Higher Level Synthesis Framework Implementation of

AES 128-bit Encryption Decryption

Afaq Younas 051-16-124114

Muhammad Shaheer Aamir

  051 - 15 - 123404

  Shah Nawaz 051 - 15-123450



Mr. Arslan Majid

A Final Year Project Report is Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Science in Telecommunication & Networks

Department of Computing & Technology Iqra University, Islamabad Campus

April 2019

Certificate

We hereby accept the work contained in this report titled: *Voice Encryption over IP Network*, as a confirmation to the required standards for the partial fulfillment of the degree of Bachelors of Telecommunication and Networks.



*Internal Examiner External Examiner*



*Project Supervisor Head of Department*

# Declaration

We hereby declare that this work, neither whole nor in part, has been copied from any source. It is further declared that I have prepared this report entirely on the basis of my personal efforts made under the sincere guidance of teachers especially my supervisor Mr. Arslan Majid. If any

 part of this thesis is proved to be copied out from any source or found to be reproduction of some

other, I will stand by the consequences. No portion of the work presented has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Afaq Younas

Muhammad Shaheer Aamir Shah Nawaz

iii

# Dedication

We dedicate this project to our parents who did financial and moral support. We would also like to dedicate this project to our supervisor Mr. Arslan Majid.

# Acknowledgements

We would definitely like to offer our thanks to ALLAH for his help and blessing throughout our studies. We might also want to offer a huge thanks to our supervisor, mentor, Mr. Arslan Majid for helping and guiding us till the end of this project. His specialized and research guidance was fundamental to the finishing of this FYP and has shown our countless lessons and bits of knowledge on the operations of academic research. Lastly, we would extend our thanks to our family for their unconditional support and immeasurable love.

# Abstract

Cryptography assumes a vital job in security of information transmission. The improvement of registering innovation forces more grounded necessities on the cryptography plans. In 2000, the Advanced Encryption Standard (AES) supplanted the DES to conquer the expanding

 prerequisites for security. In cryptography, the AES, likewise called as Rijndael, is a square figure that is received as an encryption standard by the USA government, which determines an encryption calculation fit for securing private and touchy data. This calculation is a symmetric square figure that can encode and unscramble the data. Encryption changes over information into a garbled structure known as figure content. Unscrambling of the figure content believers the information once again into its unique structure, that is called plaintext. The AES calculation is underpinning keys length of 128, 192, and 256 bits to encode and decode information in squares of 128 bits, along these lines the name moves toward becoming AES-128, AES-192 and AES256 individually. The equipment execution of the AES calculation can give elite, ease for explicit applications and dependability contrasted with its product partners.

As the need of more and more cryptographic systems is emerging there is also an area of concern of computational power and response in terms if speed. A frame work developed using a parallel

 processing network is developed which performs the encryption and decryption process at a much faster rate. Coding for the frame work was done by using a higher level synthesis tool. The synthesis tool then converted the blocks coded in C into synthesizable Verilog module. The modules were then tested in waveform analyzers and compared with standard open source Verilog implementation.

Experiment results revealed that our methodology produced correct output results whereas it achieved a slightly better speed due to its parallel processing design.

Contents

[Declaration iii](#_TOC_250010)

[Dedication iv](#_TOC_250009)

Acknowledgemens v

Absrac vi

Lis of Figures x

Lis of Tables xi

[Chapter 1 Introduction 1](#_TOC_250008)

* 1. [Overall Description 2](#_TOC_250007)
     1. Objectives 2
     2. Problem Description 2
     3. Methodology 2
     4. Product Scope 3
     5. User Classes and Characteristics 3
     6. Operating Environment 3
     7. Assumptions and Dependencies 3
  2. [External Interface Requirements 4](#_TOC_250006)
     1. User Interfaces 4
     2. Hardware Interfaces 5
     3. Software Interfaces 5
     4. Communications Interfaces 5
  3. [System Features 5](#_TOC_250005)
     1. AES Encryption and Decryption Design Feature 5
  4. [Nonfunctional Requirements 6](#_TOC_250004)
     1. Performance Requirements 6
     2. Safety Requirements 6
     3. Security Requirements 6
  5. [Scenarios 6](#_TOC_250003)
  6. [Report Structure 6](#_TOC_250002)

