IJERST

International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 4, No. 4
November 2015

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 4, No. 4, November 2015
© 2015 IJERST. All Rights Reserved

Research Paper

IMPLEMENTATION OF LOW LATENCY AES BASED ON OBFUSCATING TECHNIQUE

I N Bindhu Madhavi[1]* and S A J Mani Kumar G[2]

*Corresponding Author: I N Bindhu Madhavi ✉ bindu.immedisetty@gmail.com

High-level transformations have been known for a long time and have been used in a wide range of applications, such as pipelining, interleaving, folding and unfolding and have been used in synthesis of DSP systems. These techniques can be applied at the algorithm or the architecture level to achieve a tradeoff among different metrics of performance, such as area, speed, and power. In fact, high-level transformations naturally provide a means to obfuscate DSP circuits both structurally and functionally. Structural obfuscation is achieved by structural modification, which is realized by altering the structure of a DSP circuit by using high-level transformations. This is a so-called passive technique, which does not directly affect the functionality of the DSP circuit. Functional obfuscation is achieved by functional modification, which is realized by encrypting the normal functionality of a DSP circuit with a key. The DSP circuit cannot function correctly without the key. This is an active technique, which directly alters the functionality. High-level transformations alter the structure of a DSP circuit, while maintaining the original functionality. For instance, different folding sets lead to a family of folded architectures this can be exploited for structural obfuscation. As a result, circuits with the same functionality may have very different structures. Furthermore, high-level transformations may lead to architectures whose functionalities are not obvious. Take an extreme case for example, many filters can be folded into one multiply-accumulator (MAC), but their functionalities are not the same. In other words, one MAC with proper switches can implement many digital filters. In this project a fft with two different radixes are implemented and comparison is carried out in terms of area, power and delay by maintaining the tradeoff.

Keywords: AES, Symmetric, Obfuscation

INTRODUCTION

DSP, or Digital Signal Processing, as the term suggests, is the processing of signals by digital means. A signal in this context can mean a number of different things. Historically the origins of signal processing are in electrical engineering, and a signal here means an electrical signal carried by a wire or telephone line, or perhaps by

[1] M.Tech Student, Department of E.C.E., Chirala Engineering College, Ramapuram Beach Rd, Chirala, Andhra Pradesh 523157, India.

[2] Associate Professor, Department of E.C.E., Chirala Engineering College, Ramapuram Beach Rd, Chirala, Andhra Pradesh 523157, India.

a radio wave. More generally, however, a signal is a stream of information representing anything from stock prices to data from a remote-sensing satellite.

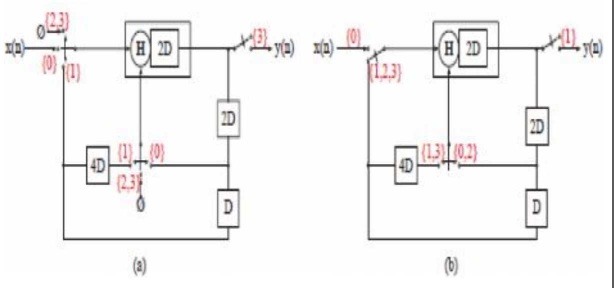
HIDING FUNCTIONALITY BY HIGH-LEVEL TRANSFORMATIONS

High-level transformations have been known for a long time and have been used in a wide range of applications, such as pipelining, interleaving, folding and unfolding, and have been used in synthesis of DSP systems. These techniques can be applied at the algorithm or the architecture level to achieve a tradeoff among different metrics of performance, such as area, speed, and power. However, the use of high-level transformations from a security perspective has not been studied before. In fact, high-level transformations naturally provide a means to obfuscate DSP circuits both structurally and functionally. Structural obfuscation and functional obfuscation are defined as follows:

- Structural obfuscation: achieved by structural modifica-tion, which is realized by altering the structure of a DSP circuit by using high-level transformations. This is a so-called “passive” technique, which does not directly affect the functionality of the DSP circuit.
- Functional obfuscation: achieved by functional modification, which is realized by encrypting the normal functionality of a DSP circuit with a key. The DSP circuit cannot function correctly without the key. This is an “active” technique, which directly alters the functionality.

