Design And Implementation of Reconfigurable Rijndael Encryption Algorithms For Reconfigurable Mobile Terminals

L.Thulasimani
Lecturer Department of Electronics and Communication Engineering
PSG College of Technology, Coimbatore
l_thulasi@yahoo.com; lthulasi@gmail.com

M.Madheswaran
Principal
Muthayammal Engineering College, Rasipuram
Madheswaran.dr@gmail

ABSTRACT:

In any wireless communication security is crucial during data transmission. The encryption and decryption of data is the major role in the wireless communication for security of the data. Encryption algorithms are used to ensure the security in the transmission channels. Similarly the area and the power consumption is another major thing to be viewed since most of the mobile terminals are battery operated. So a mobile terminal which has an encryption unit with less area and power consumption is appreciated. This paper deals with the Advanced Encryption Standard (AES) which works on a 128 bit data encrypting it with 128,192,256 bits of keys (ciphers) in a single hardware unit.

Key words: Rijindael Cipher, reconfiguration,encryption, decryption

I.INTRODUCTION:

In wireless platform AES has many applications for example blackberry enterprise solutions offer two wireless securities in thier smart phones and one of them is AES. AES is a WPA/802 11i encryption protocol used along with WEP2 is used with many encryption standards. In [2] Abdul Samiah, Arshad Aziz and Nassar Ikram implemented the AES algorithm in software using C-programming via Dev-C++4.9.9.1 compiler but software version of AES algorithm gives more payload for CPU thus it is slower when compared with hardware versions. In [3] Lia Huai, Xuecheng Zou done the hardware implementation for AES CCM using 128-bit data and 128-bit key thus he updated the work by offloading the CPU payload and using a dedicated hardware kit. But here the hardware implementation is done only for 128-bit key if we need 192,256 bit keys we need to design another dedicated hardware which is the waste of hardware and power used to activate it. The proposed work is to develop a hardware architecture that can be reconfigured to three different keys (128-bit, 192-bit, 256-bit) with a 128-bit data input and 128-bit data output is developed and the flow chart is shown in fig 1. In this paper the proposed work in which encryption process and decryption are explained with the flow of steps along with thier algorithms which are iteratively used for encrypting and decrypting and the results along with discussions are presented.

II.DESCRIPTION OF AES ALGORITHM:

Advanced Encryption Standard is the successor of Data Encryption Standard which was in use during the early 1977 to 1990. In DES encryption is based on a symmetric key algorithm that uses a 56-bit key. However by the mid 1990's, it was clear that the DES with 56-bit is insecure for many applications since the key is very small. Then it was upgraded to Triple DES which was believed to be practically secure although there are theoretical attacks. Thus in Nov-26-2001 the FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION 197(FIPS 197) specifies an algorithm called Advanced Encryption Standard (AES). AES is based on the principle known as Substitution Permutation network (SP-network) which means there will be a series of linked mathematical operations in the block cipher algorithm.AES encrypts a data block of 128-

bits which is fixed with three different key sizes 128,192,256 bits. The operations are based on Rinjdael algorithm. The input of AES algorithm is 128-bit or 16 byte data which can be specified as a block. The basic unit of processing in the AES algorithm is a byte. All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between the braces in the order (b7, b6, b5, b4, b3, b2, b1, b0). These bytes are interpreted as finite field elements using a polynomial representation as follows

$$b_7X^7+ b_6X^6+ b_5X^5+ b_4X^4+ b_3X^3+ b_2X^2+ b_1X+b_0 = \sum_{i=0}^{7} X^i b_i$$

Internally in AES algorithm operations are performed on a two-dimensional array of bytes called the state. The state consists of four rows of bytes, each containing Nb bytes, where Nb is the block length divided by 32 (4 for 128-bit key, 6 for 192-bit key, 8 for 256-bit key). Likewise the key length and number of rounds(iterations) differ from key to key as shown in table 1.

Table1 Different keys and its attributes

Algorithm	Key length (Nk words)	Block Size (Nb words)	Number of rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

For encryption and decryption there are 4 different steps

Sub Bytes: Every byte in the state is replaced by another one using Rinjdale S-box (Substitution Box).

