FPGA Implementation of Enhanced Key Expansion Algorithm for Advanced Encryption Standard

D.Rahul Gandh
Student, M.Tech(WC)
Dept.of Electronics & Communication Engg.
Pondicherry Engineering College,
Pondicherry, INDIA.

V.Kamalakannan
Research Scholar
Dept.of Electronics & Communication Engg.
Pondicherry Engineering College,
Pondicherry, INDIA

R.Balamurugan
Student, M.Tech(WC)
Dept.of Electronics & Communication Engg.
Pondicherry Engineering College,
Pondicherry, INDIA.

S.Tamilselvan
Assistant Professor
Dept.of Electronics & Communication Engg.
Pondicherry Engineering College,
Pondicherry, INDIA.

Abstract—**Cryptography is a technique related to aspects of information security such as data confidentiality, data integrity and entity authentication. In data and telecommunication systems, Security is the most important part for an effective communication, where to increase the security as well as complexity, more randomization in secret keys is necessary to enhance the cryptography algorithms. In traditional AES, Even though the round keys have high security, Power analysis attack and Saturation attack are effective to the key expansion algorithm of AES due to the deducible key rounds and it leads to security problems. As a result, a new algorithm for generation of round keys is developed for AES. On hardware platform, these algorithms are realizing with enormous memory spaces and large execution time. An alternative hardware platform scenario is provided by Field programmable gate arrays (FPGAs) due to its reconfiguration nature, marketing speed and low price. Accordingly, a hardware implementation of the AES-128 encryption and decryption algorithm with the new algorithm of round key expansion is proposed to improve the security against such attacks. This structure will experimentally simulate using Xilinx software with Verilog HDL and hardware implementation on FPGA.**

Keywords— Advanced Encryption Standard; FPGA; Key expansion; Sub word; Rot word;Rconst.

I. INTRODUCTION

Cryptography plays an important role in securing the information, which enables to store sensitive information or transmit it across insecure networks so that unauthorized persons cannot obtain it. The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001. AES is a cryptographic algorithm that is used to secure electronic data and it has a symmetric block cipher with a block length of 128 bits that can encipher (encrypt) and decipher (decrypt) the data. The AES algorithm consists of three main parts:

1. Cipher (Encryption),
2. Inverse Cipher (Decryption)
3. Key Expansion.

Encryption converts data to an indiscernible form called as cipher-text. Decryption converts the cipher text back into its indigenous form, that is original data.

The AES algorithm is capable of using different cryptographic key lengths of 128, 192, and 256 bits to encipher and decipher the data. Key expansion is used for generating the keys for 10 rounds that are produced from the original input key. Every round keys are different from one another to improve the security of the algorithm. Thus it is essential to improve the performance of the key expansion from the external attacks. But the traditional key expansion of AES has some security problem due to the key arrangements are depending upon the previous key rounds. As a result, a new algorithm of key expansion is proposed to enhance the security of the Advanced Encryption Standard for 128-bit key size.

II. ADVANCED ENCRYPTION STANDARD ALGORITHM

A. *AES Specification*

The length of the input block, the output block and the State is 128 bits for the AES algorithm which is represented by $N_b = 4$. The input 128-bits are arranged in 4 × 4 matrix into 16 bytes that reflects the number of 32-bit words (number of columns) in the State. The AES algorithm will support at least one of the three key lengths: 128, 192, or 256-bits (i.e., $N_k = 4$, 6, or 8, respectively). The length of key is represented by $N_k =$ 4, 6, or 8 which reflects the number of 32-bit words (number of columns) in the Cipher Key. The number of rounds for AES algorithm to be performed during the execution is dependent on the key size. The number of rounds is represented by N_r, where $N_r = 10$ when $N_k = 4$, $N_r = 12$ when $N_k = 6$, and $N_r = 14$ when $N_k = 8$.

AES algorithm uses a round function for both its Cipher and Inverse Cipher that is composed of four different byte-oriented transformations:

- Byte substitution using a S-box lookup table
- Row-wise permutation of the State array by different offsets
- Column-wise mixing within each column of the State array
- Addition of round key to the State

The structure of AES algorithm for both the encryption and decryption is shown in the Fig. 1.which shows the overall process.

