit of Round key is generated by substitution of last word(4th column) of key state matrix using SubBytes block. The result of SubBytes is then rotated by simply rearranging the connecting wires. The rotated word is XORed with 1st column of key and round constant for the generation of 1st word of Round Key, as shown in figure 10. As per AES Algorithm, the 1st word of Round key is then
used to generate other words by using XOR operation.

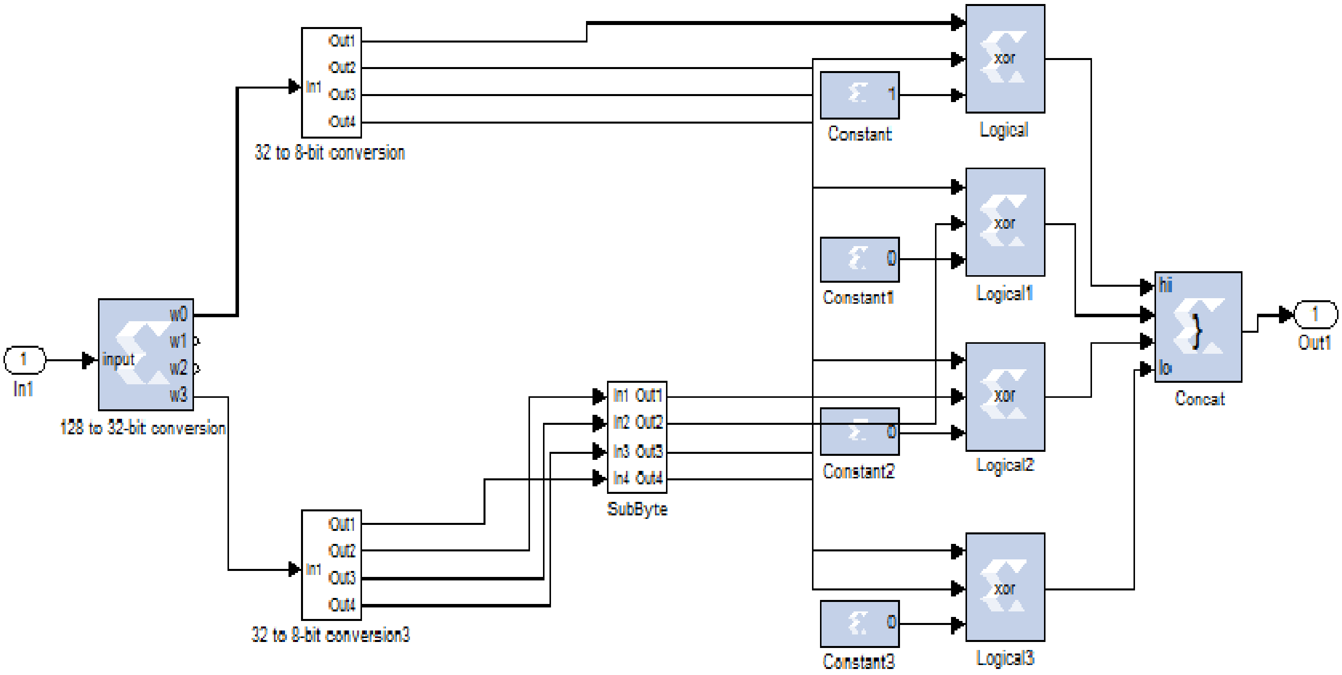


Figure.10. 1st word generation of round key

2.1.4 Add Round Key:

In AddRoundKey, Bitwise XORing between result from MixColumns and RoundKey is done. Here the expression blocks are used for XOR operation. Also BitBasher for 128 to 32 bit and 32 to 8 bit conversion is used as shown in figure 11.