Shift Rows: Every row in the state (4×4 array) is shifted left k bytes and the k depends on the key and the row number.

Mix Column: A linear mixing operation which operates on the columns of the state, combining the four bytes in each column.

Add Round key: Each byte of the state is combined with a round key, which is different key for each round and derived from the Rinjdale key schedule.

The sequence in which the operation is carried out is as follows:

Round 1:

A. Add Round key.

Following Rounds:

A. Sub Bytes.
B. Shift Rows.
C. Mix Column.
D. Add Round Key.

Final Round:

A. Sub Bytes.
B. Shift Row.
C. Add Round Key.

This is shown in fig1. The AES algorithm can be implemented in both hardware and software. The software implementation of AES algorithm is a slow process when compared with hardware process.

A.AES Encryption:

Encryption is the process of converting the plain text into a format which is not easily readable and is called as cipher. The cipher is got by doing a series of mathematical operations iteratively.

a) Sub Bytes:

In this sub bytes step the data in the plain text is substituted by some pre-defined values from a substitution box. The substitution box which is used commonly is rinjdale substitution box. The substitution box is invertible.

b) Shift Rows:

In shift rows operation the rows in the 4×4 matrix is shifted to left r bits and r varies with the rows of the matrix(r=0 for row1, r=1 for row2, r=2 for row3, r=3 for row 4). This process is illustrated in fig 2.

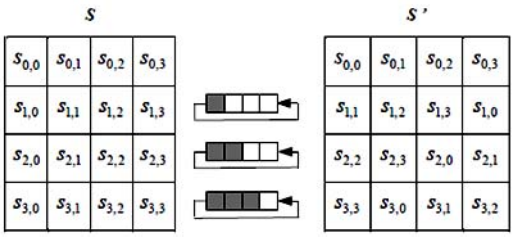

Figure 1. Shift Rows

This has the effect of moving positions of lower positions in the row, while the lowest bytes wrap around to the top of the row.

c) *Mix Columns:*

Mix column is calculated using the below formula.

$$\begin{pmatrix} Ro \\ R1 \\ R2 \\ R3 \end{pmatrix} = \begin{pmatrix} 2\ 3\ 1\ 1 \\ 1\ 2\ 3\ 1 \\ 1\ 1\ 2\ 3 \\ 3\ 1\ 1\ 2 \end{pmatrix} \begin{pmatrix} a0 \\ a1 \\ a2 \\ a3 \end{pmatrix}$$

Here a0, a1, a2, a3 is calculated using the polynomials as below

$$a(x) = \{2\}x^3 + \{3\}x^2 + \{1\}x + \{1\}.$$

The mix column transformation operates on the state column by column, treating each column as a four term polynomial. The columns are considered as polynomials over GF(28) and multiplied modulo x4 + 1 with a fixed polynomial a(x) which is got from the above formula. This can also written as a matrix multiplication

$$s'(x) = a(x) \otimes s(x)$$

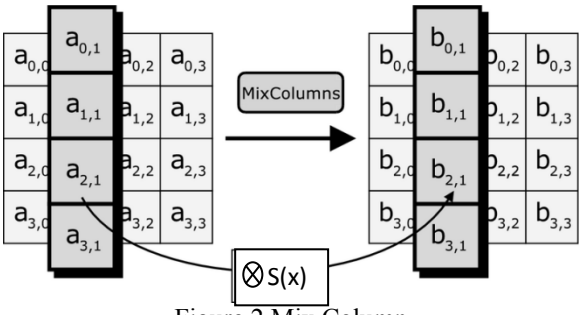

Figure 2 Mix Column

d) *Add Round Key:*

In the add round key step the 128 bit data is xored with the sub key of the current round using the key expansion operation. The add round key is used in two different places one during the start that is when round r=0 and then during the other rounds that is when 1 ≤ round≤ Nr, where Nr is the maximum number of rounds. The formula to perform the add round key is

$$S'(x) = S(x) \oplus R(x)$$

S'(x) – state after adding round key
S(x) – state before adding round key
R(x) – round key

e) *Key Expansion:*

The key expansion has three steps:

a) Byte Substitution *subword()*
b) Rotation *rotword()*
c) Xor with RCON (round constant)

The input to key schedule is the cipher key K. Key expansion generates a total of Nb(Nr + 1) words. The algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [wi], with i in the range $0 \le i < Nb(Nr + 1)$.

