International Conference on Advances in Materials, Computing and Communication Technologies
In Association withInternational Journal of Scientific Research in Science and Technology
Volume 9 | Issue 1 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)

VLSI Implementation of Modified AES Algorithm

Geetha M, Devaki T, Dhakshayini R
Department of Electronics and communication Engineering, Velammal College of Engineering and Technology, Madurai, Tamil Nadu, India

ABSTRACT

Data transferred in an electronic way is vulnerable to attacks. With an aim to protect data for secure communication, a new Hybrid non pipelined Advanced Encryption Standard (AES) algorithm based on traditional AES algorithm with enhanced security features is proposed in this work. Abysmal analysis of the AES algorithm implies that the security of AES lies in the S-box operations. This paper presents a new approach for generating S-box values (modified S-box) and initial key required for encryption/encryption (improved key generation) using PN Sequence Generator. The AES algorithm with proposed modifications shows significant improvement in the encryption quality as compared to traditional AES algorithms. The traditional AES algorithm equipped with proposed novel modified S-box technique and improved key generation technique gives an avalanche effect of 60% making it invulnerable to attacks. The proposed design is synthesized on various Field Programmable Gate Array (FPGA) devices and compared to the existing designs resulting in significant improvement in throughput. The proposed design is implemented on Spartan6 FPGA devices.

I. INTRODUCTION

With the expansion of data communications and its applications, there is a greater demand for increasing security systems and devices to guard individual information sent over the transmission channel. One of the most important techniques for securing the information is data encryption. Two kinds of cryptographic techniques' viz. symmetric and asymmetric cryptosystems have already been created and are widely used.
Symmetric cryptography techniques, the Data Encryption Standard (DES), Triple DES and Advanced Encryption Standard (AES), utilize a key that is identical to the transmitter as well as a receiver, for encrypting and decrypting the data transferred. The Asymmetric cryptography techniques, the Rivest-Shamir-Adleman algorithm (RSA), Elliptic Curve Cryptography (ECC) and Digital Signature Algorithm (DSA) makes use of different keys for encrypting and decrypting the data transferred.
Analysis of cipher strength is an essential part of security assessment of any corporate or academic entity. Cipher analysts have indicated that, out of 10 rounds of AES, about 8 rounds can be brute forced successfully on today's modern day hardware systems. The remaining 2 rounds cannot be broken in sufficient duration to make the attack on the system meaningful to the attacker. Due to the current trend of increase in computational power, it may not be

long that the entire AES cipher would be deciphered under a given duration, compromise the system under test. Consequently, extensive research is currently carried out to identify techniques to secure the AES algorithm.

Triple DES and Advanced Encryption Standard (AES), utilize a key that is identical to the transmitter as well as a receiver, for encrypting and decrypting the data transferred. The Asymmetric cryptography techniques, the Rivest-Shamir-Adleman algorithm (RSA), Elliptic Curve Cryptography (ECC) and Digital Signature Algorithm (DSA) make use of different keys for encrypting and decrypting the data transferred. For securing large amounts of data symmetric cryptography is much more appropriate. The AES algorithm, identified by the National Institute of Standards and Technology (NIST) of the United States of America is approved to displace DES.

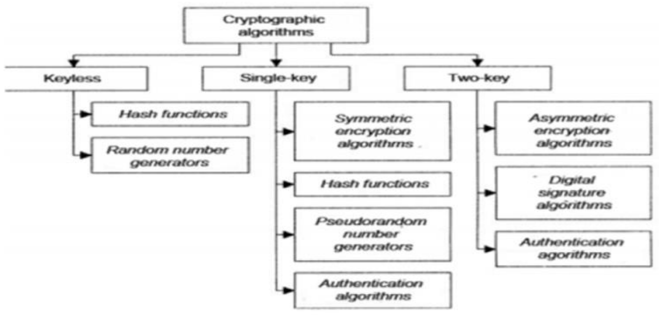

Cryptographic algorithms
Keyless
Single-key
Two-key
Hash functions
Random number generators
Symmetric encryption algorithms
Hash functions
Pseudorandom number generators
Authentication algorithms
Asymmetric encryption algorithms
Digital signature algorithms
Authentication agorithms

FIGURE 1.1 CLASSIFICATION OF CRYPTOGRAPHY

1.1 INTRODUCTION TO CRYPTOGRAPHY:

Cryptography is the science of secret codes, enabling the confidentiality of Communication through an insecure channel. It protects against unauthorized parties by preventing unauthorized alteration of use. Speaking, it uses a cryptographic system to transform a plaintext into a cipher text, using most of the time a key.

In a broader sense Cryptography is best known as a way of keeping the contents of a message secret. Confidentiality of network communications, for example, is of great importance for e-commerce and other network applications. However, the applications of cryptography go far beyond simple confidentiality. In particular, cryptography allows the network business and customer to verify the authenticity and integrity of their transactions. If the trend to a global electronic marketplace continues, cryptographic techniques will have to be developed to protect business transactions. Sensitive information sent over an open network scrambled into a form that cannot be understood by a hacker or eavesdropper. This is done using a mathematical formula, known as an encryption algorithm, which transforms the bits of the message into an unintelligible form. The intended recipient has a decryption algorithm for extracting the original message. There are many examples of information on open networks, which needs to be protected in this way, for instance, bank account details, credit card transactions, or confidential health or tax records. Cryptosystems can provide confidentiality, authenticity, integrity, and non-repudiation services. It does not provide availability of data or systems.

- Confidentiality means that unauthorized parties cannot access information.
- Authenticity refers to validating the source of the message to ensure the sender is identified.
- Integrity provides assurance that the message was not modified during transmission, accidentally or intentionally.
- Non repudiation means that a sender cannot deny sending the message at a later date, and the receiver cannot deny receiving it. if your boss sends you a message telling you that you will be receiving a raise that doubles your salary and it is encrypted, encryption methods can ensure that it really came from your boss, that someone did not alter it before it arrived to your computer, that no one else was able to read this message as it travelled over the network, and that your boss cannot deny sending the message later when he comes to his senses.

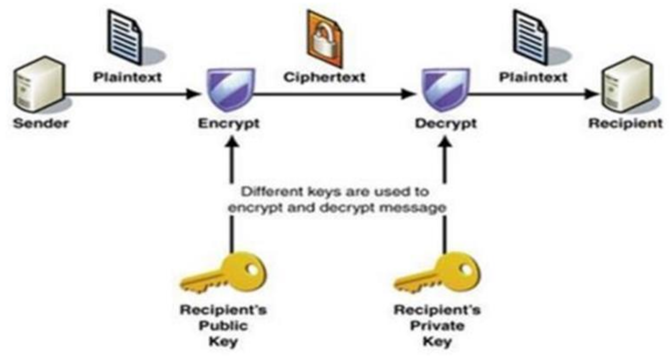


Figure 1.2 Overview Of Cryptography

1.2 PN SEQUENCE GENERATOR

A PN Sequence Generator is used for generation of S-box values and the initial key required for Encryption/Decryption. A PN Sequence Generator gives distinct values which satisfies the criterion for S-box values. These techniques result in enhancing the security of the AES algorithm as the feedback taps and seed value of the PN Sequence Generator are not known to an attacker and will make them more secure.