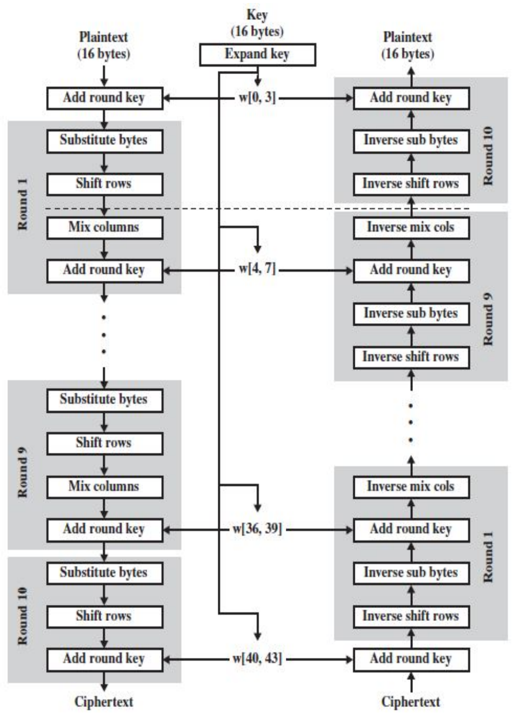


Fig. 1. Structure of AES encryption and decryption.

B. *Encryption process*

The initial process in AES encryption is the addition (xoring) of original key to the input data, which is called an initial round. This is followed by nine iterations of a normal round and ends with a modified final round. During each normal round the following operations are performed in the following order: Byte substitution, Row-wise permutation, Column-wise mixing, Addition of the round key. The final round is also a normal round without the Column-wise mixing process.

1) Byte substitution: This is a byte-by-byte substitution process shown in Fig. 2. which processes sustitution of bytes. The substitution byte for each input byte is found by using the S-Box lookup table. The size of the lookup table is 16×16. To find the substitute byte for a given input byte, we divide the input byte into two 4-bit patterns, each yielding an integer value between 0 and 15 which can represent these by hex values 0 through F. One of the hex values is used as a row index and the other as a column index for reaching into the 16×16 lookup table.

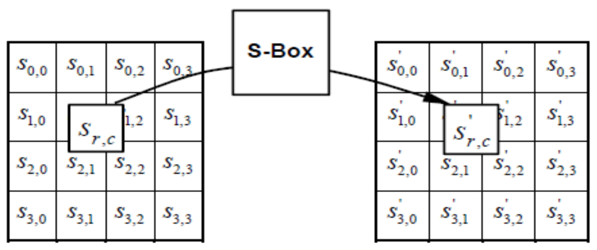


Fig. 2. Byte Substitution.

The entries in the lookup table are constructed by a combination of GF(2^8) arithmetic and bit scrambling. The goal of the substitution step is to reduce the correlation between the input bits and the output bits. The bit scrambling part of the substitution step ensures that the substitution can not be described in the form of evaluating a simple mathematical function.

2) Row-wise Permutation: The Row-wise permutation transformation consists of (i) not shifting the first row of the state array at all (ii) circularly shifting the second row by one byte to the left (iii) circularly shifting the third row by two bytes to the left and (iv) circularly shifting the last row by three bytes to the left.This operation on the state array can be represented by

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \Longrightarrow \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{bmatrix}$$

3) Column-wise mixing: This step replaces each byte of a column by a function of all the bytes in the same column.

For the bytes in the first row of the state, operation can be stated as

$$S'_{0,c} = (\{02\} \bullet S_{0,c}) + (\{03\} \bullet S_{1,c}) + S_{2,c} + S_{3,c} \quad (1)$$

For the bytes in the second row of the state can be stated as

$$S'_{1,c} = S_{0,c} + (\{02\} \bullet S_{1,c}) + (\{03\} \bullet S_{2,c}) + S_{3,c} \quad (2)$$

For the bytes in the third row of the state can be stated as

$$S'_{2,c} = S_{0,c} + S_{1,c} + (\{02\} \bullet S_{2,c}) + (\{03\} \bullet S_{3,c}) \quad (3)$$

And for the bytes in the fourth row of the state array can be stated as

$$S'_{3,c} = (\{03\} \bullet S_{0,c}) + S_{1,c} + S_{2,c} + (\{02\} \bullet S_{3,c}) \quad (4)$$

More compactly, the column operations can be shown as

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

Where, a row of the leftmost matrix multiples a column of the state array matrix, additions involved are meant to be XOR operation.