The subword()function takes a four byte input and applies the byte substitution operation and produces an output word. The rotword() takes a word [a0, a1, a2, a3] as input and performs a cyclic permutation to produce [a1, a2, a3, a0] as output word. The round constant word array rcon[i] is calculated using the below formula in rinjdale finite field.

rcon[i]=$x^{[i/4]-1}$ mod $x^8+x^4+x^3+x+1$

The first Nk words of the expanded key are filled with the cipher key. Every following word w[i] is equal to the xor of previous word w[i-1] and the word Nk positions earlier w[i-Nk]. For words in positions that are a multiple of Nk, a transformation is applied to w[i-1] prior to the XOR, followed by an XOR with a round constant Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word rotword() and byte substitution subword(). But in key expansion of 256-bit cipher if Nk=8 and i-4 is a multiple of Nk then subword() function is applied to w[i-1] prior to the xor. The algorithm for the key expansion routine is:

Table 2 Algorithm for key expansion

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
word temp
i = 0
while (i < Nk)
{
    w[i] = word(key[4*i], key[4*i+1]
    key[4*i+2]i+3])
    i = i+1
}
end while
i = Nk
while (i < Nb * (Nr+1)]
{
    temp = w[i-1]
    if (i mod Nk = 0)
    temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
    else if (Nk > 6 and i mod Nk = 4)
          temp = SubWord(temp)
    end if
```

```
        w[i] = w[i-Nk] xor temp
    i = i + 1
}
end while
end
```

Thus with all the above operations the algorithm for the encryption of the data is as follows. Since it begins and ends with the add round key operation there is no wasted un keyed step in the beginning or the end.

Table 3 Algorithm for encryption

```
byte state[4,Nb]
state = in

AddRoundKey(state, keySchedule[0, Nb-1])

for round = 1 step 1 to Nr–1
 {
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state)
        keySchedule[round*Nb, (round+1)*Nb-1])
}
SubBytes(state)
ShiftRows(state)
AddRoundKey(state,              keySchedule[Nr*Nb,
(Nr+1)*Nb-1])
out = state
```

B.AES Decryption:

The decryption of the data which was encrypted using the AES is done by inverting all the encryption operations with the same key with which it is encrypted since the AES is a symmetric encryption standard. In the decryption process the sequence of the transformations differs from that of the encryption but the key expansion for encryption and decryption are the same. However several properties of the AES algorithm allow for an equivalent decryption with the same sequence of transformations as that in encryption. The operations of the decryption are listed below

a) Inverse Sub Bytes.
b) Inverse Shift Rows.
c) Add Round Key.
d) Inverse mix columns.

Inverse Sub Bytes: This operation is same as it is in the encryption process but the only difference is the inverse of the substitution box is used here since the substitution box which we used in the encryption is invertible.

Inverse Shift Rows: The inverse shift rows operation inverses the shift row operation in the encryption process by right shifting the elements in the rows.

Add Round Key: The add round key process is the same as that of the one in the encryption process.

Inverse Mix Columns: In inverse mix column operation the same operation in the mix column is done but with the different matrix as specified below.

the algorithm for decryption process is:

Table 4 Algorithm for decryption