For securing large amounts of data symmetric cryptography is much more appropriate. The AES algorithm, identified by the National Institute of Standards and Technology (NIST) of the United States of America is approved to displace DES. Analysis of cipher strength is an essential part of security assessment of any corporate or academic entity. Cipher analysts have indicated that, out of 10 rounds of AES, about 8 rounds can be brute forced successfully on today's modern day hardware systems. The remaining 2 rounds cannot be broken in sufficient duration make the attack on the system meaningful to the attacker. Due to the current trend of increase in computational power, it may not be long that the entire AES cipher would be deciphered under a given duration, compromise the system under test.

Consequently, extensive research is currently carried out to identify techniques to secure the AES algorithm. In this work, a PN Sequence Generator is used for generation of S-box values and the initial key required for Encryption/Decryption. A PN Sequence Generator gives distinct values which satisfies the criterion for S-box values. These techniques result in enhancing the security of the AES algorithm as the feedback taps and seed value of the PN Sequence Generator are not known to an attacker and will make the algorithm invulnerable to brute force attack. The key contributions of this work are as follows:

A new approach for generation of S-box values using PN Sequence Generator is presented. A PN sequence generator generates a distinct sequence of random numbers based on the initial seed value and feedback taps. This property of a PN Sequence Generator is used for generating dynamic S-box which enhances the strength of the cryptosystem. This work presents a PN Sequence Generator based design of a Key Generation Block for generating initial key internally. A new secure initial key generation technique eliminates the need to apply the key externally strengthens the modified AES algorithm against attack as compared to traditional AES algorithms.

The techniques proposed in this paper are synchronized at both encryption and decryption ends for a secure AES algorithm. The AES algorithm with modified S-box values and improved key generation technique is tested on Strict Avalanche Criterion for key sensitivity and an avalanche effect of 60% is achieved. Also, the proposed design is optimized for speed and area and compared with existing FPGA implementations. The implementation of AES algorithm with modified S-box values using Spartan6 XC6SLX150-3FGG900 FPGA device achieves a throughput of 3.039 Gbps with latency of 10 clock cycles.

II. LITERATURE SURVEY

2.1. An efficient AES implementation using FPGA with enhanced security features.

Zodpe,H. and Sapkal, A., 2020. An efficient AES implementation using FPGA with enhanced security features. Journal of King Saud University-Engineering Sciences, 32(2), pp.115-122.

Data transferred in an electronic way is vulnerable to attacks. With an
aim to protect data for secure communication, a new Hybrid non pipelined Advanced Encryption Standard (AES) algorithm based on traditional AES algorithm with enhanced security features is proposed in this work. Abysmal analysis of the AES algorithm implies that the security of AES lies in the S-box operations. This paper presents a new approach for generating S-box values (modified S-box) and initial key required for encryption/encryption (improved key generation) using PN Sequence Generator. The AES algorithm with proposed modifications shows significant improvement in the encryption quality as compared to traditional AES algorithms. The traditional AES algorithm equipped with proposed novel modified S-box technique and improved key generation technique gives an avalanche effect of 60% making it invulnerable to attacks. The proposed design is synthesized on various Field Programmable Gate Array (FPGA) devices and compared to the existing designs resulting in significant improvement in throughput. The proposed design is implemented on Spartan6 FPGA device

2.2. Efficient implementation of the AES algorithm for security applications

Chodowiec, P. and Gaj, K., 2003, September. Very compact FPGA implementation of the AES algorithm. In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 319-333). Springer, Berlin, Heidelberg.
In this paper a compact FPGA architecture for the AES algorithm with 128-bit key targeted for low-cost embedded applications is presented. Encryption, decryption and key schedule are all implemented using small resources of only 222 Slices and 3 Block RAMs. This implementation easily fits in a low-cost Xilinx Spartan II XC2S30 FPGA. This implementation can encrypt and decrypt data streams of 150 Mbps, which satisfies the needs of most embedded applications, including wireless communication. Specific features of Spartan II FPGAs enabling compact logic implementation are explored, and a new way of implementing Mix Columns and Inverse Mix Columns transformation using shared logic resources is presented.

2.3. AES Encryption Algorithm Based on the High Performance Computing of GPU

Rao, M., Kaknjo, A., Omerdic, E., Toal, D. and Newe, T., 2018, October. An efficient high speed AES implementation using Traditional FPGA and LabVIEW FPGA platforms. In 2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 93-937). IEEE.
The LabVIEW FPGA platform is based on a graphical programming approach, which makes FPGA programming easy and the I/O interfacing. The LabVIEW FPGA significantly improves the design productivity and helps to reduce the time to market. On the other hand, a traditional FPGA platform is helpful to get an efficient/optimized design by providing control over each bit using HDL programming languages. This work utilized traditional as well as LabVIEW FPGA platforms to get an optimized high speed design of AES (Advanced Encryption Standard). The AES is a secure and reliable cryptographic algorithm that is used worldwide to provide encryption services, which hide the information during communication over untrusted networks, like the Internet. Here, AES core is proposed to secure the communication between ROV (Remotely Operated Vehicle) and control station in a marine environment; but this core can be fit in any other high speed electronic communications. This work provides encryption of 128-bytes, 256-

bytes and 512-bytes set of inputs (individually and simultaneously) using a 128-bit key. In simultaneous implementation, all the above mentioned set of inputs is encrypted in parallel. This simultaneous implementation resulted in throughput of Gbps range.

2.4. SMS encryption using AES on Android application

Ismaili, Z.E.A.A. and Moussa, A., 2009. Self-partial and dynamic reconfiguration implementation for AES using FPGA. ArXiv preprint arXiv:0909.2369.
This paper addresses efficient hardware/software implementation approaches for the AES (Advanced Encryption Standard) algorithm and describes the design and performance testing algorithm for embedded systems. Also, with the spread of reconfigurable hardware FPGAs (Field Programmable Gate Array) embedded cryptographic hardware became cost-effective. Nevertheless, it is worthy to note that nowadays, even hardwired cryptographic algorithms are not safe. From another side, the self-reconfiguring platform is reported that enables an FPGA to dynamically reconfigure itself under the control of an embedded microprocessor. Hardware acceleration significantly increases the performance of embedded systems built on programmable logic. Allowing an FPGA-based Micro Blaze processor to self-select the coprocessors used can help reduce area requirements and increase a system's versatility. The architecture proposed in this paper is an optimal hardware implementation algorithm and takes dynamic partially reconfigurable FPGA. This implementation is a good solution to preserve confidentiality and accessibility to the information in the numeric communication.

2.5. AES implementation on Xilinx FPGAs suitable for FPGA based WBSNs

Rao, M., Newe, T. and Grout, I., 2015, December. AES implementation on Xilinx FPGAs suitable for FPGA based WBSNs. In 2015 9th International Conference on Sensing Technology (ICST) (pp. 773-778). IEEE.
The Advanced Encryption Standard (AES) is a symmetric key Block cipher that is used to provide data confidentiality in many embedded systems. Data confidentiality of each information is very important, either the information is related with bank account statements, credit card numbers, trade secrets, government documents or personal information. The confidentiality of a patient's physiological data is an important issue in traditional wireless body sensor networks (WBSNs) due to the limited hardware resources, which makes traditional WBSNs not suitable for the implementation of security mechanisms. The Xilinx FPGAs (Field Programmable Gate Arrays) is a suitable option for FPGA based WBSNs, because of the availability of more logic resources and performance of FPGA. In this paper an FPGA based WBSN approach is discussed and an efficient implementation of AES is provided on latest Xilinx FPGAs (Artix-7, Virtex-7, Virtex-6, Virtex-4 and Spartan-6) that can be used to provide data confidentiality in FPGA based WBSNs. The presented efficient implementation technique of AES uses Block RAM resources of FPGA to get an optimized architecture with respect to power, speed and area. The results are provided throughput, slices, TPA and power. The XPA (Xilinx Power Analyzer) tool of Xilinx is used for power analysis.

