International Journal For Technological Research In Engineering
Volume 1, Issue 3, November - 2013 *ISSN (Online) : 2347 - 4718*

Advanced Encryption Standard - VHDL Implementation

M. Komala Subhadra[1]
[1]Department of Electronics and Communication Engineering
Sree Nidhi Institute of Science and Technology(SNIST)
Hyderabad, Andhra Pradesh, India

***Abstract:** The new Advanced Encryption Standard (AES) has been recently selected by the US government for protecting sensitive social information. Due to its simplicity and elegant algebraic structure, the choice of the AES algorithm has motivated the study of a new approach to the analysis of block ciphers.*
The Advanced Encryption Standard can be programmed in software or built with pure hardware. However Field Programmable Gate Arrays (FPGAs) offer a quicker, more customizable solution. Field Programmable Gate Array (FPGA) serves perfectly the cryptosystems in high speed communications links or servers.
This paper proposes an efficient FPGA implementation of AES using VHDL. An AES encryptor is designed and implemented in FPGA. An AES decryptor is also designed and integrated with the AES encryptor to yield a full functional AES en/decyptor. VHDL stands for Very High Speed Integrated Circuit Hardware Description Language. Xilinx software is used for the simulation and optimization of the synthesizable VHDL code. All the transformations of both Encryption and Decryption are simulated using an iterative design approach in order to minimize the hardware consumption.

***Keywords:** Encryption, Key Generation, FPGA, VHDL, Decryption.*

I. INTRODUCTION

The Advanced Encryption Standard which will be referred to as AES is the current industrial standard and has been in vogue since 2001. It is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. It is based on the Rijndael cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal which was evaluated by the NIST during the AES selection process

In cryptography, encryption is the process of encoding messages (or information) in such a way that hackers cannot read it, but that authorized parties can. In an encryption scheme, the message or information (referred to as plaintext) is encrypted using an encryption algorithm, turning it into an unreadable ciphertext. This is usually done with the use of an encryption key, which specifies how the message is to be encoded. Any adversary that can see the ciphertext should not be able to determine anything about the original message. An authorized party, however, is able to decode the ciphertext using a decryption algorithm, that usually requires a secret decryption key, that adversaries do not have access to.

The AES encryption method finds extensive use in the electronic and computational industry as most of the arithmetic operations that we generally use are not the ones used in here rather its the ones which are highly electronic efficient and can be implemented using shift registers and exclusive OR gates which any processor is efficient in handling. This makes the complicated algorithm run very quickly and using minimal processor power and minimal hardware.

II. DESCRIPTION OF THE ALGORITHM

The AES algorithm is a symmetric-key scheme. In symmetric-key schemes, the encryption and decryption keys are the same. Thus communicating parties must agree on a secret key before they wish to communicate. It is also known as private-key scheme.

AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. AES operates on a 4ÃŮ4 order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state.

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The number of cycles of repetition are as follows:

a) 10 cycles of repetition for 128-bit keys.
b) 12 cycles of repetition for 192-bit keys.
c) 14 cycles of repetition for 256-bit keys.

Here, The algorithm is used in encrypting data which is stored in blocks of 128-bit (here) and is encrypted with a 128-bit cipher key which is essential to decrypt the information locked in the encrypted data.

The Encryption program needs two pieces of inputs one, is the input data which is to be encrypted and the other being the Cipher key with which the information will be locked . The Algorithm is in turn divided into two distinct parts :

a) Encyption
b) Key Generation

We will first look into the Encryption Algorithm then Key generation followed by the Decryption Algorithm.

A. Encryption Algorithm

The Encryption process contains 10 rounds and each round contains 4 different transformations at the end of the 10 rounds the encrypted values are produced. The Transformations are :

1. Substitution Bytes
2. Shift Rows
3. Mix Columns
4. Add Round Key

Each Round need not use all the transformations. The first round does only the 4th transformation while the last round that is round 10 does only three transformations.