```
byte state[4,Nb]
state = in
AddRoundKey(state,              keySchedule[Nr*Nb,
(Nr+1)*Nb-1])
for round = Nr-1 step -1 downto 1
{
        InvShiftRows(state)
        InvSubBytes(state)
        AddRoundKey(state)
        keySchedule[round*Nb, (round+1)*Nb-1])
        InvMixColumns(state)
}
InvShiftRows(state)
InvSubBytes(state)
AddRoundKey(state, keySchedule[0, Nb-1])
out = state
```

III. Simulation results of software Implementation.

A mat lab program was developed using the above algorithms and the simulation results for different keys are tabulated below in table 5.

128-bit data and 128-bit key	
data =	key =
'32' '43' 'f6' 'a8' '88' '5a' '30' '8d' '31' '31' '98' 'a2' 'e0' '37' '07' '34'	'2b' '7e' '15' '16' '28' 'ae' 'd2' 'a6' 'ab' 'f7' '15' '88' '09' 'cf' '4f' '3c'
cipher =	dedat =
'00' '96' 'b0' 'db' '43' '9e' 'd6' '82' '06' '9e' 'f9' 'ac' 'bd' 'aa' 'ce' '3e'	'32' '43' 'f6' 'a8' '88' '5a' '30' '8d' '31' '31' '98' 'a2' 'e0' '37' '07' '34'
128-bit data and 192-bit key	
data =	key1 =
'32' '43' 'f6' 'a8' '88' '5a' '30' '8d'	'8e' 'da' 'c8' '80' '62' '52' '73' '0e' '10' '90' 'f8' '2c' 'b0' '64' 'f3' '79' 'ea' '6b' 'f7' '52' '2b' 'e5' 'd2' '7b'

'31' '31' '98' 'a2' 'e0' '37' '07' '34'	
cipher = 'd3' '5e' '9e' '35' '87' 'b0' '04' '8b' '76' '3f' '56' 'ff' 'd4' '7f' '49' '6e'	dedat = '32' '43' 'f6' 'a8' '88' '5a' '30' '8d' '31' '31' '98' 'a2' 'e0' '37' '07' '34'

Here the simulation result shows the data, key used, cipher (encrypted data) and decrypted data. The next step is to develop a hardware architecture using VERILOG and FPGA.

IV. Hardware Implementation of Rijndael encryption Algorithm