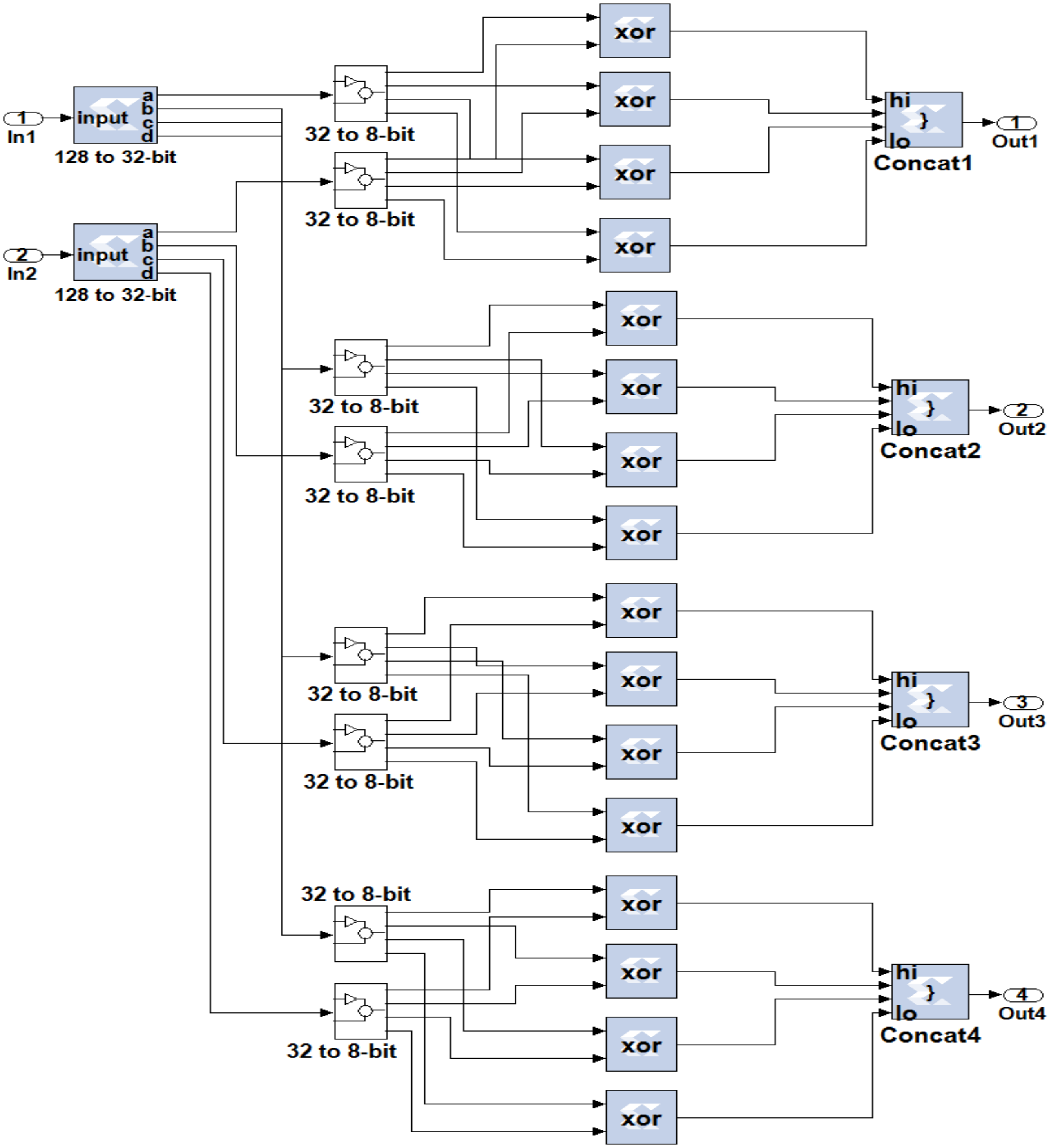
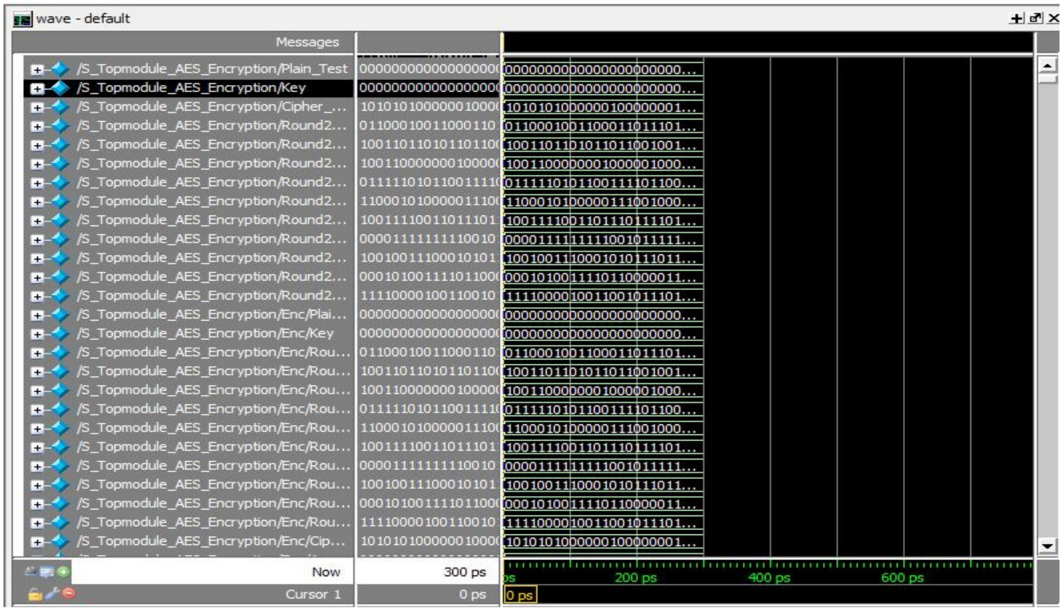