High-level transformations alter the structure of a DSP circuit, while maintaining the original functionality. For instance, different folding sets lead to a family of folded architectures; this can be exploited for structural obfuscation. As a result, circuits with the same functionality may have very different structures. Furthermore, high-level transformations may lead to architectures whose functionalities are not obvious. Take an extreme case for example, many filters can be folded into one multiply-accumulator (MAC), but their functionalities are not the same. In other words, one MAC with proper switches can implement many digital filters. It is important to note this kind of structural obfuscation can be applied beyond the architecture level. For example, at the HDL level or the gate-level net list, high-level transformations can also lead to an obfuscated version of a DSP circuit. Therefore, circuits with different functionalities could have a similar structure by employing high-level transformations. Comparing the folded structures (a) and (b) in Figure 1, it can be observed that the two structures are exactly the same, except the switch instances. However, their functionalities are different, i.e., the former implements a 1st-order IIR filter, while the latter a 2nd-order IIR filter. In conclusion, if the switch instances are invisible to the adversary, the DSP systems will be hard to reverse engineer. The adversary who only has knowledge of the structural information but lacks knowledge of the switch instances cannot easily discover the functionality of a DSP circuit.

Figure 1: (a) Structural Obfuscation (b) Functional Obfuscation

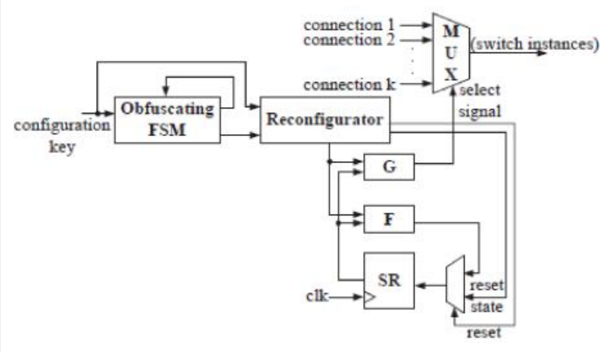


DESIGN FLOW OF THE EXISTING DSP CIRCUIT OBFUSCATION APPROACH

Design Methodology

In this section, we propose a novel DSP hardware protection methodology through obfuscation by hiding functionality via high-level transformations. This approach helps the designer to protect the DSP design against piracy by controlling the circuit configuration among the generated variation modes of the original design. The detailed design flow is described below:

Figure 2: Proposed Secure Switch Design



Step 1: DSP algorithm. This step generates the DSP algorithm based on the DSP application.

Step 2: High-level transformation selection. Based on the specific application, appropriate high-level transformation should be chosen according to the performance requirement (e.g., area, speed, power or energy).

Step 3: Obfuscation via high-level transformation. Selected high-level transformations are applied simultaneously with obfuscation where variation modes, and different configurations of the switch instances are designed.

Step 4: Secure switch design. The secure switch is designed based on the variations of high-level transformations. Note that different configure data could be mapped into the same mode, which only involves simple combinational logic synthesis.

Step 5: Two-level FSM generation. The reconfigurator and the obfuscating FSM are incorporated into the DSP design as shown in Figure 2. The configuration key is generated at this step.

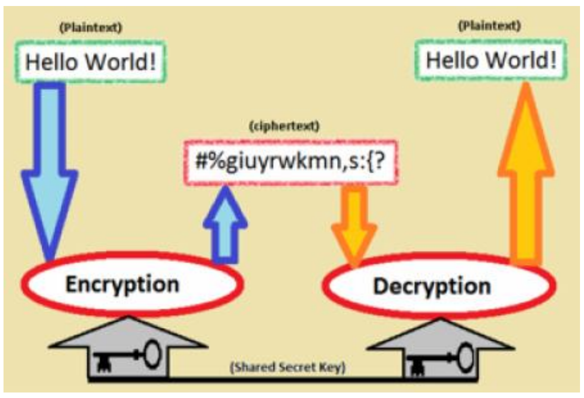
Step 6: Design specification. This step includes the HDL and netlist generation and synthesis of the DSP system.

The proposed design methodology does not require signif-icant changes to established verification and testing flows. In fact, the obfuscated DSP circuit with the correct key behaves just like the original circuit.