Many hardware implementation of Rijndael encryption algorithm using VHDL is available. In most of the case hardware implemetataion of AES uses only the AES-128 candidate. some software implementation of AES192 and AES -256 are available. In the proposed architecture all candidates of AES i.e. AES-128, AES192 and AES-256 are implemented in the same device. The proposed design is implemented using basic key components of AES encryption algorithms. Iterative looping techniques followed to implement the entire design modules of AES encryption algorithm. The key controller unit, key expansion unit, and round function unit and mix column unit every thing are implemented in hardware.

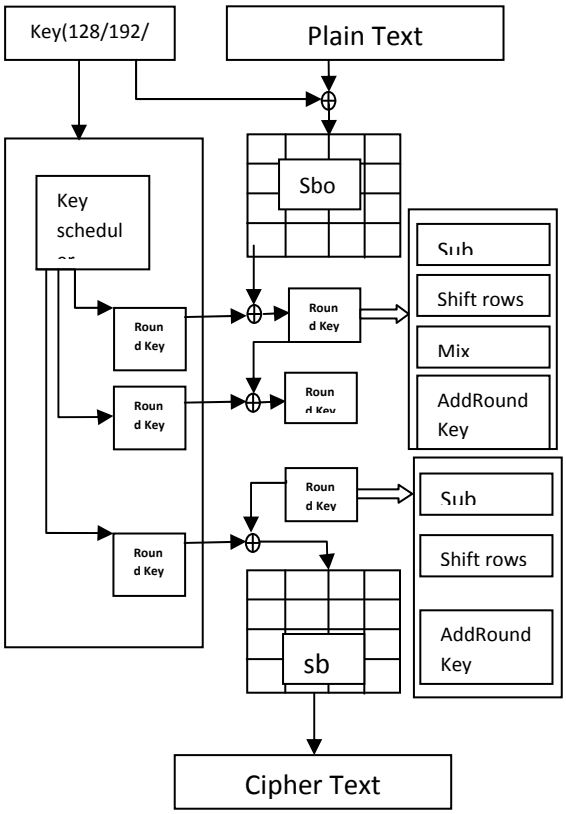

Figure 3. Flow Chart Of AES Encryption

Testing and Verification

To ensure the proposed design gives a better results in terms of area and throughput the design is implemented Xilinx 9.2 and FPGA device XC2V6000BF957-6 used for downloading. In table 1. The device utilization summary of complete algorithm i.e. AES-128, AES-192, AES-256 in same hardware is shown.

Table 1. device utilization summary of AES Encryption

FPGA Device	XC2V6000BF957-6	
Allocated Area	Used/Available	% utilization
CLB Slices	2943/33972	8
No. of LUT	5802/67584	8
IOs	836	
IOBs	384/684	56

The proposed design is compared with other different design implementation in Xilinx FPGA device. The proposed device is implemented in XC2V6000BF957-6 to have sufficient memory to implement all the three different rijndael algorithm. The maximum core frequency is 62.5 MHz Each round are completed in one clock cycle and one clock cycle for registering the input, so total clock cycle need for processing 128-bit data is 12 clock for AES-128. As a result 666.7 Mbps of throughput is achieved. Throughput calculated by other researchers is listed in table2.

Design	Device	Area (CLBs)	Throughput Mbits/Sec
Gaj	XCV1000BG560-6	2902	331.5
Dandalis	XCV 1000	5673	353.0
Proposed design	XC2V600BF957-6	2943/33792	666.7