III. EXISTING SYSTEM

3.1 Overview of AES algorithm

The AES algorithm performs operations on 12bit plaintext and uses identical keys for encryption as well as decryption. The AES algorithm processes facts, obstructs 128-bit parts and performs 10, 12 and 14 rounds of operations employing a cipher secret of duration 128-bits, 192-bits and 256-bits . The algorithm operates on a data block composed of a 44 byte matrix known as the state. The essential procedures of the AES algorithm are carried out on the state. The operations of AES Encryption algorithm with 128-bit key size.

The AES-128 algorithm can be divided in three stages viz. adding initial round key, rounds 1–9 and the final round. In the initial round, the 128-bit plaintext is Exclusive-O Red with 128-bit initial key. In each cipher round, Sub Bytes(), Shift Rows(), Mix Columns() and Add Round Keys() transformations are performed on a two-dimensional 44 arrays of bytes known the states. In the final round, Sub Bytes(), Shift Rows(), and Add Round Keys() operations are performed on the states.

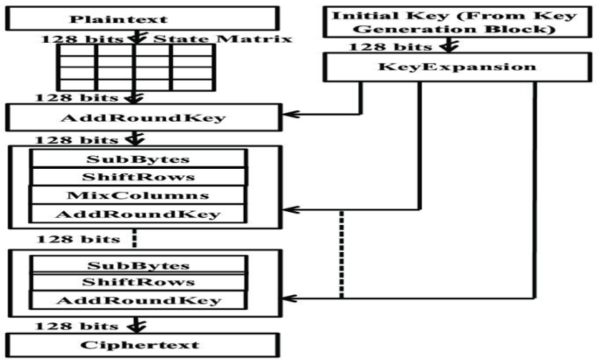


Figure 3.1 Block diagram of AES Algorithm

3.2 SubBytes transformation

The Sub Bytes transformation is the only non-linear and an invertible byte's transformation. It makes AES potent enough against attacks. The Sub Bytes transformation substitutes each byte from the state matrix with the value stored in S-box. The S-box is formed of a lookup table of size 256 bytes. The S-box values are calculated by taking a multiplicative inverse in a finite field where the input element with all bits zero is mapped to itself and applying affine transformation.

The multiplicative inverse in a finite field is given by The affine transformation is expressed

3.3 ShiftRows transformation

The Shift Rows transformation shifts rows 1, 2 and 3 of the State matrix cyclically towards left by 1, 2 and 3 positions . The offset value is dependent on the row number. Thus the 1st row remains unchanged. Cyclic rotation of rows imparts diffusion property in the AES algorithm. The Shift Rows transformation is represented.

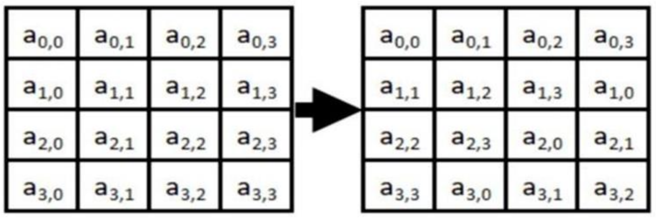

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$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

Figure 3.2 Shiftrows Transformation

3.4 MixColumns transformation

The MixColumns transformation performs operations on each column of the state matrix one at a time. It is a linear diffusion process. Each column of the state matrix is considered as a four-term polynomial over. The column is then multiplied by modulo(y4+ 1) with a fixed polynomial a(y).

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Figure 3.3 Mix Column Transformation

3.5 AddRoundKey transformation

The AddRoundKey transformation is the final transformation in each round. In this transformation the round key obtained is XORed with the state by bitwise operation. The Nb words from the key schedule of each Round Key are XORed with the columns of the State.

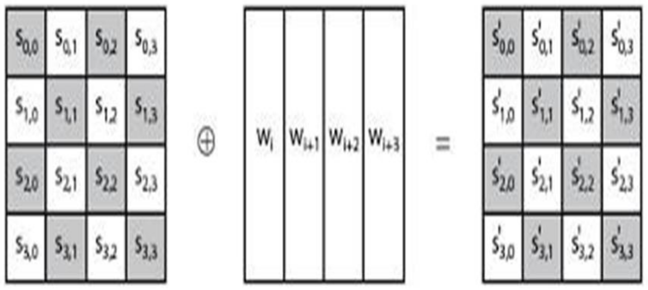

$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$\oplus$
W_i W_{i+1} W_{i+2} W_{i+3}
=
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{1,1}$ $S'_{1,2}$ $S'_{1,3}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$

Figure 3.4 AddRoundKey Transformation

3.6 Key expansion module

The key expansion module generates 128-bit keys required for each round of AES algorithm based on the initial 128-bit key. The key expansion module consists of Sub Bytes, Shift Rows and Round Const functions. Sub Bytes and Shift Rows functions are explained. The Round Cons function performs a bitwise XOR operation using a round constant array. Round constant array contains values given by [x i1, {00}, {00}, {00}] with xi1 being powers of x (x denoted as {02}) in the field. Thus each round key is generated column wise. Design approach for AES algorithm with enhanced security.

3.7 Design Approach of AES Algorithm With Enhanced Security

As compared to traditional AES algorithms, the proposed work suggests techniques to modify the S-box values using PN Sequence Generator to improve the quality of encryption. The initial key required for encryption/decryption is also generated using the PN Sequence Generator instead of using a pre-defined key. In the proposed work, 8-bit PN Sequence Generator is used for generating the S-box

values and initial key. The key and plaintext sensitivity.

The avalanche values for traditional AES algorithms are compared with avalanche values for AES algorithms with modified S-box values. For this comparison, pre-defined initial keys as well as initial keys generated using 8-bit PN Sequence Generator are considered. Further the proposed design is synthesized using different FPGA devices and comparison with existing FPGA implementations for speed and area optimization is done.

3.8 S-box values generation using 8-bit PN Sequence generator

A PN Sequence Generator is used to generate a sequence of pseudorandom binary numbers. A PN Sequence Generator is designed using Linear Feedback Shift Register (LFSR) described by the Generator Polynomial. LFSR is a shift register whose input bit is a linear function of previous state and is generated by XORing selected bits from all the bits of the shift register. The number of states generated by the LFSR is determined by the feedback taps of the Generator Polynomial. The S-box of AES algorithm consists of 256 distinct 8-bit values. For generation of these S-box values, 8-bit PN Sequence Generator is used with maximal length feedback taps to generate 281 = 255 random values across (01)hto (FF)h. The value (00) is randomly fed into the S-box. To generate, 8-bit maximum length sequence, the Generator polynomial can be set to a value from the following feedback taps viz. [8 6 5 2], [8 6 53], [8 6 5 4], [8 7 6 1], [8 6 4 3 2 1], [8 4 3 2], [8 7 6 5 2 1], [8 65 1], [8 5 3 1]. In this paper, the tap [8 6 5 4] is selected as a proof of concept and the generator polynomial formed is represented.