[Chapter 2 Literature Review 7](#_TOC_250001)

* 1. [Introduction 8](#_TOC_250000)
  2. Related Works 8
     1. Terminology 8
     2. Existing Implementations 8
     3. Categorization of Existing Techniques/Works/Research 8
     4. Languages 9
     5. Limitations/Gaps within Existing Techniques/Works 10
  3. Proposed Improvements in Existing Works 10
  4. Summary 11

Chapter 3 System Design 12

* 1. Introduction 13
     1. Purpose 13
     2. System Overview 13
     3. Design Map 13
  2. Design Considerations 14
     1. Assumptions 14
     2. Constraints 14
     3. Design Methodology 14
     4. Risks and Volatile Areas 14
  3. Architecture 14
     1. Oerview 15
     2. Components, Subsystems or Modules 1 …N 16
     3. Strategy 1…N 17
  4. Database Schema 18
     + 1. New Fields(s) 18
       2. Fields Change(s) 19

3.4.2 Data Migration 19

* 1. High Level Design 19
  2. Low Level Design 19
     1. Module 1…N 20

Module 1 20

SBOX 20

* 1. User Interface Design 20
     1. Screenshots 1… N 20
  2. Summary 20

Chapter 4 Implementation 22

4.1. Discussion 23

* 1. Implementation Tools and Technologies 24
  2. Summary 25

Chapter 5 Testing 26

* 1. Testing Techniques Employed for This Project 27
  2. Test Cases 27
  3. Test Results 27
  4. Summary 32

Chapter 6 Data Analysis and Results 33

|  |  |  |
| --- | --- | --- |
|  | 6.1. The Empirical Sudy Mehodology 34 | |
|  | 6.1.1. | Merics for Resul Comparison 34 |
|  | 6.1.2. | Merics for Resul Comparison 34 |
|  | 6.2. Satistical Evaluation 34 | |

  6.3. Summary 35

Chapter 7 Conclusions and Future Work 36

* 1. Contributions 37
     1. Contribution 1 37
     2. Contribution 2 37
  2. Findings 37
  3. Future Work.  37
     1. Improvements in the existing System 38
     2. Further System Designs 38

References 39

# List of Figures

*Figure 1.1: General Overview 2*

*Figure 1.2: High Level Synthesis 3*

*Figure 1.3:S-Box 4*

*Figure 2.1: High Level Synthesis Terminology 8*

*Figure 2.2: Starting screen of Vivado HLS to open or create a project 11*

*Figure 3.1: Design Map 14*

*Figure 3.2: AES Architecture 15*

*Figure 3.3: AES Encryption Description 16*

*Figure 3.4: Encryption Block 16*

*Figure 3.5: Decryption Block 17*

*Figure 3.6: High Level Design 19*

*Figure 3.7: Low Level Design 19*

*Figure 3.8: Visual Studio Interface 20*

*Figure 4.1: Vivado HLS Interface 24*

*Figure 4.2: ISIM Interface 25*

*Figure 5.1: Result of Mix Columns Block Wave Diagram 28*

*Figure 5.2: Result of S-Box Block Wave Diagram 29*

*Figure 5.3: Result of Shift Rows Block Wave Diagram 30*

*Figure 5.4: Result of Sub Bytes Block Wave Diagram 31*

# List of Tables

Table 3.1: Table of Fields 18

Table 3.2: Table of Variables 19

Table 6.1: Table of C-Based Code 34

Table 6.2: Table of Verlog-Based Code 34

## Chapter 1 Introduction

### Overall Description

It is a research-based project about implementation of Voice Encryption and decryption using AES Algorithm.