Figure 11: Add round key internal structure

III. Result:

The hardware implementation results are targeted for Xilinx Virtex-6 xc6vsx315t-3ff1156 FPGA. The design has been implemented using Xilinx System Generator tool in MATLAB and the generated verilog code (.v file) and testbench are then synthesized and simulated using Xilinx ISE Foundation 14.1 and Mentor Graphic ModelSim, respectively. The design occupies 50 BRAM's and 1002 numbers of slices out of 49200 (2%). It operates on 254.453MHz frequency and offers latency of 3.930ns.



```
wave - default
Messages
/S_Topmodule_AES_Encryption/Plain_Test   0000000000000000000   0000000000000000000000...
/S_Topmodule_AES_Encryption/Key          0000000000000000000   0000000000000000000000...
/S_Topmodule_AES_Encryption/Cipher_...   1010101000000010000   1010101000000010000001...
/S_Topmodule_AES_Encryption/Round2...    011000100110001101    0110001001100011011101...
/S_Topmodule_AES_Encryption/Round2...    1001101101011011001   1001101101011011001001...
/S_Topmodule_AES_Encryption/Round2...    1001100000000100000   1001100000000100001000...
/S_Topmodule_AES_Encryption/Round2...    0111110101100111110   0111110101100111101100...
/S_Topmodule_AES_Encryption/Round2...    1100010100000111001   1100010100000111001000...
/S_Topmodule_AES_Encryption/Round2...    1001111001101110111   1001111001101110111101...
/S_Topmodule_AES_Encryption/Round2...    0000111111111001011   0000111111111001011111...
/S_Topmodule_AES_Encryption/Round2...    1001001110001010111   1001001110001010111011...
/S_Topmodule_AES_Encryption/Round2...    0001010011110110000   0001010011110110000011...
/S_Topmodule_AES_Encryption/Round2...    1111000010011001011   1111000010011001011101...
/S_Topmodule_AES_Encryption/Enc/Plai...  0000000000000000000   0000000000000000000000...
/S_Topmodule_AES_Encryption/Enc/Key      0000000000000000000   0000000000000000000000...
/S_Topmodule_AES_Encryption/Enc/Rou...   011000100110001101    0110001001100011011101...
/S_Topmodule_AES_Encryption/Enc/Rou...   1001101101011011001   1001101101011011001001...
/S_Topmodule_AES_Encryption/Enc/Rou...   1001100000000100000   1001100000000100001000...
/S_Topmodule_AES_Encryption/Enc/Rou...   0111110101100111110   0111110101100111101100...
/S_Topmodule_AES_Encryption/Enc/Rou...   1100010100000111001   1100010100000111001000...
/S_Topmodule_AES_Encryption/Enc/Rou...   1001111001101110111   1001111001101110111101...
/S_Topmodule_AES_Encryption/Enc/Rou...   0000111111111001011   0000111111111001011111...
/S_Topmodule_AES_Encryption/Enc/Rou...   1001001110001010111   1001001110001010111011...
/S_Topmodule_AES_Encryption/Enc/Rou...   0001010011110110000   0001010011110110000011...
/S_Topmodule_AES_Encryption/Enc/Rou...   1111000010011001011   1111000010011001011101...
/S_Topmodule_AES_Encryption/Enc/Cip...   1010101000000010000   1010101000000010000001...
Now        300 ps
Cursor 1   0 ps     0 ps
```