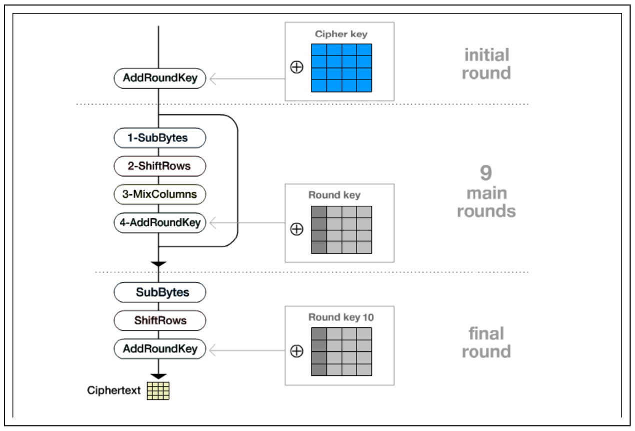


Figure 1: AES Encryption Process

1. Substitution Bytes: This is the first transformation to be done on the input data i.e, the input matrix.This step is also known as SubBytes.In the SubBytes step, each byte in the array is updated using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over g f (28), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points.

   The Affine Transformation is used to generate the Sub Box which is the lookup table from which the values are substituted.

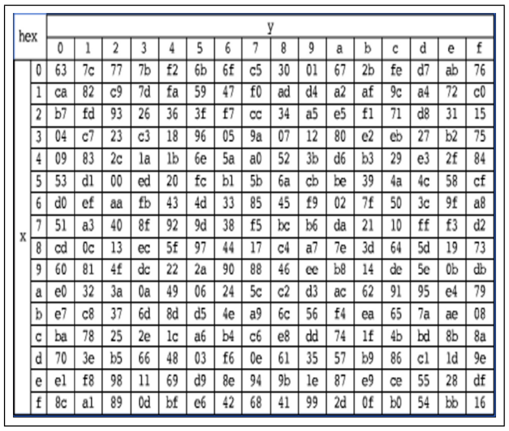
   The procedure to be followed to substitute the bytes in the matrix are :

   a) Select any element say 19 which is in hexadecimal notation.

   b) The S-box element in the 1st row and 9th column is to be selected and substituted in its place (i.e., d4).

   c) Similarly for any element in the matrix.

2. Shift Rows: This is the Second Transformation in the series of 4 Transformations and is extremely simple to implement. It involves rotating the rows of the input matrix circularly upwards.



hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 2: Substitution Box (S-Box)

3. Mix Columns: This is probably the most complicated of the 4 Transformations in the Algorithm and involves the Modulo Matrix Multiplication of the input matrix with the following matrix ;

$$\begin{matrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{matrix}$$

   Each column is multiplied with the matrix and the answer thus obtained will replace the input data and will go ahead to undergo further transformations in the consecutive rounds.

4. Add Round Key: This is the final transformation in which the Round Key which is specific to the particular round is added to the result of the above three transformations, Addition in this case is again Modulo addition which is simply the bit-wise XOR of the participating values. Round Key in the case of the first Round is simply the Cipher Key and in the other Rounds is the Key generated specifically for that particular round using the Key Generation Algorithm.

   The Add Round key in the current situation in which we are only considering the first round thereby the round key is just the cipher key.

B. Key Generation Algorithm

There are a set of 11 Rounds in the Encryption process counting the initial round also, and every Round without fail follows the Add Round Key Transformation. Hence we need a total of 10 keys which are to be generated from the single key given as the input, generating the required keys is known as the key schedule.

The following are the steps involved in the process of generating the Keys required

Write down the entire sequence of 11 keys we need a matrix which has 4 rows and 44 columns wherein the Round keys have to be calculated. Consider the Cipher key below :

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

The entire Sequence of keys when written out in a matrix form with the first column numbered 0 to last column being numbered 43 can be divided into two distinct groups i.e, multiples of 4 and non-multiples of 4.