* + 1. Objectives

Project is about giving a speech signal as input to an Analog to Digital Converter (ADC) at the transmitter end which will covert voice into its digitized form and then voice samples will be stored it on FPGA Block rams. The voice will be decrypted at transmitter end. After sending it through a wired communication channel (Ethernet cable), it will be received on second FPGA the data will be decrypted and later convert from binary to speech signal through Digital to Analog Converter (DAC). We will work on two approaches namely

* + - * Offline Method (recorded Voice Samples Stored in memory)
      * Real-Time Method (Taking input from microphone and listening to output on Speaker)



ADC



FPGA

FPGA



DAC

*Figure 1.1: General Overview*

* + 1. Problem Description

Encryption is an efficient method for protection of speech communications. Voice Encrypt or digitize the conversation at transmitter end and apply a cryptographic technique to the resulting bit-stream. In order to decipher the speech correct encryption scheme must be used. Voice Encryption helps us in private and confidential manner. It is

nearly impossible to decrypt voice into its original form again. We will compare the efficiency and performance of standardized C/C++ implementations and open source

Verilog codes with ours and compare the results.

* + 1. Methodology

We will first write code for our algorithm in higher level language usually in c or C+

+.The C implementation will be synthesized into its equivalent Verilog implementation using higher level synthesis tool. Xilinx VIVADO HLS can convert into a higher language code into its parameterized Verilog modules. After that performance comparison will be done in terms of efficiency and latency.

2



Verilog Code



HLS



  C/C++

*Figure 1.2: High Level Synthesis*

* + 1. Product Scope

Privacy is everyone’s need nowadays. Through our project, we are devoted to make it

 better by figuring out about the performance and speed of AES Encryption Algorithm in Voice Encryption.

* + 1. User Classes and Characteristics

A public key and private key will be used. It’s a choice that whether public key or private key is being used on sender end and receiver end will use the alternate key. Private Key should not be compromised.

* + 1. Operating Environment

The project will operate in Visual studio, ISE design and VIVADO HLS, including the hardware platform of Spartan 6.

* + 1. Assumptions and Dependencies



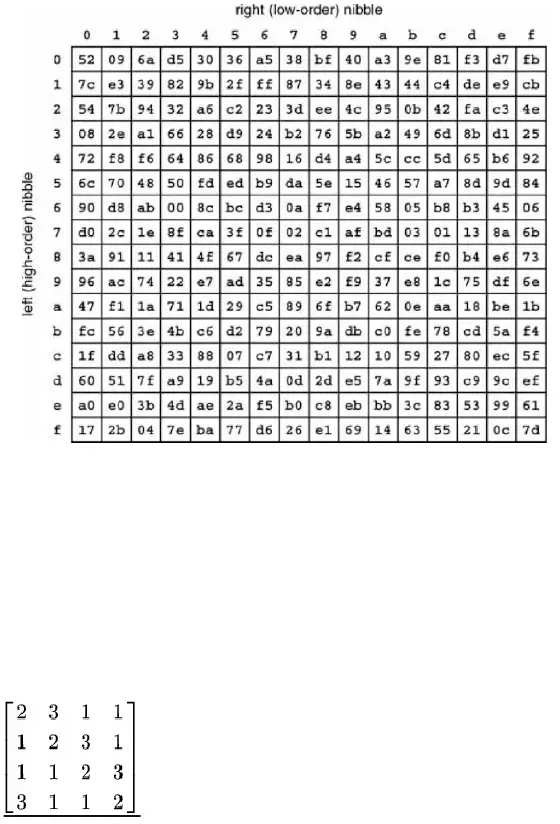
Main limitations in AES Algorithm  are that there are some input parameters like

generating a 10 round key. It makes it hard to determine key. Details are given below:

* Message is encrypted with cipher key using XOR.
* There are 4 transformations applied to the encrypted message in each round which are Sub Bytes, Shift Rows, Mix Columns and Add Round Key.
* In the final round Mix Columns is not applied.