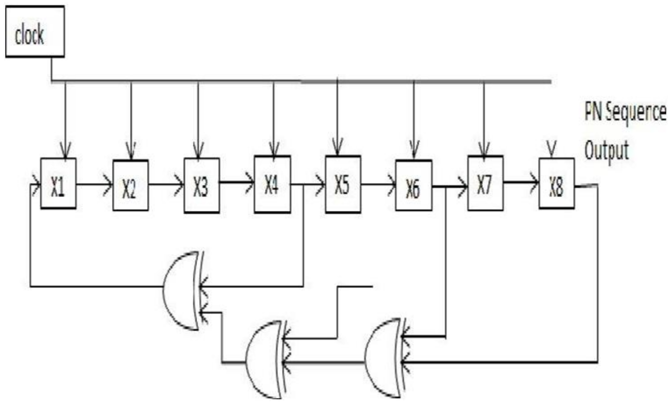


Figure 3.5 PN Sequence Generator

3.9 Key generation using 8-bit PN Sequence Generator

In the work, the key generation block uses an 8-bit PN Sequence Generator to generate the initial key required for the encryption/decryption process. The 8-bit PN Sequence Generator generates 255 values of 8-bit each which can be concatenated to form the 128-bit initial key. The key generation block internally selects the bytes for concatenation and forms the 128-bit key. The output states (each of 8-bit) of the PN Sequence Generator are stored in the Lookup Table (LUT) and are given as an input to 256:1 multiplexer as shown in Fig. 4. An 8-bit counter generates the value of 8-bit select lines of the multiplexer. The counter is designed to count 16 states as to select 16 input bytes and form the 128-bit (168) value at the output. Thus the key will consist of 16 distinct bytes and many distinct keys can be obtained by changing the initial value of the counter for select lines. These keys can then be dynamically applied to different blocks of 128-bit plaintext from the entire message to be encrypted. For example, setting feedback taps [8 6 5 4] and initial seed value as (1d) for the PN Sequence Generator and setting the initial value of the counter for select lines as (00)h, (1d0e070381c06030984c2693492492c9)h is the initial key generated for encryption/decryption.

The generation of key using the key generation block eliminates the need of using external predefined keys. Moreover, the generated key is dependent on the feedback taps and initial seed value of the PN Sequence Generator and change in these parameters will change the value of the initial key. In an adversary tries to decipher the data using brute force attack, then due to use of a different key for each message, the attacker will be unable to decrypt any useful information from the message. Further, the brute force attack will identify the initial key, but without the knowledge of the S-box. The attacker will not be able to decipher the input text. Thus these two modifications ensure a very high encryption quality for the cipher.

IV. PROPOSED SYSTEM

4.1 AES Algorithm

The AES is a private key block cipher that processes data blocks of 128 bits with key length of 128, 192, or 256 bits. The AES algorithm's operations are performed on a 2- D array of 4 times 4 bytes known the State. The initial State is the plaintext and the final State is the ciphertext. The State consists of 4 rows of bytes.

As the block length is 128 bits, each row of the State contains 4 bytes. The four bytes in each column form a 32 bit word. After an initial round key addition, a round function consisting of four transformations Sub Bytes, Shift Rows, Mix Columns and Add Round Key is applied to each data block. The round function is applied 10, 12, or 14 times Random Key and Key Dependent S-box Generation 65 depending on the key length. AES-128 applies the round function 10 times, AES-192 12 times, and AES-256– 14 times. The transformations are reversible linear and non-linear operations to allow decryption using their inverses. Every transformation affects all bytes of the State. The transformation Sub Bytes is a nonlinear byte substitution that operates on each byte of the State using a table (S-box). The numbers of the table is computed by a finite field inversion followed by an affine transformation. The resulting table is an S-box. The Shift Rows transformation is a circular shifting operation, which rotates the rows of the State with different numbers of bytes (offsets). The offset equals the row index: the second row is shifted one byte to the left, the third row two bytes to the left, the fourth row three bytes to the left and 1st row four bytes to the left.

Mix Columns transformation mixes the bytes in each column by multiplying the State with the polynomial modulo x4 +1. The State bytes are the coefficients of the polynomial. The Add Round Key transformation is an XOR operation that adds the round key to the State in each round. The initial round key equals the secret key.

4.2 Random Session Key Generation

` The random session keys are generated through cryptographically generated random numbers. In this method we can take advantage of the encryption logic available to produce random numbers. The AES encryption algorithm is used as the heart of the pseudorandom number generation. The procedure is to generate session keys from a master key Km, i.e., the 128-bit key to the main AES encryption algorithm. A counter with period N provides input to the encryption logic. In AES algorithm 128-bit keys are to be produced, a counter with period 2128 is used. After each session key is produced the counter is incremented by one. Thus, the pseudorandom numbers produced by this scheme cycle through a full period: Each of the outputs X0, X1, X2…. XN-1 is based on a different counter value and X0 ≠ X1 ≠ X2…. ≠XN-1. Because the master key is protected, it is not computationally feasible to deduce any of the secret keys through knowledge of one or earlier keys. The same plain text can generate different cipher texts using session keys. Brute force attackers will not be able to discover the key.

4.3 Substitution S-boxes

Substitution is a nonlinear transformation which performs confusion of bits.

A nonlinear transformation is essential for every modern encryption algorithm and is proved to be a strong cryptographic primitive against linear and differential cryptanalysis. Nonlinear transformations are implemented as lookup tables (S-boxes). An S-box with p input bits and q output bits are denoted p → q. The DES uses eight 6→4 S-boxes. S-boxes are designed for software implementation on 8-bit processors. The block ciphers with 8→8 S-boxes are SAFER, SHARK, and AES. For processors with 32-bit or 64-bit words, S-boxes with more output bits provide high efficiency.

The Snefru, Blowfish, CAST, and SQUARE use 8 → 32 S-boxes. The S-boxes can be selected at random as in Snefru, can be computed using a chaotic map, or have some mathematical structure over a finite Galois field. Examples of the last approach are SAFER, SHARK, and AES. S-boxes that depend on key values are slower but more secure than key independent ones.

In the AES, the S-box generates two transformations in the Galois fields GF(2) and GF(28). S-box is a nonlinear transformation where each byte of the State is replaced by another byte using the substitution table. The transformation: S-box finds the multiplicative inverse of the byte in the field GF(28). Since it is an algebraic expression, it is possible to mount algebraic attacks. Hence, it is followed by an affine transformation. The affine transformation is chosen in order to make the Sub Bytes a complex algebraic expression while preserving the nonlinearity property. The both S box transformations can be expressed in a matrix form.

4.4 Linear and Differential Cryptanalysis

Linear and differential cryptanalysis uses the input-output correlation and the difference propagations of the cipher in order to extract partial or whole bits of the secret key. Linear cryptanalysis exploits a cipher's weakness expressed in "linear expressions". In Matsui's terminology a linear expression for one round is an equation for a certain modulo two sum of round input bits and round output bits as a sum of round key bits. The expression should be satisfied with probability much more than 0.5 to be useful. In 1991 was introduced a crypto analytic technique known as differential cryptanalysis [14]. It was successfully applied to attack a variety of SPNs, including DES. Differential cryptanalysis requires knowledge of the XOR tables of S-boxes. For an n × n S-box, S, the XOR table has rows and columns indexed by 0, 1,…,2n -1, and the table entries are defined as follows: if i, j ∈{0, 1,…,2n-1}, position (i, j) in the XOR table contains value |{X ∈{0, 1}n : S(X) ⊕ S(X ⊕i)=j}|, s-block cipher is a S box.