1. Generating the columns C4m: These columns are filled in a different way i.e, in the case of the Multiples of 4 i.e,(C4;C8;C12:::::::). The following is the algorithm followed: Consider the Column which is to be calculated then the ones just before it will be Cm 1 this column has to undergo two different transformations :
   a) Rotation circularly upwards
   b) Substitution of bytes in the rotated column
   c) ubstitution of bytes in the rotated column.The 4*10 round constant matrix happens to be :

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

   d) To the resulting column the column Cm 4 in the key generating Sequence has to be added the result is the value of the column Cm .

   This completes the generation of the 4m Column.
2. Generating the Columns C4m+1;4m+2;4m+3: The next set of columns to be generated in this case are the non multiples of 4 which are relatively simple to generate and the process is as follows:

   The column to be generated if considered Cn then selecting the earlier column Cn 1 and adding it to the column Cn 4 will generate the rest of the columns.

   Following the same algorithm will generate the entire Key Sequence. Each of the key are used in the different Rounds in the corresponding Add Round Key Transformation in which the key will be added to the resulting matrix and the resulting answer is used as the input to the next Round.

C. Decryption Algorithm

The Decryption Algorithm follows the exact same transformations with their effect negated, the transformations are :

1. Inverse Substitution Bytes - Inverse SubBytes is simply a transformation in which the value in the input is looked up at the S-Box and the corresponding Row and Column are recorded and substituted in the place of the input matrix to give out the result.
2. Inverse Mix Columns - In this case the matrix with which the input matrix is to be Modulo multiplied is :

14	11	13	09
09	14	11	13
13	09	14	11
11	13	09	14

   Here the elements of the matrix are all in decimal notation and not in the traditional hexadecimal notation.
3. Inverse Shift Rows - This Transformation is again simply the inverse of its counter part where in the input matrix was circularly rotated to the left, in this case the input matrix's rows are rotated to the right.
4. Inverse Add Round key - The Transformation as its counterpart is simply the bit wise addition of the input with the correspondiRound Keys. The only thing that sets this Transformation apart from its counter in Encryption process is that, here in the first Round we add to the input the last four Columns of the Key sequence which is different from the Add Key in Encryption process where in the first round would add the input to the Cipher Key. In this case though we add the Cipher Key to the final step i.e, before we end up getting the final decrypted value .

III. VHDL IMPLEMENTATION OF AES

Few topics to be known before getting into the implementation are the following:

a) FPGA: A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing-hence "field- programmable". The FPGA configuration is generally specified using a hardware description language(HDL). Contemporary FPGAs have large resources of logic gates and RAM blocks to implement complex digital computations. The ability to update the functionality after shipping, partial re- configuration of a portion of the design and the low non- recurring engineering costs offer advantages for many applications. The FPGA used here is XC7K325T.

   FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"âĂŤsomewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

b) VHDL: VHDL (VHSIC Hardware Description Language) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits. VHDL can also be used as a general purpose parallel programming language. VHDL is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. A simulation program is used to test the logic design using simulation models to represent the logic circuits that interface to the design. This collection of simulation models is commonly called a testbench. VHDL has constructs to handle the parallelism inherent in hardware designs, VHDL is strongly typed and is not case sensitive. In order to directly represent operations which are common in hardware, there are many features of VHDL, such as an extended set of Boolean operators including nand and nor. VHDL also allows arrays to be indexed in either ascending or descending direction. One can design hardware in a VHDL IDE (for FPGA implementation such as Xilinx ISE, Altera Quartus, Synopsys Synplify or Mentor Graphics HDL Designer) to produce the RTL schematic of the desired circuit. After that, the generated schematic can be verified using simulation software which shows the waveforms of inputs and outputs of the circuit after generating the appropriate testbench. To generate an appropriate testbench for a particular circuit or VHDL code, the inputs have to be defined correctly. For example, for clock input, a loop process or an iterative statement is required. A final point is that when a VHDL model is translated into the "gates and wires" that are mapped onto a programmable logic device such as a CPLD or FPGA, then it is the actual hardware being configured, rather than the VHDL code being "executed" as if on some form of a processor chip.

c) Implementation: The industrial implementation of the Encryption process and the Decryption process is done in a FPGA and there by the code for the above process explained is written in VHDL which can later, be burnt onto a FPGA. I n this case there are 3 modules ,one for the Encryption process, one for Decryption and one final module which outputs the encrypted and decrypted values.