For Sub Bytes, S-Box is used to provide confusion capability.



*Figure 1.4: S-Box*

* For Shift Rows, we rotate 1 byte in 2nd row, 2 bytes in 3rd row and 3bytes in 4th

row.

* In Mix Columns, each column obtained from Shift Rows is multiplied with Rijndael’s Galois Field.

 Add Round Key involves adding the obtained state from Mix Columns with the round key using a special type of arithmetic.

### External Interface Requirements

* + 1. User Interfaces

User interface will be a C/C++ console using Microsoft Visual Studio.

* + 1. Hardware Interfaces

Hardware interface that can be used for this project are minimum core i-3 laptop or computer for Microsoft Visual Studio and Xilinx VIVADO, a medium like Ethernet cable for IP and a SPARTAN 6 FPGA board for implementing it.

* + 1. Software Interfaces

Software interface that can be used for this project is Xilinx VIVADO. It will exponent e in high-level synthesis tool, VIVADO is capable of converts C Code to equivalent RTL (register transfer level) Verilog code.

* + 1. Communications Interfaces

Standard TCP/IP will be used for communication Interfaces. Communication standards can be used, such as FTP or HTTP.

### System Features

The main System feature is performing encryption and decryption. System will perform encryption on first block of text and so on.

* + 1. AES Encryption and Decryption Design Feature 1.4.1.1 Description and Priority

We takec iap h1e2r8 t-ebxi**t**. pPlraiionrtietyx tf oarn tdh e1 2fe8a-tbuirte kweiyl l fboerhpirgohc.essing in Decryption block to generate a

* + - 1. Stimulus/Response Sequences

By processing we generate response number and their waveforms.

* + - 1. Functional Requirements

It will ask for 128-bit number. By getting 128-bit number and key system will be able to create Private key to Cipher.

REQ-1: 128-bit plain text REQ-2: 128-bit key

### Nonfunctional Requirements

* + 1. Performance Requirements

AES Algorithm Matrix for Intel Dual-Core i7-4500U 1.80GHz is:



Bits

Time

128

192

256

260

Less t han 2 s ec

16 se c

35 m in

1 ho ur

* + 1. Safety Requirements

Main safety requirements include damage, harm or possible loss that could result from the use of this product is increasing temperature of device. Safeguards or actions that must be taken are to keep device on a proper temperature.

* + 1. Security Requirements

The security requirement that is needed is that Private Key is not compromised it must be a secret.

### Scenarios

There are three scenarios of AES system design 128 bit, 192 and 256-bit numbers. In this system design we can only use 128-bit number.

### Report Structure

Overall structure of this report includes description of AES, implemented procedures and techniques. In chapter 1 we define methodology assumptions, dependences, system features and functional, non-functional requirements. Chapter 2 contains terminology,

categorization of existing techniques, limitations and proposed improvements. Chapter 3 includes system design, Chapter 4 Implementation, Chapter 5 testing of implemented

techniques, Chapter 6 Date Analysis of implemented and existing techniques and overall conclusion of whole project and ideas on future work in Chapter 7.

## Chapter 2 Literature Review

### Introduction

The uttermost important aspect in our daily life is communication. The main issue in our communication is security to preserve its integrity, availability and proper access control as well as confidentiality. Therefore, Communication protection is essential from misuse. Voice encryption and Decryption is a research-based project about implementation of

Encryption and Decryption of voice signal over an IP network using AES algorithm. Encryption is efficient method that is used for security of communications.

### Related Works

* + 1. Terminology

Existing Voice encryption techniques are implemented in C/C++ and Verilog. We can use these existing techniques for our research and convert the existing C/C++ code in Verilog

 by synthesizing it by VIVADO then comparing it to existing Verilog and C/C++ techniques by speed testing, complexity and security.