4) Addition of round key: The key words are generated by key expansion process. Round Key is added to every State by a XOR operation where each Round Key consists of N_b words.

Those N_b words are each added into the columns of the State, such that *[w_i]* are the key schedule words and round is a value in the range 0 round N_r. In the Cipher, the initial Round Key addition occurs when round = 0, prior to the first application of the round function. Add Round Key transformation to the N_r rounds of the Cipher occurs when 1<round <N_r. A simple xor operation with the state to key word is shown in Fig. 3.

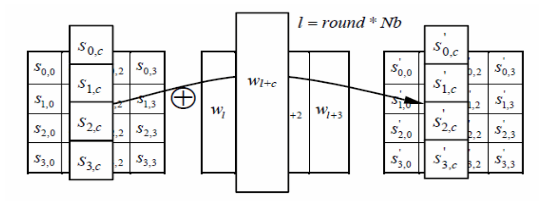


Fig. 3. Addition of Round Key XORs each column of the State with a word.

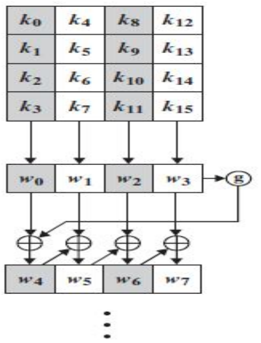
C. *Traditional AES Key Expansion*

The AES key expansion algorithm takes input 128-bits as 16-bytes which is represented by k_0, k_1,…k_{15}. The first 4-key words w_0, w_1, w_2, w_3 are obtained from those columns which is shown in Fig. 4. This original key words are sufficient to generate add round keys for each of the 10 rounds of the cipher.

In the traditional AES key expansion algorithm, supposing that the original key words are (w_0, w_1, w_2, w_3), it generates the sub-key words (w_4,w_5 ..., w_{43}) for 10 rounds.(i.e)
Original key: w_0, w_1, w_2, w_3
1st round key: w_4, w_5, w_6, w_7
2nd round key: w_8, w_9, w_{10}, w_{11}
…
10th round key: w_{40}, w_{41}, w_{42}, w_{43}

The key expansion process of the first round is

- $w_4 = w_0 + Subword(Rotword(w_3))+Rcon(1)$
- $w_5 = w_1 + w_4$
- $w_6 = w_2 + w_5$
- $w_7 = w_3 + w_6$



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7

Fig. 4. Key Expansion process.

From the second round, the key expansion process will be same upto 10th round as follows.

- $w_8 = w_4 + Subword(Rotword(w_7))+Rcon(2)$
- $w_9 = w_5 + w_8$
- $w_{10} = w_6 + w_9$
- $w_{11} = w_7 + w_{10}$

RotWord performs a one-byte circular left shift on a word. This means that an input word [B_0, B_1, B_2, B_3] is transformed into [B_1, B_2, B_3, B_0] . SubWord performs a byte substitution on each byte of its input word, using the S-box. The result is XORed with a round constant. The Fig. 5. shown the process of function g.

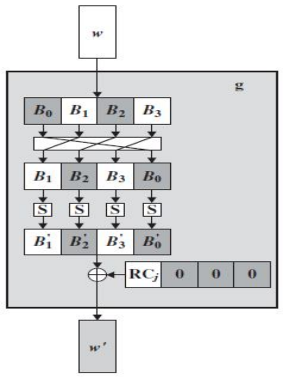


Fig. 5.Function g block.

The first four words of the expanded key are defined from the original keys. From the second round each added word *w[i]* depends on the immediately preceding word *w[i-1]* and the word four positions back *w[i-4]* from the previous round. In three out of four cases, a simple XOR is used. A complex function g is used in generation of the expanded key to make higher security.

D. *Decryption process*

In AES decryption, the operations are in reverse order. It starts with an initial round followed by nine iterations of an inverse round and ends with an Addition of Round Key. An inverse round consists of the following operations in this order: Addition of Round Key, Inverse column-wise mixing, Inverse Row-wise permutation and Inverse Substitution of Bytes. An initial round is an inverse round without the Inverse Column-wise mixing and addition of round key has its own inverse for both the encryption and decryption process.