A. Encryption Algorithm

$$\begin{aligned} InitialRound: &\quad aes\_addkey \\ Rounds1to9: &\quad aes\_sbyte \\ &\quad aes\_shiftrow \\ &\quad aes\_mixclms \\ &\quad aes\_addkey \\ Round10: &\quad aes\_sbyte \\ &\quad aes\_shiftrow \\ &\quad aes\_mixclms \end{aligned}$$

The above is simply the outline and considering each transformation a function we can use for loops and if conditions to get the work done. The output of each transformation in each round is stored in an array and the output of the current round is used as the input of the next round .

1. Substitution bytes:
   aes sbyte: The input matrix contains elements which are each an 8 bit number, the algorithm is to split the number into the first four bits and the last four bits and thereby convert the value into hexadecimal and look up the s-box to give out the substitution value.

   Hardware implementation of this transformation would be storing the S-box in a ROM module and looking it up whenever necessary, or generating the values dynamically.
2. Shift Rows:
   aes shiftrow: This just involves moving the first row by 0 second by 1 third by 2 and fourth by 3 cells and hence as observed every nth row is to be moved by n-1. This transformation can be implemented using shift registers.
3. Mix Columns:
   aes mixclms : This transformation involves the multiplication of the input with the specified matrix :

$$\begin{matrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{matrix}$$

   And as it is known in the case of modulo multiplication, multiplication with 3 is simply not adding the number three times rather it is the shifting of the number once and then adding it onto itself as;

$$0011 = 0010+0001$$

   Where in multiplication with two will be shifting the number by one cell to the left and multiplication with one will yield the same number. The above matrix when observed is the result of shifting the first row circularly through the number of the row to the right. So we can as well keep the sequence 02 03 01 01 constant and rotate the Column under consideration. As every number in the Column will be multiplied with 2,3,1 the procedure followed is to genrate another matrix which contains the modulo multiplied values of the ones under consideration. This has been done because multiplication with 2 involves shifting(or sometimes the mod multiplication), multiplication with 3 involves mod multiplication with 2 and adding the number to itself, Similarly multiplication with 1 is just the number. Hence as it has been seen that every step involves shifting, a separate matrix which contains the modmultiplied values of the current values has been generated and added when required. Hence if A contains the input then B contains the shifted values(or in the modulo multiplied with 2 value which need not always be the shifted value) of every element in the same position in matrix B, then multiplication with 2 is the value of the element in same position in B. Multiplication with 3 is element in A + the corresponding element in B.

   The hardware implementation of the above is done using XOR gates and the shift registers which are used in the case of multiplication.
4. Add Round Key:
   aes addkey: The process is simply the XOR of the current matrix with the corresponding Round Key. Generation of the Round key will be dealt with in the Key generation Section. As for now consider the key sequence has been generated. This transformation can be implemented in the hardware scale using only XOR gates.

B. Key Generation Algorithm

1. Generating the columns C4m: On the hardware scale, it can be implemented by shift registers for Rotation Word Transformation where as the Substitution bytes transformation will be implemented by a ROM module with the S-box as look up table and XOR gates for the addition.

2. Generating the Columns C4m+1;4m+2;4m+3: This on the hardware scale can be implemented using XOR gates alone and registers to store the values.

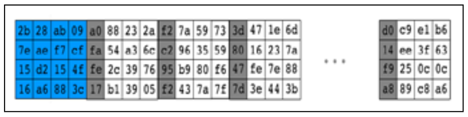

2b 28 ab 09 a0 88 23 2a f2 7a 59 73 3d 47 1e 6d
7e ae f7 cf fa 54 a3 6c c2 96 35 59 80 16 23 7a
15 d2 15 4f fe 2c 39 76 95 b9 80 f6 47 fe 7e 88
16 a6 88 3c 17 b1 39 05 f2 43 7a 7f 7d 3e 44 3b
. . .
d0 c9 e1 b6
14 ee 3f 63
f9 25 0c 0c
a8 89 c8 a6

Figure 3: Complete Key Sequence

C. Decryption Algorithm

The decryption algorithm and process is the inverse of its counterpart.