Architecture of the Obfuscated DSP Circuit

The complete system of the proposed obfuscated DSP circuit is illustrated in Figure 3. The DSP circuits are obfuscated by introducing a FSM whose state is controlled by a key. The FSM enables a reconfigurator that configures the functionality mode of the DSP circuit. High-level transformations lead to many equivalent circuits

Figure 3: Advanced Encryption Standard (AES)



and all these create ambiguity in the structural level. High-level transformations also allow design of circuits using same data path but different control circuits. For example, a data path may implement a 3rd-order or a 6th-order digital filter, or in general a (3l) order filter, where l is a positive integer. These correspond to different modes. While these modes generate outputs that are functionally incorrect, these may represent correct outputs under different situations, since the output is meaningful from a signal processing point of view. Finally, other modes lead to non-meaningful outputs. The initialization key and the configure data must be known for the circuit to work properly. Consequently, the circuit behaves as an obfuscated circuit.

The selective application of technological and related procedural safeguards is an important responsibility of every Federal organization in providing adequate security to its electronic data systems. This publication specifies a cryptographic algorithm, the Advanced Encryption Standard (AES) which may be used by Federal organizations to protect sensitive data. Protection of data during transmission or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by the data.

The algorithms uniquely define the mathematical steps required to transform data into a cryptographic cipher and also to transform the cipher back to the original form. The Advanced Encryption Standard is being made available for use by Federal agencies within the context of a total security program consisting of physical security procedures, good information management practices, and computer system/ network access controls.

Data encryption (cryptography) is utilized in various applications and environments. The specific utilization of encryption and the implementation of the AES will be based on many factors particular to the computer system and its associated components. In general, cryptography is used to protect data while it is being communicated between two points or while it is stored in a medium vulnerable to physical theft. Communication security provides protection to data by enciphering it at the transmitting point and deciphering it at the receiving point. File security provides protection to data by enciphering it when it is recorded on a storage medium and deciphering it when it is read back from the storage medium. In this the key must be available at the transmitter and receiver simultaneously during communication.

Cryptography is probably the most important aspect of communication security becoming increasingly important as a basic building block for computer security. Cryptographic systems are characterized along three independent dimensions:

The Type of Operations Used for Transforming Plaintext to Ciphertext: All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext is mapped into another element, and transposition, in which element in the plaintext are rearranged. The fundamental requirement is that no information be lost. Most systems referred to as product systems, involve multiple stages substitutions and transpositions.

The Number of Keys Used: If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.

The Way in Which the Plaintext is Processed: A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along. Before beginning, we define some terms. A symmetric encryption scheme has five ingredients:

Plaintext: This is the original intelligible message or data that is fed into the algorithm as input.

Encryption Algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.

Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.

Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertext. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.

Decryption Algorithm: This is essential the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

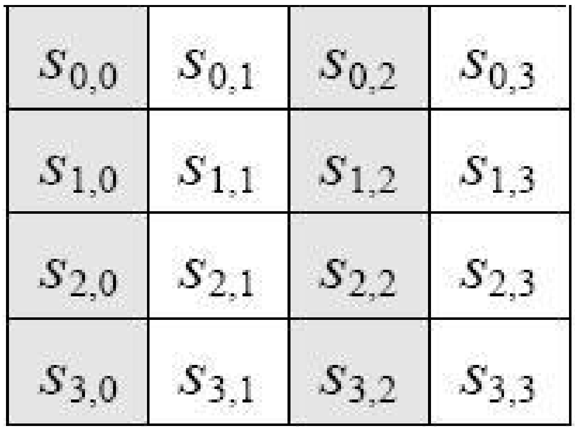
Advanced Encryption Algorithm (AES)

The Advanced Encryption Algorithm (AES), a symmetric block cipher algorithm that can processes blocks of 128-b, using cipher keys with lengths of 128,192 and 256-bits. The input and output for the AES algorithm each consist of sequence of 128-b (digit with values of 0 or 1). Nk = 8. The only key-block-round combination that confirms to the standard is given in Figure 5 [2]. Here we consider Nr = 10 and Nk = 4 for design of the architecture.

The Figure 3 shows the complete structure of AES algorithm (both encryption and decryption process) (Batra, 2005). In FIPS 197 standard, the block is depicted as square matrix of bytes. The block is copied into state array, which is modified at each stage of encryption or decryption processes. Similarly, the 128-b key is depicted as a square matrix of bytes.

The key is then expanded into an array of key schedule words, each word is of 4-bytes and the total key scheduled is 44 words for the 128-b key input.