To secure the cipher against these attacks, the nonlinearity of the S-box should satisfy: the maximum input-output correlation and the difference propagation probability should be minimum. There are two ways to fight against linear and differential cryptanalysis. One is built S-boxes with low linear and differential probabilities. The other is to design the round transformation that only trails with many active S-boxes occur. The round transformation designed in such a way that differential steps with few active S-boxes are followed by differential steps with many active S-boxes. The object of this proposal is an AES cipher using key-dependent S-boxes. The fact that the S-boxes are unknown is one of the main strengths of our cipher system, since both linear and differential cryptanalysis require known S-boxes. If the S-boxes are generated from the key in sufficiently random fashion, each S-box has a high probability of being complete, possessing fairly high nonlinearity. It is not apparent that the pseudo random nature of the S-boxes introduces any weakness into the system. Ideal randomness of S-box cannot be achieved. Ideal randomness is not mathematically possible for the following reasons: the value of all elements in the S box difference table should be even, since a-b = b-a. Since the S-box is bijective, the input difference of 0 will lead to an output difference of 0. The element corresponding to row = 0 and column = 0 at the difference table will be 2n and all other elements in row = 0 and column = 0 will be 0.

V. SIMULATION SOFTWARE DESCRIPTION

The electronics industry has achieved a phenomenal growth over the last two decades, due to the advent of VLSI. The number of applications of integrated circuits in high-performance computing, telecommunications and consumer electronics has been rising steadily and at a very fast pace. Typically, the required computational power (or, in other words, the intelligence) of these applications is the driving force for the fast development of this field. The current leading-edge technologies (low bit-rate video and cellular communications) already provide the end users a certain amount of processing power and portability. This trend is expected to be continued with very important implications of VLSI and systems design.

As more and more complex functions are required in various data processing and telecommunications devices, the need to integrate these function in the small system /package is also increasing . The level of integration as measured by the number of logic gates in a monolithic chip has been steadily rising for almost three decades, due to the rapid progress in processing technology and interconnect technology. Shows the evolution of logic complexity in integrated circuits over the last three decades, and marks the milestone of each era. Here, the numbers for circuit complexity should be interpreted only as representative examples to show the order of magnitude. A logic block can contain ten to a hundred transistors depending upon the function.

The important message here is that the logic complexity per chip has been increasing exponentially. The monolithic integration of many functions on a single chip usually provides:

- Less area / volume and compactness.
- Less power consumption.
- Less testing requirements at system level.
- Higher reliability, due to improved on-chip Interconnects.
- Higher speed, due to significantly reduced interconnection length.
- Significant cost savings.

Therefore, the current trend of integration will also continue in the foreseeable future.

A minimum size of 0.25 microns was readily achievable by the year 1995. As a direct result of this, the integration density has also exceeded previous expectations-the 64 Ambit Dramsand the Intel Pentium microprocessor chip containing more than 3 million transistors were already available by 1994,pushing the envelope of the integration density.

The design process, at various levels, is usually evolutionary in nature. It starts with a given set of requirements. Initial design is developed and tested against the requirements. When requirements are not met, the design has to be improved. If such improvement is either not possible or too costly, then the revision of requirements and its impacts analysis .

The VLSI design flow consists of three major domains, namely:

- Behavioral domain
- Structural domain
- Geometrical Layout domain.

The design flow starts from the algorithm that determines the behavior of the target chips. The corresponding architecture of the processor is defined. It is mapped onto the chip surface by floor planning. The next design evolution in the behavioral domain defines finite state machines(FSMs), which are structurally implemented with functional modules are registers and the arithmetic logic units (ALUs). These modules are then geometrically placed onto the chip surface using CAD tools for automatic module placement followed by routing, with a goal of minimizing the interconnects area and signal delays. The third evolution starts with behavioral modules that are then implemented with leaf cells. At this stage the chip is described logic gates (leaf cells) which can be placed and interconnected by using a cell placement and routing program. The last evolution involves a detailed Boolean description of leaf cells and mask generation. In standard-cell based design, leaf cells are already pre-designed and stored in a library for logic design use.

5.1 DESIGN HIERARCHY:

The use of hierarchy or "divide and conquer" technique involves dividing a module and then repeating this operation on the sub-modules until the complexity of the smaller parts becomes manageable. This approach is very similar to the software where large programs are split into smaller and smaller sections until simple subroutines, with well-defined functions and interfaces can be written. The design of VLSI chips can be represented in three domains. Correspondingly, a hierarchy structure can be described in each domain separately. It is important for the simplicity of design that the hierarchies in different domains can be mapped into each other easily.

In the physical domain, partitioning a complex system into its various functional blocks will provide a valuable guidance for the actual realization of these blocks on chip. Obviously, the approximate shape and size (area) of each sub-module should be estimated in order to provide a useful floor plan.

5.2 VLSI DESIGN STYLES:

Several design styles can be for chip implementation of

specified algorithms or logic functions. Each design style has its own merits and shortcomings, a proper choice has to be made by designers in order to provide the functionality at low cost.

5.3 VLSI DESIGN PRESENT SCENARIO:

Programmable logic devices viz FPGA, CPLDs and their design tools have changed dramatically in the last 15 years, today with the advent of million gates, high performance, system level devices, the concept of SOC(system on chip) has arrived. Now you can create unique designs that were never possible before, get them to market longer, as a designer ,your "windows of innovation" is practically unlimited.

The FPGA and CPLD market is increasing and is about to grow more than the ASIC market, both device and value.

Companies like Xilinx, Altera, Quick logic, Vantis Lattice and Cypress are the major players in this market. A Million gate devices from them are already available. PLDs are one finding applications in areas, which formerly were the domain of ASICs only.

5.4 OVERVIEW OF HDL:

Computer hardware and electronics industries, ever since their inception, have been working hand in hand. Technology development in either of them helped and supported the other. For over a decade, computer researchers and programmers, and electronic component manufacturers have been on the lookout for a tool that can bridge the gap of exchanging data between designers and chip manufacturing companies.

A Hardware Description Language (HDL) can be used to model and explain the behavior of any electronic component. The component can be as small as a gate or as complex as a complete multi-layered circuit board or a digital system.

An HDL provides mechanisms to specify design specifications that are clear, unambiguous and simple from a small basic component till the complete system has been designed Today, we have many HDL including the popular ones such as VERILOG HDL and Very High-Speed Integrated Circuit HDL (VHSIC-HDL) and others such asISP, UDLI, Abel, and HiLo.

5.5 Behavioural Modelling & Timing in Verilog

Behavioral models in Verilog contain procedural statements, which control the simulation and manipulate variables of the data types. These all statements are contained within the procedures. Each of the procedure has an activity flow associated with it.

During simulation of the behavioral model, all the flows defined by the 'always' and 'initial' statements start together at simulation time 'zero'. The initial statements are executed once, and the always statements are executed repetitively. In this model, the register variables a and b are initialized to binary 1 and 0 at simulation time 'zero'. The initial statement is then completed and is not executed again during that simulation run. This initial statement is containing a begin-end block (also called a sequential block) of statements. In this begin-end type block, an is initialized followed by b.

5.5.1 Procedural Assignments

It is for updating reg, integer, time, and memory variables. There is a significant difference between procedural assignment and continuous assignment as described below –

Continuous assignments drive net variables and are evaluated and updated whenever an input operand changes value.

Procedural assignments update the value of register variables under the control of the procedural flow constructs that surround them.