The table 2. Show s that proposed design outperform all designs based on iterative looping in terms of area and throughput. The performance of AES-192 and AES 256 is also verified and simulation results are given. The novel architecture to implement all aes candidates in same hardware is proposed.

Conclusion:

Thus the Encryption unit for a mobile terminal is designed which encloses an encryption and decryption unit using AES algorithm which works with 128-bit data and three different keys 128, 192, 256 bits. Thus it can reduce the space by enclosing three different encryption standards in a single architecture and the power consumption can also be reduced which makes it usable in battery operated network devices having Bluetooth and wireless communication devices like software radio.

REFERENCES

[1] "FPGA implementation and performance evaluation of AES-CCM cores for wireless networks" Ignacio Algredo-Badillo, Claudia Feregrino-Uribe, Ren´e Cumplido, Miguel Morales-Sandoval Department of Computer Science,

[2] "An Efficient Software Implementation of AES-CCM for IEEE 802.11i Wireless Standard" Abdul Samiah, Arshad Aziz and Nassar Ikram-2007

[3] "An Energy-Eficient AES-CCM Implementation for IEEE802.15.4 Wireless Sensor Networks" Lian Huai, Xuecheng Zou, Zhenglin Liu, Yu Han-2009 international conference.

[4] "Efficient Sequential Architecture for the AES CCMMode in the 802.16e Standard" Jae Deok Ji Seok Won Jung-2009 international conference.

[5] "Announcing the ADVANCED ENCRYPTION STANDARD (AES)" Federal Information Processing Standards Publication 197 November 26, 2001

[6] "An Efficient FPGA Implementation of Advanced Encryption Standard Algorithm"Shuenn-Shyang Wang and Wan-Sheng N

[7] "AES 128 Encryption/Decryption" David Leifker & Gentre Graham Bradley University Department of Electrical Engineering Advisor: Dr. Vinod Prasad

[8] K. Gaj, P. Chodowiec: Comparison of the Hardware Performance of the AES Candidatesusing Reconfigurable Hardware: The ThirdAdvanced Encryption Standard CandidateConference, April 13-14, 2000, New York,USA.