Figure 4: State Array



$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

These sequences will sometimes refer to as blocks and the number of bits they contain will be referred to as their length. Internally, the AES algorithm's operation is are performed on a two-dimensional array of bytes called the state as shown in the Figure 4. The state consists of four rows of bytes, each containing Nb bytes, where

Figure 5: Block Diagram of AES

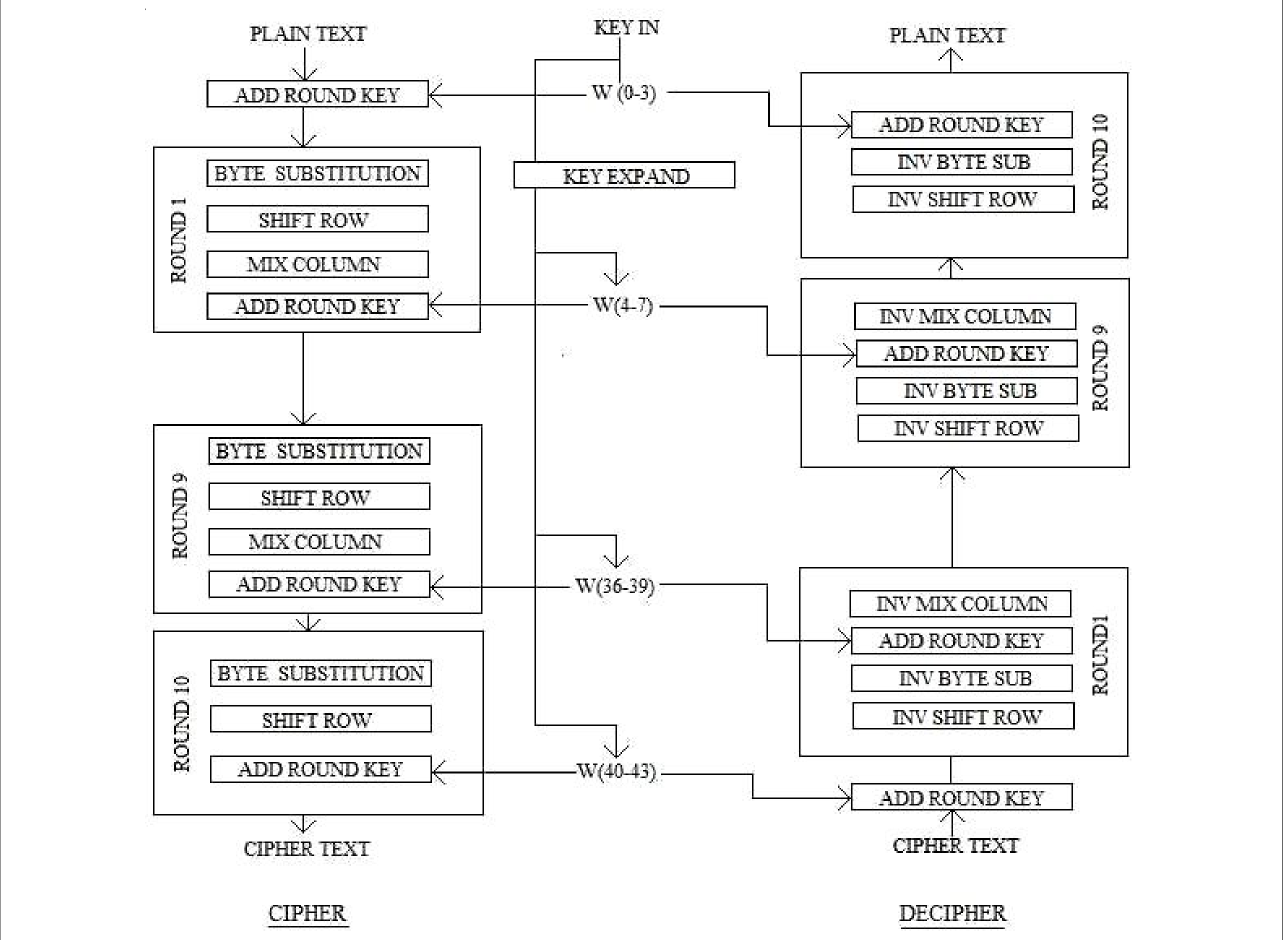


Table 1: AES Version Description Table

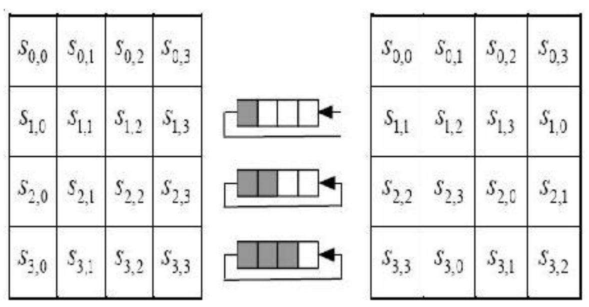
	Key Length (*Nk words*)	Block Size (*Nb words*)	Number of Rounds (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Nb is the block length. Here Nb = 4, which reflects the number of 32-b words (number of columns) in the state (Kirovski *et al.*, 1998).