The right-hand side of a procedural assignment can be any expression that evaluates to a value. However, part-selects on the right-hand side must have constant indices. The left-hand side indicates the variable that receives the assignment from the right-hand side. The left-hand side of a procedural assignment can take one of the following forms –

- register, integer, real, or time variable – An assignment to the name reference of one of these data types.
- bit-select of a register, integer, real, or time variable – An assignment to a single bit that leaves the other bits untouched.
- part-select of a register, integer, real, or time variable – A part-select of two or more contiguous bits that leaves the rest of the bits untouched. For the part-select form, only constant expressions are legal.
- memory element – A single word of a memory. Note that bit-selects and part-selects are illegal on memory element references.
- concatenation of the above – A concatenation of the previous four forms can be specified, which effectively partitions the result of the right-hand side expression and assigns the partition parts, in order, to the various parts of the concatenation.

5.5.2 Delay in Assignment (not for synthesis)

In a delayedassignmentΔttime units pass before the statement is executed and the left-hand assignment is made. With intra-assignment delay, the right side is evaluated immediately but there is a delay of Δt before the result is placed in the left-hand assignment. If another procedure changes a right-hand side signal during Δt, it does not affect the output. Delays are not supported by synthesis tools.

5.5.3 Blocking Assignments

A blocking procedural assignment statement executed before the execution of the statements that follow it in a sequential block. A blocking procedural assignment statement does not prevent the execution of statements that follow it in a parallel block.

5.5.4 Non Blocking (RTL) Assignments

The non-blocking procedural assignment allows you to schedule assignments without blocking the procedural flow. You can use the non-blocking procedural statement whenever you want to make several register assignments within the same time step without regard to order or dependence upon each other.

5.6 Looping Statements

There are four types of looping statements. They provide a means of controlling the execution of a statement zero, one, or more times.

- forever continuously executes a statement.
- repeat executes a statement a fixed number of times.
- while executes a statement until an expression becomes false. If the expression starts out false, the statement is not executed at all.
- for controls execution of its associated statement(s) by a three-step process, as follows –
    - Executes an assignment normally used to initialize a variable that controls the number of loops executed
    - Evaluates an expression—if the result is zero, the for loop exits, and if it is not zero, the for loop executes its associated statement(s) and then performs step 3
    - Executes an assignment normally used to modify the value of the loop control variable, then repeats step 2

VI. RESULT AND DISCUSSION

6.1 Base Work Result

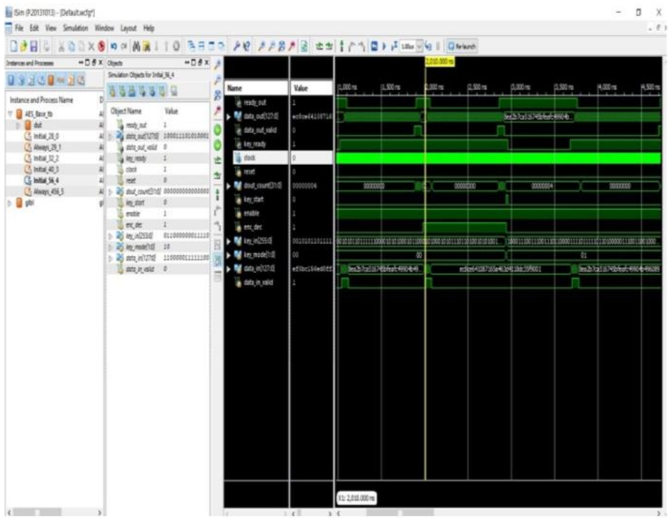
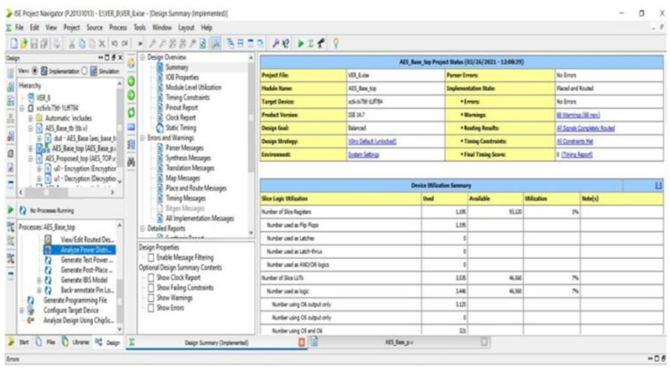


Figure 6.1 Simulation results base work

Here ,the figure 6.1 represents the simulation result of the base paper. The data_in represents plain text which we give ,then it gives the cipher text after the encryption process which is represented as data_out valid. The encrypted text is decrypted in the receiver side by giving the key which is mentioned in the figure as data_out.

Base- Area 1195



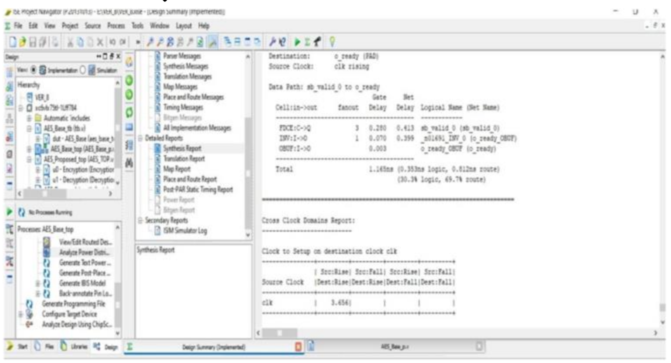
AES_Base_top Project Status (03/26/2021 - 12:08:29)			
Project File:	VER_8.xise	Parser Errors:	No Errors
Module Name:	AES_Base_top	Implementation State:	Placed and Routed
Target Device:	xc6vlx75t-1ff784	• Errors:	No Errors
Product Version:	ISE 14.7	• Warnings:	68 Warnings (68 new)
Design Goal:	Balanced	• Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	All Constraints Met
Environment:	System Settings	• Final Timing Score:	0 (Timing Report)

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	1,195	93,120	1%	
Number used as Flip Flops	1,195			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	3,535	46,560	7%	
Number used as logic	3,446	46,560	7%	
Number using O6 output only	3,125			
Number using O5 output only	0			
Number using O5 and O6	321			

Figure 6.2 AES architecture Area

The total number of slice registers available in the FPGA device is 93,120. The number of slice registers which are used by the program is 1,195. The utilization percentage is 1. In the proposed work we are going to minimize the number of slice registers as maximum as possible. The number of slice registers mentioned here is representing the area used by the program. By minimizing the area we can improve the performance of the project.

Base - Latency 1.165ns



```
Destination:       o_ready (PAD)
Source Clock:      clk rising

Data Path: sb_valid_0 to o_ready
                          Gate    Net
  Cell:in->out   fanout   Delay   Delay  Logical Name (Net Name)
  ---------------------------------------  ------------
   FDCE:C->Q         3    0.280   0.413  sb_valid_0 (sb_valid_0)
   INV:I->O          1    0.070   0.399  _n01691_INV_0 (o_ready_OBUF)
   OBUF:I->O              0.003          o_ready_OBUF (o_ready)
  ---------------------------------------
  Total                   1.165ns (0.353ns logic, 0.812ns route)
                                  (30.3% logic, 69.7% route)

=========================================================================

Cross Clock Domains Report:
--------------------------

Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |    3.656|         |         |         |
---------------+---------+---------+---------+---------+
```

Figure 6.3 AES architecture Base latency

Latency is the delay between a user's action and a web application'sresponse to that action, often referred to in networking terms as the total round trip time it takes for a data packet to travel. It decreases the performance of the software.Thereforewe need to reduce the latency period in the proposed work. Here the latency period is 1.165ns. If latency increases the speed will decrease,and vice versa.