1) Inverse Row-wise permutation: For decryption, the corresponding step shifts the rows in opposite manner. The first row is left unchanged, the second row is shifted to the right by one byte, the third row to the right by two bytes, and the last row to the right by three bytes, all are circularly shifted.

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \quad ====> \quad \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,3} & S_{1,0} & S_{1,1} & S_{1,2} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,1} & S_{3,2} & S_{3,3} & S_{3,0} \end{bmatrix}$$

2) Inverse column-wise mixing: It is the inverse process of column-wise mixing that operates on the State column-by-column and taking each column as a four term polynomial. The columns are considered as polynomials over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial a-1(x), given by a-1(x) = {0b} x^3 + {0d} x^2 + {09} x + {0e}, this can be written as a matrix multiplication.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

3) Inverse substitution of bytes: The inverse substitution byte for each input byte is found by using the inverse s-box lookup table. The size of the lookup table is 16×16. To find the inverse substitution byte for a given input byte, we divide the input byte into two 4-bit patterns, each yielding an integer value between 0 and 15 which can represent these by hex values 0 through F. One of the hex values is used as a row index and the other as a column index for reaching into the 16×16 lookup table. The entries in the lookup table are constructed by a multiplicative inverse of GF(2^8).

III. Proposed Key Expansion Algorithm

It is essential to enhance the security of a cryptographic algorithm in data communication. In Advanced Encryption Standard, the traditional key expansion algorithm has some security problems due to the deducible key arrangements. It is known the 4-words of round key in each round can be obtained from the 4-words of former round key. On the contrary, the 4-words of former round key can also be deduced by the latter ones. So, if one round key is known, the attacker can deduce the sub-key of every round and even the seed key. Meanwhile, Power analysis attack and Saturation attack which are effective to AES, both make use of the simple key word arrangement of key expansion algorithm. As a result, the Advanced Encryption Standard with a new algorithm of enhanced key expansion arrangement is proposed.

In order not to make the key generating algorithm more complicated, it is necessary to complicate the operations. The generating algorithm is as follows

- $w_i = w_{i-8} + Subword(Rotword(w_{i-4})) + Rcon(i/4)$ where $i = 8, 12 \ldots 40$
- $w_i = w_{i-8} + w_{i-4}$; $(8 < i < 44$ *and i is not a multiple of* $4)$

For the sub-key words of the 1st round, it can be only generated from the original key. It has "one way" character so that derivation can only be done from former to later.

- $w_4 = w_0 + w_2$
- $w_5 = w_1 + w_3$
- $w_6 = w_4 + w_5$
- $w_7 = w_5$ + Subword(Rotword(w_6))+Rcon(1)

From the second round, the key expansion process will be same as follows upto 10th round.

- $w_8 = w_0 + w_4$
- $w_9 = w_1 + w_5$
- $w_{10} = w_2 + w_6$
- $w_{11} = w_3$ + SubWord(Rotword(w_7))+Rcon(2)

Supposing attackers have known the key (w_4, w_5, w_6, w_7), it is hard to deduce the original key (w_0, w_1, w_2, w_3) , because w_7 only depends on w_5 and w_6, w_6 only depends on w_4 and w_5,while w_5 depends only on w_1 and w_3. Even if w_5 known, 2^{32} exhaustive attacks need to get w_1 and w_3 (each character length is 32bit). For the same reason, to get w_0 and w_2 from w_4, 2^{32} times attacks are needed. Thus, to get the first round sub-key, still they need 2^{64} times to guess the original key.

According to the analysis above, the attacker needs to crack two successive rounds of sub-key words to get the whole key bits. So the proposed algorithm for key expansion has higher key security compared to the traditional AES key expansion algorithm, but the complexity remains the same.