For AES algorithm, the length of the cipher key, K, is 128,192 or 256 bits. The key length is represented by Nk = 4, 6, or 8, which reflects the number of 32-b words (number of columns) in the cipher key. 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

The Cipher and Decipher process is explained in the pseudo code in Figure 4 and Figure 5

Figure 6: Key-Block-Round Combination


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

(Kirovski *et al.*, 1998). The individual transformation – Byte substitution, Shift row, Mix column and Add round key- processes the state and is described in the following subsection. As shown in the Figure 4, all Nr rounds are identical with the exception of the final round, which does not include Mix column transformation and as such same in the decipher process shown in the Figure 5. Let us now see the detailed description of each of the four stages used in AES. For each stage both encryption and decryption transformation will be explained. This is followed by a discussion of key expansion process used in AES.

Byte Substitution

The substitute byte transformation, called the byte sub, is a simple table lookup. The process is shown in the Figure 6. AES defines a 16X16 matrix of byte values, called an S-Box that contains a permutation of all possible 256 8-b values which is shown in the Table 1. Each individual byte of the state is mapped into a new byte in the following way: The leftmost 4-b of the byte value is used as a row value and the rightmost 4-b are used as a column value. These row and column values serve as indexes into the S-Box to select a unique 8-b output value. For example, the hexadecimal value {95} references row 9, column 5 of the S-Box, which contains the value {ad}. Accordingly, the value {95} is mapped into the value {ad}.

The process of byte substitution is same for the decryption process but it makes use of inverse S-Box as shown in the Table 2, which is applied to each byte of the state (Kirovski *et al.*, 1998).

In this step, a normal left circular shift operation is done where in the first row is not altered and the next three rows are moved by 1, 2, 3 bytes respectively (Roy *et al.*, 2008). Conceptually, this is shown in the Figure.

Figure 7: AES S-Box

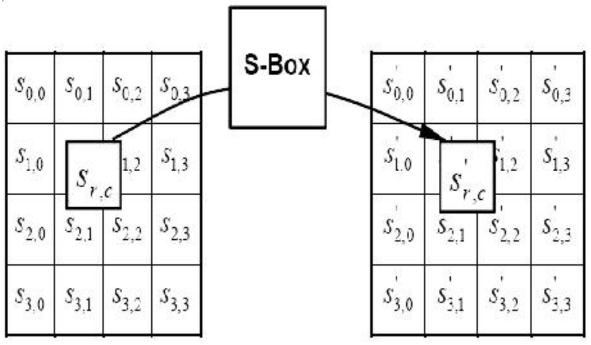
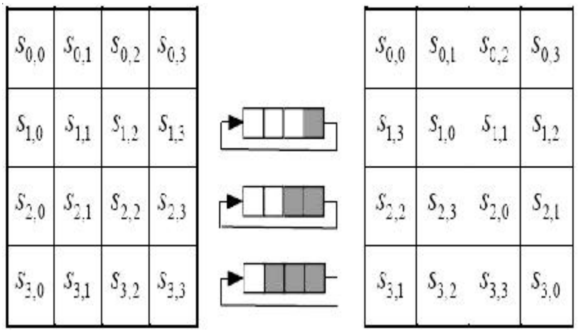


Figure 8: AES Shift Row



$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \ominus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Whereas in decryption transformation, the rows are subjected to right circular shift wherein the first row is untouched and row 2, 3 and 4 are shifted by 1, 2 and 3 bytes respectively. This process is shown in the Figure

$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

Let s'(x) = a(x) x s(x):

Mix Column

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

The mix column transformation operates on the state column-by-column, treating each column as a four term polynomial. The column are considered as polynomial.