Base - Power Consumption 82%

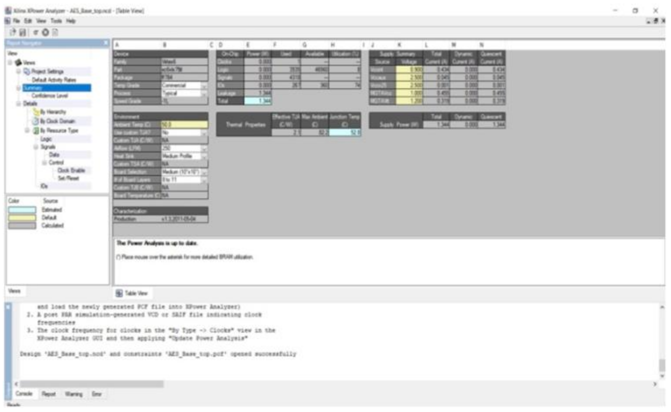


Figure 6.4 AES Architecture Base Power Consumption

Power consumption is usually measured in units of watts (W) or kilowatts (kW). The energy used by equipment is always more than the energy really needed. The power taken by the machine is 1344w. The power used by the base work is (74+8)%. We all know that the maximum power consumption will decrease the performance. Therefore, in the proposed work we need to minimize the percentage usage of the power consumption.

6.2 Proposed Work Result

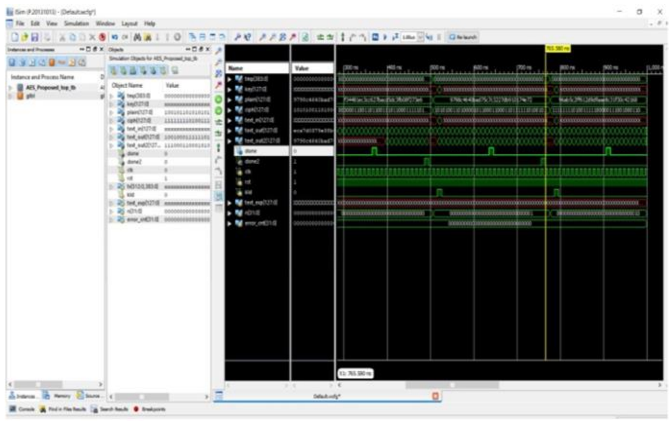
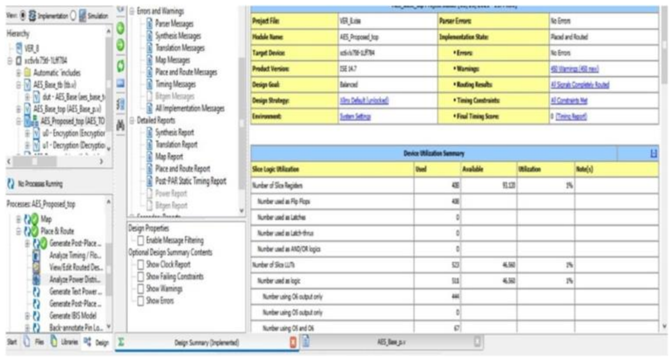


Figure 6.5 Proposed Work simulation Result

The simulation result of the proposed work is show in figure 6.5 which includes the random key generation and the PN sequence generator modification . Here the plain text is encrypted and gives the cipher text differently in multiple attempts of the same plain text which improves the security of our information. Whenever the done is 1 , it is decrypted for the receiver in this program. The decrypted text is represented here as the text_out2 in the above figure.

Proposed -Area 408



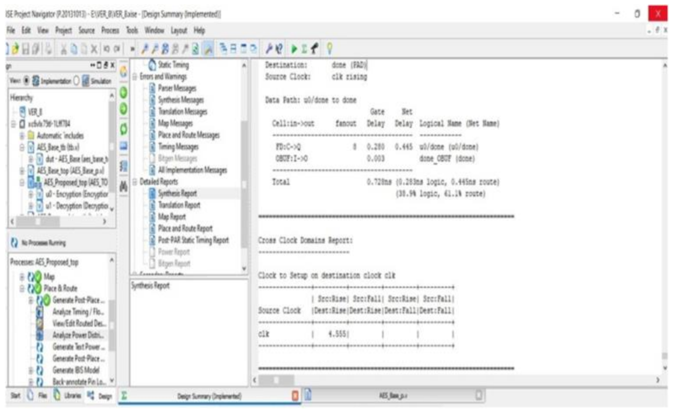
Project File:	VER_8.xise	Parser Errors:	No Errors
Module Name:	AES_Proposed_top	Implementation State:	Placed and Routed
Target Device:	xc6vlx75t-1ff784	• Errors:	No Errors
Product Version:	ISE 14.7	• Warnings:	48 Warnings (48 new)
Design Goal:	Balanced	• Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	All Constraints Met
Environment:	System Settings	• Final Timing Score:	0 (Timing Report)

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	408	93,120	1%	
Number used as Flip Flops	408			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	523	46,560	1%	
Number used as logic	513	46,560	1%	
Number using O6 output only	444			
Number using O5 output only	0			
Number using O5 and O6	67			

Figure 6.6 AES Architecture Proposed Area

The total number of slice registers available in the FPGA device is 93,120. The number of slice registers which are used by the program is 408. The utilization percentage is 1. In the proposed work the number of slice registers is 1195, we reduced it to 408. Therefore, the area used by the software reduces which Automatically increases the performance.

Proposed - Latency 0.728ns



```
Destination:       done (PAD)
Source Clock:      clk rising

Data Path: u0/done to done
                              Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   FD:C->Q               8   0.280   0.445  u0/done (u0/done)
   OBUF:I->O                 0.003          done_OBUF (done)
  ----------------------------------------
  Total                      0.728ns (0.283ns logic, 0.445ns route)
                                     (38.9% logic, 61.1% route)

=========================================================================

Cross Clock Domains Report:
--------------------------

Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |    4.555|         |         |         |
---------------+---------+---------+---------+---------+

=========================================================================
```

Figure 6.7 AES Architecture Proposed Latency

The latency period in the base work is 1.165ns . The proposed work reduces the latency period to 0.728ns. The reduction in latency period increases the speed of the process. we can calculate the speed by using the latency period .

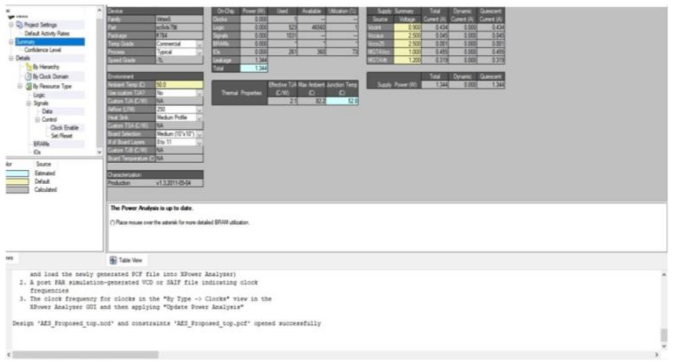
SPEED =1/LATENCY

=1/0.728ns

SPEED = 1373Mhz

In the base work ,the speed of the process is 858Mhz . It is increased to 1373Mhz in the proposed work.