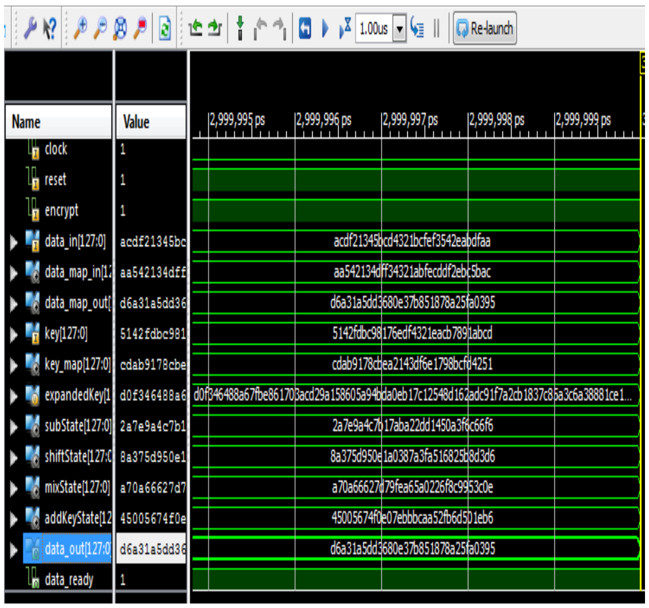
IV. Simulation Result of AES Encryption and Decryption using Proposed Key Expansion Algorithm

In the test case, the AES encryption and decryption for a 16-byte data is simulated using the Xilinx software tool. The encryption process of the AES algorithm generates a cipher text for a given data.

A. Encryption

In the encryption process the AES algorithm generates a cipher text for a given 16-byte data and cipher key. The simulation result is shown in Fig. 6.

Input	= [acdf21345bcd4321bcfef3542eabdfaa]
Key	= [5142fdbc98176edf4321eacb7891abcd]
Cipher text	= [d6a31a5dd3680e37b851878a25fa0395]



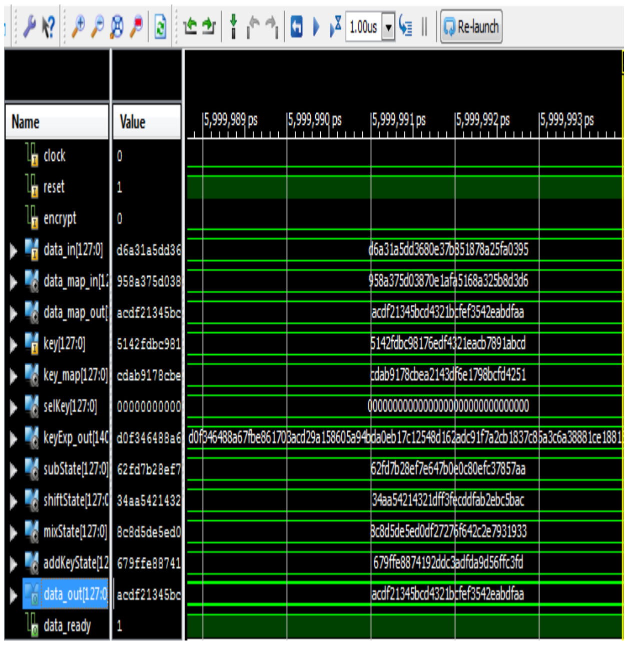
```
Name              Value             2,999,995 ps  2,999,996 ps  2,999,997 ps  2,999,998 ps  2,999,999 ps
clock             1
reset             1
encrypt           1
data_in[127:0]    acdf21345bc       acdf21345bcd4321bcfef3542eabdfaa
data_map_in[1:    aa542134dff       aa542134dff34321abfecddf2ebc5bac
data_map_out[     d6a31a5dd36       d6a31a5dd3680e37b851878a25fa0395
key[127:0]        5142fdbc981       5142fdbc98176edf4321eacb7891abcd
key_map[127:0]    cdab9178cbe       cdab9178cbea2143df6e1798bcfd4251
expandedKey[1     d0f346488a6       d0f346488a67fbe861703acd29a158605a94bda0eb17c12548d162adc91f7a2cb1837c85a3c6a38881ce1...
subState[127:0]   2a7e9a4c7b1       2a7e9a4c7b17aba22dd1450a3f6c66f6
shiftState[127:0  8a375d950e1       8a375d950e1a0387a3fa516825b8d3d6
mixState[127:0]   a70a66627d7       a70a66627d79fea65a0226f8c9953c0e
addKeyState[12    45005674f0e       45005674f0e07ebbbcaa52fb6d501eb6
data_out[127:0]   d6a31a5dd36       d6a31a5dd3680e37b851878a25fa0395
data_ready        1
```