Figure 12. Generated output in modelsim

Table 1. Comparison of FPGAs

Design	Device	Frequency MHz	Latency (ns)	Slices
[14] design	Virtex-E XCV1000e-8	129.2	-	11,719
[15] design	Virtex-E XCV2000e-8	158	-	5810 + 100BRAM
[16] Design	Virtex-E XCV3200e-8	145	-	15,112
[12] Dsign (Excl. key expansion)	Virtex-II Pro XC2VP20	169.1	420	9,446 Excl. KE
[13] Design	Virtex-II XC2V4000	184.1	163	16,938
[11] Design no key expand	Virtex-E XCV1000E8	168.4	416	11,022 Excl. KE
Proposed scheme	Virtex-6 xc6 vsx315t	254.453	3.930	1002 + 50BRAM

Table 2. Final results

Parameters	Observation
Number Of BRAMs	50
Number Of Slices	1002 out of 49200 (2%)
Latency	3.930ns

IV .Conclusions:

The paper presents reconfigurable platform used with Highlevel language approach and the work presented here uses efficient implementation of AES using Xilinx System Generator; the approach not only reduces the overall utilization but also gives good enough clock frequency and latency. This paper shows performance comparison to the various FPGA (Verilog) implementations. It is the user friendly design for HLL users and gives fast design to market.

References:

1. William Stalling, “Cryptography and NetworkSecurityPrinciplesand Practices”, Prentice Hall, sixth edition, 2013

2. L. Floyd, “Digital Fundamental with VHDL,” Pearson Education, pp.362-368, 2003

3. Data Encryption Standard (DES), FIPS PUB 46-3, October 1999 J.Zambreno, D.Honbo, A.Choudhary, R.Simha, and B.Narahari, “High performance Software Protection Using Reconfigurable
4. Architectures”, Proceedings of the IEEE, volume 94, No. 2, February 2006.

5. N. A. Saqib, C. K. Koc, A .D. Pérez, F. Rodriguez-Henriquez, “Cryptographic Algorithms on Reconfigurable Hardware”, Signals and Communication Technology, Springer, vol. 26, pp. 362, 2007.

[1]**RAMESH S** received the B.E. Degree – First Class - in Electronics and Communication Engineering from National Engineering College, K.R.Nagar, Nalattinputhur, Madurai Kamaraj University, Madurai, India in 1993 M.E degree – First Class with Distinction - in Computer Science Engineering from ManonmaniamSundaranar University, Tirunelveli, India in 2004 Now he is working as a Sr. Lecturer in Sankar Polytechnic College Tirunelveli.