1. Inverse substitution bytes : On the hardware scale this is implemented using a ROM module which can have the lookup table and will give out Row and Column of the input which will be substituted in the place of the input.
2. Inverse mix columns: This can be carried out using shift registers for multiplication and XOR gates for the addition.
3. Inverse shift rows: This can also be implemented on the hardware scale using shift registers.
4. Inverse add round key: This transformation can be implemented by the use of XOR gates for addition.

D. Implementation Output - Inspection

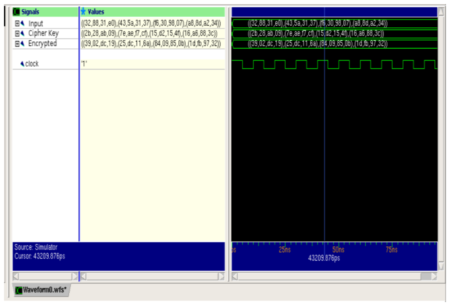
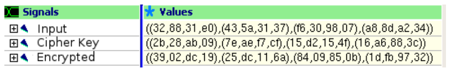

Signals | Values
Input ((32,88,31,e0),(43,5a,31,37),(f6,30,98,07),(a8,8d,a2,34))
Cipher Key ((2b,28,ab,09),(7e,ae,f7,cf),(15,d2,15,4f),(16,a6,88,3c))
Encrypted ((39,02,dc,19),(25,dc,11,6a),(84,09,85,0b),(1d,fb,97,32))
clock '1'
Source: Simulator
Cursor: 43209.876ps
Waveform0.wfs*

Figure 4: Simulator Screenshot for Encryption



Signals	Values
Input	((32,88,31,e0),(43,5a,31,37),(f6,30,98,07),(a8,8d,a2,34))
Cipher Key	((2b,28,ab,09),(7e,ae,f7,cf),(15,d2,15,4f),(16,a6,88,3c))
Encrypted	((39,02,dc,19),(25,dc,11,6a),(84,09,85,0b),(1d,fb,97,32))

Figure 5: Encryption

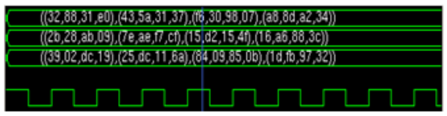

((32,88,31,e0),(43,5a,31,37),(f6,30,98,07),(a8,8d,a2,34))
((2b,28,ab,09),(7e,ae,f7,cf),(15,d2,15,4f),(16,a6,88,3c))
((39,02,dc,19),(25,dc,11,6a),(84,09,85,0b),(1d,fb,97,32))

Figure 6: Encryption bit wise

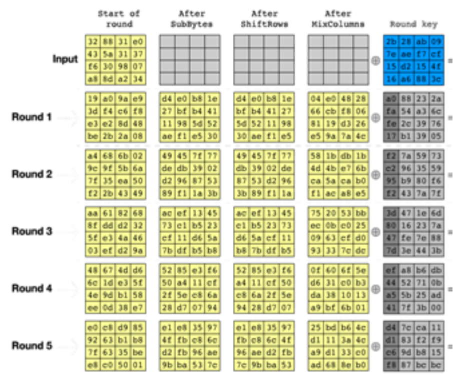


Figure 7: Encryption process (Round 0 to 5)

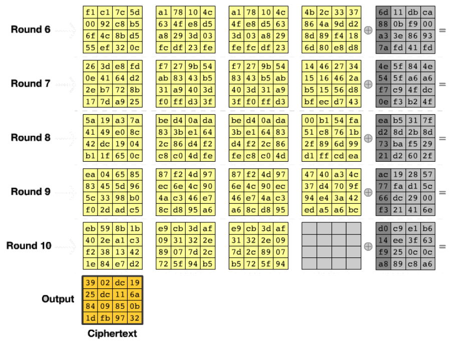


Figure 8: Encryption Process (Round 6 to 10)