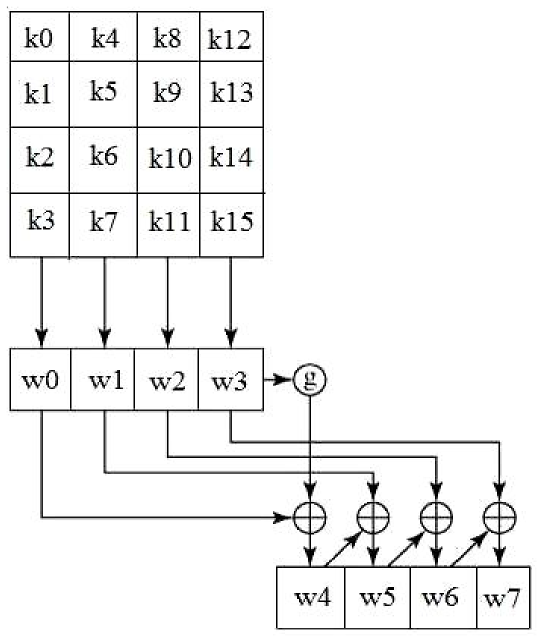
Key Expansion

The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The following pseudocode describes the expansion (Batra, 2005) KeyExpansion (byte key[16], word w[44])

```
{
    word temp
        for (i = 0; i < 4; i++)
  w[i]=(key[4*i],key[4*i+1],key[4*i+2],key[4*i+3]);
    for (i = 4; i < 44; i++)
        {
            temp = w[i-1];
            if (i mod 4 = 0)temp = SubWord (RotWord (temp)) XOR Rcon[i/4];
            w[i] = w[i-4] XOR temp
        }
    }
```

The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time. Each added word w[i] depend on the immediately preceding word, w[i-1], and the word four positions back, w[i-4]. In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used. Figure.11 illustrates the generation of the first eight words of the expanded key, using the symbol g to represent that complex function. The function g consists of the following subfunctions:

Figure 9: Key Expansion



1. RotWord performs a one-byte circular left shift on a word. This means that an input word [b0, b1, b2, b3] is transformed into [b1, b2, b3, b0].
2. SubWord performs a byte substitution on each byte of its input word, using the S-box.
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j] which is given in the Table 3 over GF (2^8) and multiplied with modulo x^4+1 with a fixed polynomial a(x),

Figure 10: Round Constant Rcon [j]

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

RESULTS

RTL Schematic

The RTL SCHEMATIC gives the information about the user view of the design. The internal blocks