Proposed - Power consumption 74%



The Power Analysis is up to date.

Place mouse over the asterisk for more detailed BRAM utilization.

```
     and load the newly generated PCF file into XPower Analyzer)
  2. A post PAR simulation-generated VCD or SAIF file indicating clock
     frequencies
  3. The clock frequency for clocks in the "By Type -> Clocks" view in the
     XPower Analyzer GUI and then applying "Update Power Analysis"

Design 'AES_Proposed_top.ncd' and constraints 'AES_Proposed_top.pcf' opened successfully
```

Figure 6.8 AES Architecture Proposed power consumption

Power consumption is usually measured in units of watts (W) or

kilowatts (kW). The energy used by equipment is always more than the energy really needed. The power taken by the machine is 1344w. The power used by the proposed work is (73+1)%. We all know that the maximum power consumption will decrease the performance. Therefore ,we reduced the power consumption from 82% to 74%.

6.3 Performance Metrics:

	Area	Speed	Power Consumption (712mwatts)	Latency	Throughput
Base AES	1195	858 MhZ	82%	1.165ns	6.7Gbps
Proposed AES	408	1373 Mhz	74%	0.728ns	9.94Gbps

From the above table ,the performance metrics such as area,power consumption, latency are comparatively reduced from the base paper and then the speed and throughput are comparatively increased from the base paper result. Hence we achieved the modified implementation of the AES algorithm.

6.4 APPLICATIONS OF AES ALGORITHM IN VARIOUS FIELD

Secure Communication

- ATM
- DVD C
- Secure Networks
- Secure video surveillance systems
- IEEE 802.11i (Wi-Fi); IEEE 802.15.3, IEEE 802.15.4 (Zigbee), MBOA (WiMedia), 802.16e, Wibree.
- Secure Storage
- Defence application
- Confidential Corporate Documents
- Government Documents
- FBI Files
- Personal Storage Devices

6.5 ADVANTAGES OF AES ALGORITHM

- AES can be implemented on both hardware and software
- It is the most robust security protocol.
- It supports larger key sizes (upto 256 bits).
- It consumes less power & offers faster speed.

VII. CONCLUSION

Encryption using hardware platforms is widely being used in order to secure data and enhance throughput. In this paper, techniques to enhance the encryption quality of AES algorithm and its implementation on FPGA are proposed. First, the S-box values in the modified AES algorithm are generated using PN Sequence Generator. Second, the initial key required for encryption/decryption is also based on the output of PN Sequence Generator. The result of encryption for the modified AES algorithm is tested on the Strict Avalanche Criterion for 2048 variations and the average percentage avalanche effect is achieved for the modified AES algorithm as compared to the traditional AES algorithm. Thus the modifications suggested, results in improved quality of encryption. FPGAs are used for efficient hardware implementation of the modified AES algorithm. The results for throughput and area are compared with existing non-pipelined and pipelined designs and are found to achieve better performance.

VIII. REFERENCES

[1]. Zodpe, H. and Sapkal, A., 2020. An efficient AES implementation using FPGA with enhanced security features. Journal of King Saud University-Engineering Sciences, 32(2), pp.115-122..

[2]. Chodowiec, P. and Gaj, K., 2003, September. Very compact FPGA implementation of the AES algorithm. In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 319-333). Springer, Berlin, Heidelberg.

[3]. Rao, M., Kaknjo, A., Omerdic, E., Toal, D. and Newe, T., 2018, October. An efficient high speed AES implementation using Traditional FPGA andLabVIEW FPGA platforms. In 2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 93-937). IEEE.

[4]. Ismaili, Z.E.A.A. and Moussa, A., 2009. Self-partial and dynamic reconfiguration implementation for AES using FPGA. arXiv preprint arXiv:0909.2369.

[5]. Rao, M., Newe, T. and Grout, I., 2015, December. AES implementation on Xilinx FPGAs suitable for FPGA based WBSNs. In 2015 9th International Conference on Sensing Technology (ICST) (pp. 773-778).IEEE.

[6]. Jeff Moser."A stick Figure Guide to the Advanced Encryption Standard(AES) Colin Percival."Cryptographic Right Answers"

[7]. _Encryption_Standard_Algorithm_for_Information_SecurityJeffrey Goldberg."Guess Why We're moving to 256-bit AES keys"

[8]. Bruce Schneier."Another New AES Attack".quote:"AES-128 provides more than enough security margin for the foreseeable future. Biclique Cryptanalysis of the Full AES'' (PDF). Archived from the original (PDF) on March 6, 2016. Retrieved May 1, 2019.

[9]. John Schwartz (October 3, 2000). "U.S. Selects a New Encryption Technique". New York Times. Archived from the original on March 28, 2017.

[10]. Bernstein, Daniel. "Why switch from AES to a new stream cipher". cr.yp.to. Retrieved 17 February 2021.

[11]. Ahmad, N., Hasan, R., Jubadi, W.M., 2010. Design of AES S-box using combinational logic optimization. IEEE Symp. Ind. Electron. Appl.,696-699

[12]. Alexandru, C., Fratila, R., 2011. Cryptographic Applications using FPGA Technology. Mobile Embedded Distrib. Syst. 3, 10–16.

[13]. Ali, L., Aris, I., Hossain, F.S., Roy, N., 2011. Design of an ultra high speed AES processor for next generation IT security. Comput. Electr.Eng. 37, 1160–1170.

[14]. Bogdanov, A., Khovratovich, D., Rechberger, C., 2011. Biclique cryptanalysis of the Full AES. Adv. Cryptol. ASIACRYPT 2011, 344–371.https://doi.org/10.1007/978-3-642-25385-0. Chih Peng,

[15]. F., Hwang, J.K., 2008. FPGA implementations of high throughput sequential and fully pipelined AES algorithm. Int. J. Electr. Eng. 15, 447–455.

[16]. Chih-Pin, S., Tsung-Fu, L., Chih-Tsun, H., Cheng-Wen, W., 2003. A high-throughput low-cost AES processor. IEEECommun. Mag., 86–91

[17]. Chodowiec, P., Gaj, K., 2003. Very Compact FPGA implementation of the AES algorithm. CHES 2003. LNCS 2779, 319–333.

[18]. Ebrahim, M., Khan, S., Khalid, U. Bin, 2014. Symmetric algorithm survey: a comparative analysis. Int. J. Comput. Appl. 61, 12–19.

[19]. Elbirt, A.J., Yip, W., Chetwynd, B., Paar, C., 2001. An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. IEEE Trans. Very Large Scale Integr. Syst. 9, 545–557.https://doi.org/10.1109/92.931230.

[20]. Farooq, U., Aslam, M.F., 2017. Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPG

[21]. A. J. King Saud Univ. Comput. Inf. Sci. 29, 295–302.

[22]. Fips-197, 2001. Advanced encryption standard (AES). Natl. Inst. Stand.Technol. 8–12.https://doi.org/10.1016/S1353-4858(10)70006-4.

[23]. Gaj, K., Chodowiec, P., 2009. FPGA and ASIC implementations of AES.Cryptogr. Eng.235–294.https://doi.org/10.1007/978-0-387-71817-0.Gangadari, B.R., Rafi Ahamed, S., 2016.

[24]. Design of cryptographically secure AES like S-Box using second-order reversible cellular automata for wireless body area network applications.Healthcare Technol. Lett. 3, 177–183.