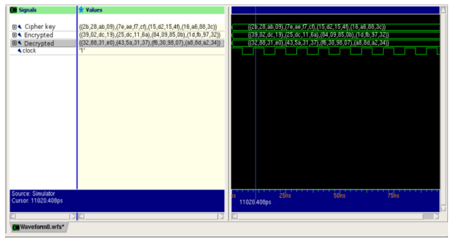
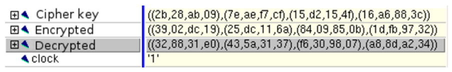

Signals | Values
Cipher key ((2b,28,ab,09),(7e,ae,f7,cf),(15,d2,15,4f),(16,a6,88,3c))
Encrypted ((39,02,dc,19),(25,dc,11,6a),(84,09,85,0b),(1d,fb,97,32))
Decrypted ((32,88,31,e0),(43,5a,31,37),(f6,30,98,07),(a8,8d,a2,34))
clock '1'
Source: Simulator
Cursor: 11020.408ps
Waveform0.wfs*

Figure 9: Simulator Screenshot for Decryption



Cipher key	((2b,28,ab,09),(7e,ae,f7,cf),(15,d2,15,4f),(16,a6,88,3c))
Encrypted	((39,02,dc,19),(25,dc,11,6a),(84,09,85,0b),(1d,fb,97,32))
Decrypted	((32,88,31,e0),(43,5a,31,37),(f6,30,98,07),(a8,8d,a2,34))
clock	'1'

Figure 10: Decryption

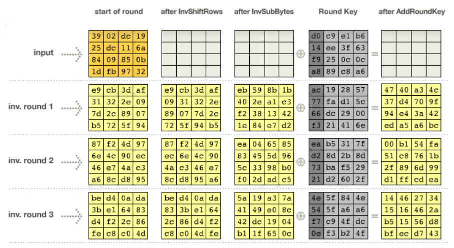


Figure 11: Decryption process (Round 0 to 3)

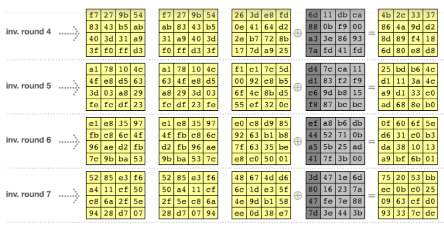


Figure 12: Decryption process (Round 4 to 7)

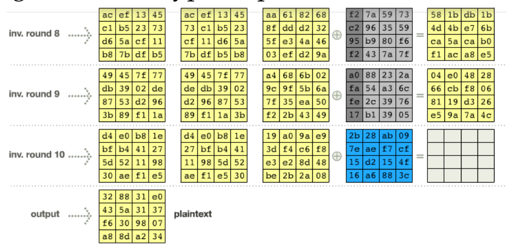


Figure 13: Decryption process (Round 8 to 10)

IV. Conclusion

Security is no longer an afterthought in anyone's software design and development process. AES is an important advance and using and understanding it will greatly increase the reliability and safety of your software systems. Because of the advantages of FPGA like the ability to update the functionality after shipping, partial re-configuration of a portion of the design and the low non-recurring engineering costs make it convenient to be used for the following application.

References

[1] `en.wikipedia.org/wiki/Advanced_Encryption_Standard_process`.

[2] `csrc.nist.gov/publications/fips/fips/197/fips-197.pdf`.

[3] `en.wikipedia.org/wiki/Cryptanalysis`.

[4] `www.formaestudio.com/rijndaelinspector/`.

[5] `blog.ultrassecreto.com/wpcontent/uploads/2009/06/projectofinal.html`.

[6] `www.x-n2o.com/aes-explained/`.

[7] `www.ensilica.com/pfs/A_study_of_aes_and_its_efficient_implementation_on_eSi_RISC_r1.0.pdf`.

[8] `msdn.microsoft.com/en-us/magazine/cc164055.aspx#S9`.