Fig. 6. Encryption Simulation Result

B. *Decryption*

The cipher text obtained in the encryption process is given as the input of decyption process and the original input data is obtained by decryption.The simulation result shown in Fig. 7.

input text	= [d6a31a5dd3680e37b851878a25fa0395]
Key	= [5142fdbc98176edf4321eacb7891abcd]
Original data	= [acdf21345bcd4321bcfef3542eabdfaa]



```
Name              Value
clock             0
reset             1
encrypt           0
data_in[127:0]    d6a31a5dd36     d6a31a5dd3680e37b851878a25fa0395
data_map_in[12    958a375d038     958a375d03870e1afa5168a325b8d3d6
data_map_out[     acdf21345bc     acdf21345bcd4321bcfef3542eabdfaa
key[127:0]        5142fdbc981     5142fdbc98176edf4321eacb7891abcd
key_map[127:0]    cdab9178cbe     cdab9178cbea2143df6e1798bcfd4251
selKey[127:0]     00000000000     00000000000000000000000000000000
keyExp_out[140    d0f346488a6     d0f346488a67fbe861703acd29a158605a94bda0eb17c12548d162adc91f7a2cb1837c85a3c6a38881ce1881
subState[127:0]   62fd7b28ef7     62fd7b28ef7e647b0e0c80efc37857aa
shiftState[127:0  34aa5421432     34aa54214321dff3fecddfab2ebc5bac
mixState[127:0]   8c8d5de5ed0     8c8d5de5ed0df27276f642c2e7931933
addKeyState[12    679ffe88741     679ffe8874192ddc3adfda9d56ffc3fd
data_out[127:0    acdf21345bc     acdf21345bcd4321bcfef3542eabdfaa
data_ready        1
```

Fig. 7. Decryption Simulation Result

Conclusion

The aim of this proposed algorithm is to increase the level of security for Advanced Encryption Standard and providing a faster processing time. In addition, the difficulty of attacking the key is increased which powers the ability to resist major attacks due to the high randomization of key expansion in the new algorithm. The algorithm was synthesized using XILINX 14.3 and implemented on FPGA. The simulation result for AES-128 is obtained by simulating the proposed key expansion algorithm using Verilog Hardware Description Language.

References

[1] AI-Wen Luo, Qing-Ming Yi, Min Shi. "Design and Implementation of Area-optimized AES on FPGA" ,IEEE Inter.conf.chal sci com engin.,978-1-61284- 109-0/2011.

[2] H. Mestiri, N. Benhadjyoussef, M. Machhout and R. Tourki, "A Comparative Study of Power Consumption Models for CPA Attack,"

[3] International Journal of Computer Network and Information Security,Vol. 5, No. 3, pp. 25-31, 2013.

[4] A. Moh'd, Y. Jararweh and L. Tawalbeh, "AES-512: 512-bit Advanced Encryption Standard algorithm design and evaluation," 7th International Conference on Information Assurance and Security (IAS 2011), pp. 292-297, 2011.

[5] M. Mozaffari-Kermani, and A. Reyhani-Masoleh, "Concurrent structureindependent fault detection schemes for the advanced encryption standard," IEEE Transactions on Computers, Vol. 59, pp. 608-622, 2010.

[6] H. Mestiri, N. Benhadjyoussef, M. Machhout and R. Tourki,
"A Comparative Study of Power Consumption Models for CPA Attack," International Journal of Computer Network and Information Security, Vol. 5, No. 3, pp. 25-31, 2013.

[7] J.Yang, J.Ding, N.Li and Y.X.Guo,"FPGA-based design and implementation of reduced AES algorithm" IEEE Inter.Conf. Chal Envir Sci Com Engin(CESCE).,Vol.02, Issue.5-6, Jun 2010, pp.67-70.

[8] A.M. Deshpande, M.S. Deshpande and D.N. Kayatanavar, "FPGA Implementation of AES Encryption and Decryption"IEEE Inter.Conf.Cont,Auto,Com,and Ener., vol.01,issue 04, Jun.2009, pp.1-6.