[9] A. Dandalis, V. K. Prasanna, J. D. P. Rolim: A Comparative Study of Performance of AESCandidates Using FPGAs: The Third Advanced Encryption Standard Candidate Conference,April 13-14, 2000, New York, USA.

[10] Dennis Ka Yau Tong, Pui Sze Lo, Kin HongLee, Philip H.W. Leong, "A System LevelImplementation of Rijndael on a Memory-slotbased FPGA Card", Proceedings of the 2002IEEE International Conference on FieldProgrammable Technology (FPT), Hong Kong,pp. 102-109, 2002

[11] A.J. Elbert, E. Yip, B. Chetwynd, C. Paar: An FPGA Implementation and Performance Evaluation of the AES Block Cipher CandidateAlgorithm Finalists, IEEE Transactions on VLSI,August 2001, vol. 9, no. 4, pp. 545-557.

[12] M. McLoone, J.V McCanny: High Performance Single-Chip FPGA Rijndael Algorithm Implementations, CHES 2001, pp. 65-76.

[13] "Announcing the Advanced Encryption Standard(AES)" Federal Information Processing Standards Publication 197, November 26, 2001.

AUTHORS PROFILE

L. Thulasimani has obtained her BE and ME degree from Coimbatore Institute of Technology, India in 1998 and 2001 respectively. She has started her teaching profession in the year 2001 in PSNA Engineering College, Dindigul. At present she is an Lecturer in department of Electronic and Communication Engineering in PSG college of Technology, Coimbatore .She has published 4 research papers in International and National conferences. She is a part time Ph.D research scalar in Anna University Chennai. Her areas of interest are Wireless security, Networking and signal processing. She is a life member of ISTE and IEEE.