Figure 11: Simulation Result and RTL Schematic

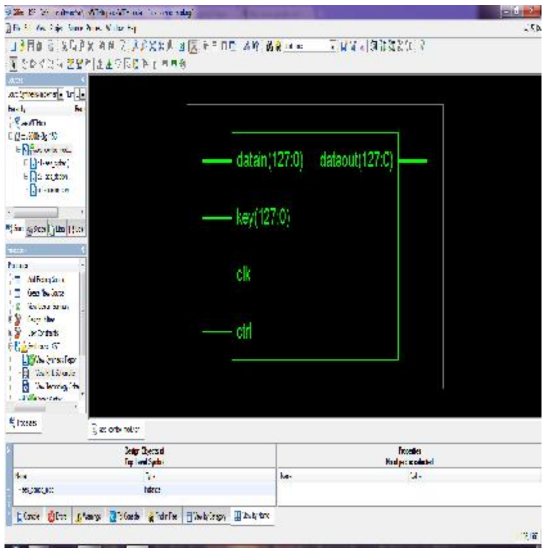


Figure 12: Internal Schematic

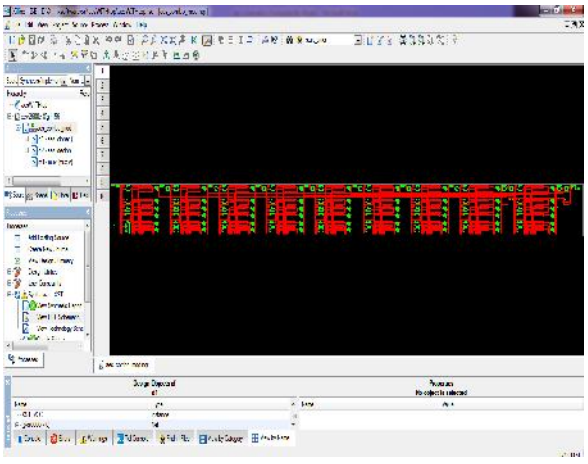
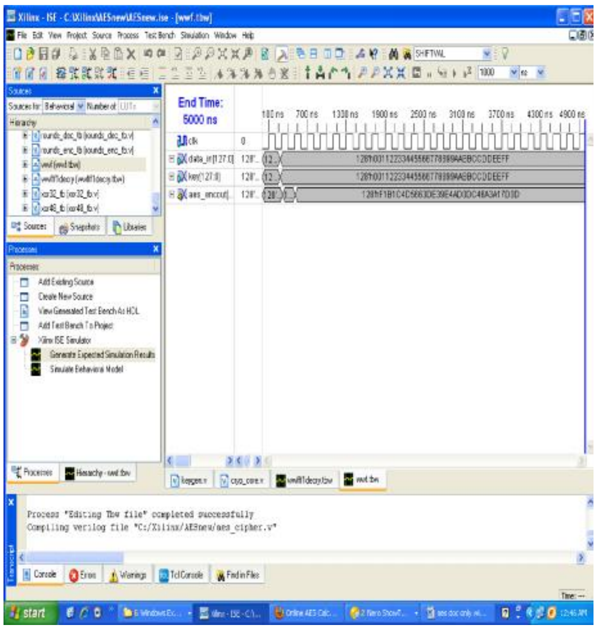


Figure 13: Simulation Result



128'h00112233445566778899AABBCCDDEEFF

128'h00112233445566778899AABBCCDDEEFF

128'hF1B1C4D5663DE39E4AD0DC48A3A17D3D

```
Process "Editing Tbw file" completed successfully
Compiling verilog file "C:/Xilinx/AESnew/aes_cipher.v"
```

contains the basic gate representation of the logic. These basic gate realization is purely depend upon the corresponding FPGA selection and the internal database information.

Waveform

In the waveform which is shown above, A and B signals represents the inputs which we are applying to the design. Similarly A x B is the output signal for the design. The constant WIDTH signal here is 32. To obtain the required outputs force the inputs logic with the required values. All the signals which are shown in the above waveform are in DECIMAL mode. Here in the waveform the multiplication operation is performed between the inputs A and B and the corresponding result is stored in the signal A x B.

CONCLUSION

In this project, we have given 128 bits input and 128 bits security key and observed how it is delivered at the output with security. In this project there is no revealing of the original message to the hackers. The original message can be revealed to only sender and the receiver. So, in future, any propriety information can be transmitted securely by using this project (military or banking purposes). Wide variety of applications, such as secure internet (ssl), electronic financial transactions, remote access servers, cable modems, secure video surveillance and encrypted data storage. The future scope of our project is to extend 128 bits inputs to n bits (n is any integer value). The functionality is verified by using XILINX ISE and the RTL is developed based on the VERILOG HDL language. In this project security oriented structural architecture along with maintaining functionality is carried out in terms of area, power and delay by maintaining the tradeoff.

REFERENCES

1. Alkabani Y M and Koushanfar F (2007), "Active Hardware Metering for Intellectual Property Protection and Security", in *Proceedings of the USENIX Security Symposium*, pp. 1-16.
2. Batra T (2005), "Methodology for Protection and Licensing of HDL IP", http://www.design-reuse.com/articles/12745.
3. Chakraborty R S and Bhunia S (2008), "Hardware Protection and Authentication Through Netlist Level Obfuscation", in *Proceedings of International Conference on Computer Aided Design ICCAD*, pp. 674-677.
4. Kirovski D, Hwang Y-Y, Potkonjak M and Cong J (1998), "Intellectual Property Protection by Watermarking Combinational Logic Synthesis Solutions", in *Proceedings of International Conference on Computer Aided Design (ICCAD)*, pp. 194-198.
5. Koushanfar F and Alkabani Y (2010), "Provably Secure Obfuscation of Diverse Watermarks for Sequential Circuits", in *Proceedings of International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 42-47.
6. Roy J A, Koushanfar F and Markov I (2008), "EPIC: Ending Piracy of Integrated Circuits", in *Proceedings of Design, Automation and Test in Europe (DATE)*.
7. Suh G E and Devadas S (2007), "Physical Unclonable Functions for Device Authentication and Secret Key Generation", in *Proceedings of the 44th Annual Design Automation Conference*, pp. 9-14.

ISSN 2319-5991

04

9 772319 599001