Dr. M. Madheswaran has obtained his Ph.D. degree in Electronics Engineering from Institute of Technology, Banaras Hindu University, Varanasi in 1999 and M.E degree in Microwave Engineering from Birla Institute of Technology, Ranchi, India. He has started his teaching profession in the year 1991 to serve his parent Institution Mohd. Sathak Engineering College, Kilakarai where he obtained his Bachelor Degree in ECE. He has served KSR college of Technology from 1999 to 2001 and PSNA College of Engineering and Technology, Dindigul from 2001 to 2006. He has been awarded Young Scientist Fellowship by the Tamil Nadu State Council for Science and Technology and Senior Research Fellowship by Council for Scientific and Industrial Research, New Delhi in the year 1994 and 1996 respectively. He has published 120 research papers in International and National Journals as well as conferences. His field of interest includes semiconductor devices, microwave electronics, optoelectronics and signal processing. He is a Senior member of IEEE, Fellow of IETE, and IE and member of ISTE

In [13], there are several test vectors for all candidates of AES. Different test vectors are used to test the proposed design. The test vectors for all the case are given below: Plaintext: 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Key: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Cipher text: 69 c4 e0 d8 6a 7b 04 30 d8 cd b7 80 70 b4 c5 5a
In Figures 4 gives the schematic proposed design and 5,6 and 7, the simulation outputs of the abovementioned test vectors of the encryption algorithms are given. The following figures show schematic diagram and simulation outputs of the AES encryption .

Appendix and Simulation Results

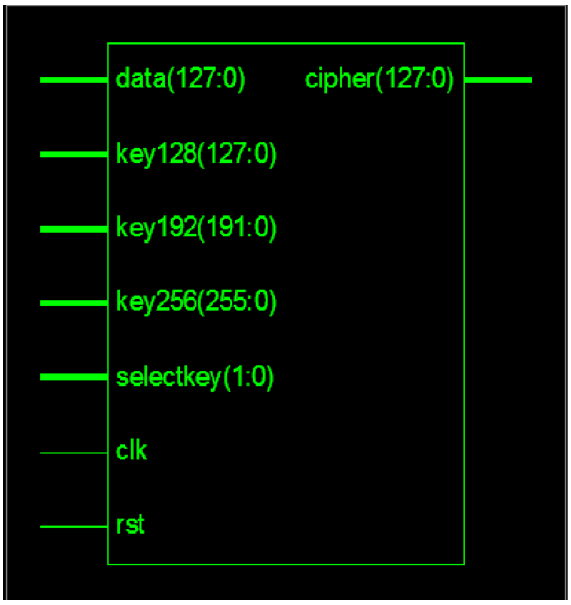

Figure 4 schematic Diagram AES-128/192/256 Architecture

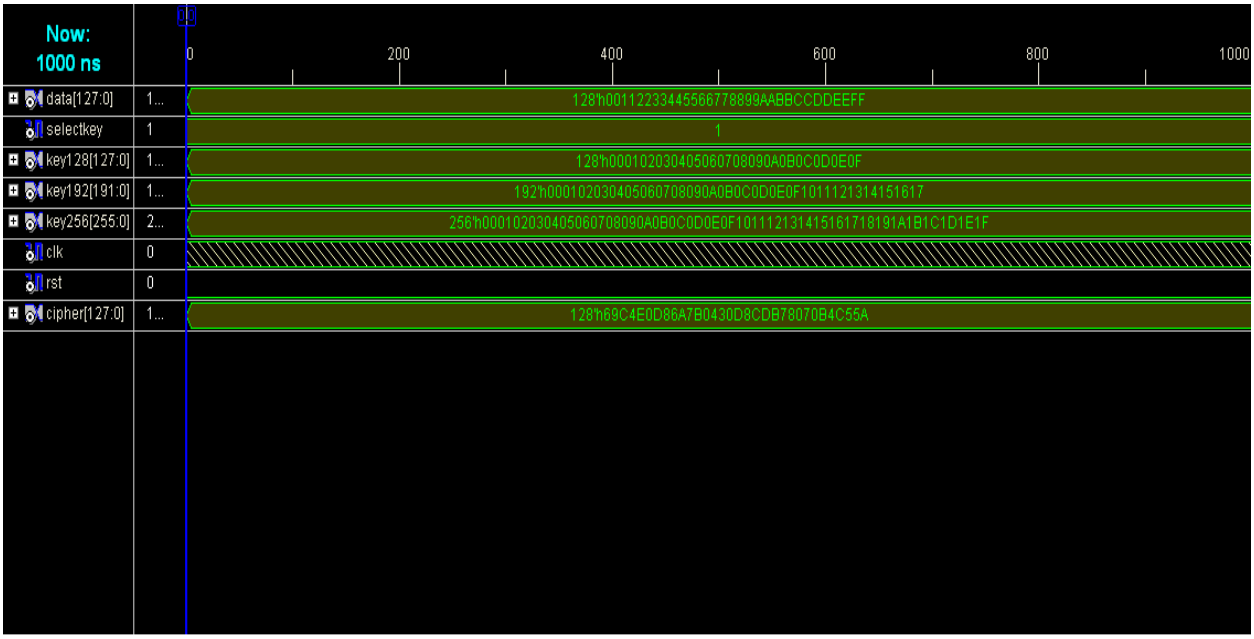

Figure 5 Simulation output of AES-128 Encryption

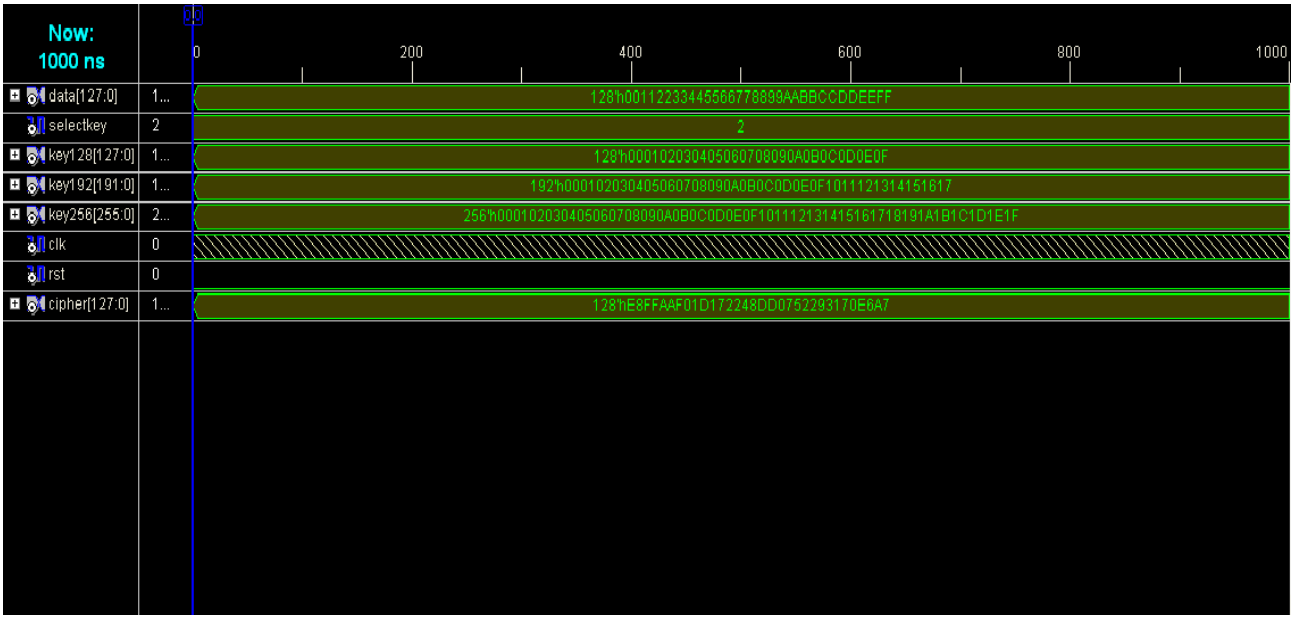

Figure 6. Simulation output of AES-192 Encryption

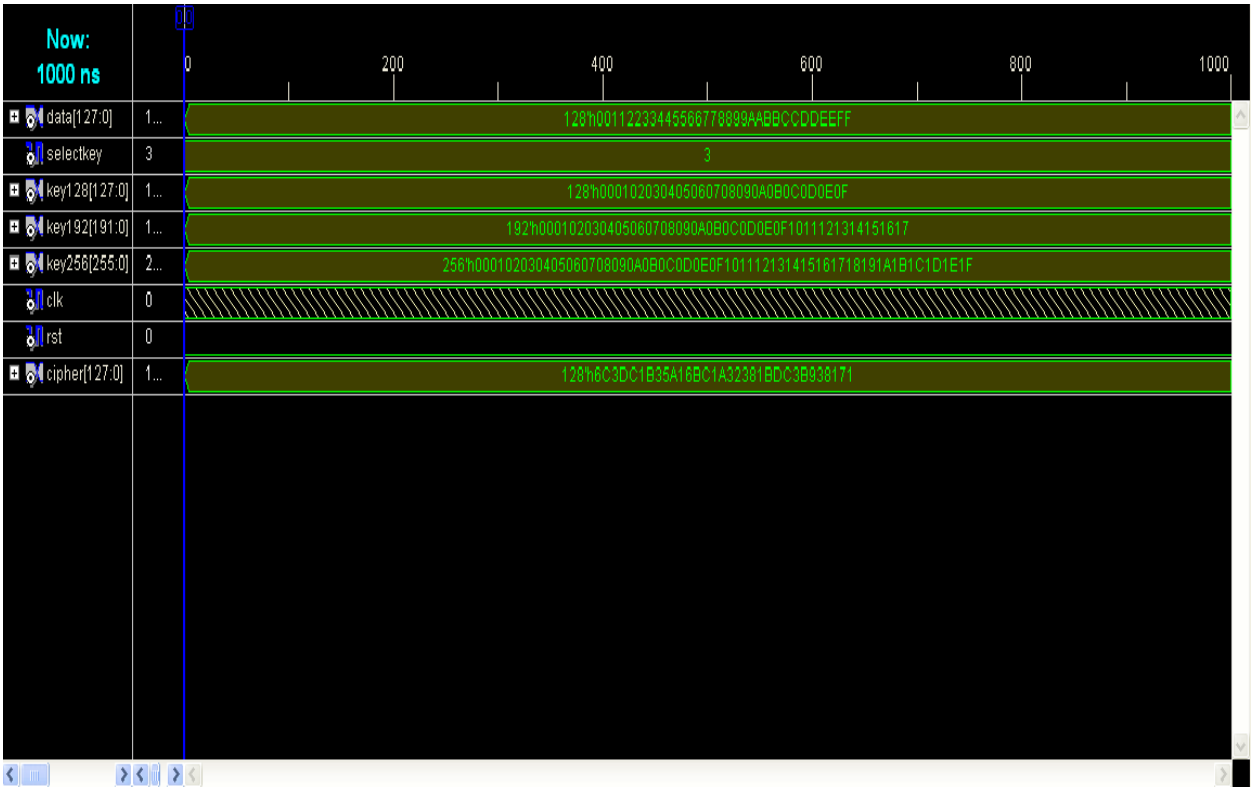

Figure 7.Simulation output of AES-256 Encryption