  C/C++

  HLS



Verilog Code

*Figure 2.1: High Level Synthesis Terminology*

* + 1. Existing Implementations

As Rijndael has existed as the Advanced Encryption Standard since 2000, there are already many implementations of the algorithm. These implementations are available in many different programming languages. As the AES standard is open, organizations or users which wish to implement the Rijndael algorithm are free to do so.

* + 1. Categorization of Existing Techniques/Works/Research

Encryption techniques as far as hardware implementations are concerned can be broken down into following categories

a. Implementation in C/C++

 b. Implementation in Java

1. HDL based Implementations

8

1. GPU based implementation
   * 1. Languages

As previously stated, this project requires the implementation of the Rijndael’s algorithm

in systems constructed with several different programming languages. Therefore it is important to examine the history and nature of these languages, so as to understand how

they might be used and how cryptography might be used in a system developed in that language. The languages to be used in this project are Java, C, JavaScript and Perl.

* + - 1. Java

Java is an object-oriented language that was created and developed by software engineers at Sun Microsystems during the early 1990s. Code written in Java is compiled into byte code which can then be executed on the Java Virtual Machine. In

 practice this means that programs written in Java can be run on any computer which supports a JVM, regardless of the underlying hardware or operating system, which is a key feature of the language. While Java can be used to create desktop applications, it can also be used to create Java applets. These are applications which can be distributed over the internet and run within a web browser. Therefore a potential application for cryptography would be an applet which encrypts data on the client’s computer before submitting it to a server.

* + - 1. C

The programming language C was originally designed in 1972 and used as the systems language for UNIX. It gradually evolved in the years following until in 1990 the American National Standards Institute (ANSI) set the standard for the language, known as ANSI C. C is a relatively low level programming language that gives the

 programmer more control over aspects such as memory allocation. Like Java, C has

 been widely used and has a variety of potential applications, but for this language, a suitable program might be one that encrypts or decrypts files from a command

 prompt.

* + - 1. Perl

Perl is a scripting language like JavaScript, and grew because of its strengths in text manipulation. Perl is often regarded as a general purpose programming language which can be used to automate a variety of different tasks. While there is no ‘typical’ Perl system, one which used cryptography might do so for the encryption of data

 being passed between other systems.

* + - 1. JavaScript

JavaScript is a scripting language that was created in the early 1990s. It is commonly used on the internet to provide client-side functionality in web pages. It has no technical relation to Java, although both languages have a similar syntax. A system using JavaScript on the internet might make use of cryptography when displaying

encrypted content to a user.

* + 1. Limitations/Gaps within Existing Techniques/Works

The implementations are done in C, C++, Perl, Java and JavaScript is purely for simulation purposes and have no relevance to hardware in general. The category C and Java involve no usage of hardware but in GPU based implementation are still in

 beginning stage. So we are left with one option that is HDL based implementation. Current project will minimize the gap of C and HDL as main algorithm blocks will be done in higher level language will be ported to synthesizable HDL so that they can function as per requirement.

* Key Creation Complexity

AES algorithm is limited because of 10 rounds of encryption.

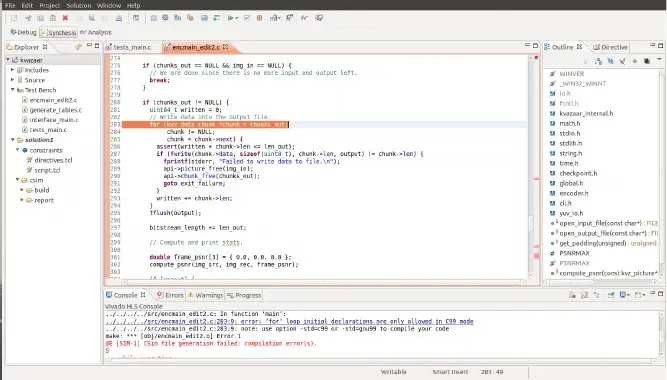
* + 1. Comparative Analysis of different AES implementations

Encryption has a strong presence in today’s digital electronics with the frequent transmission and storage of sensitive data. AES as the standard encryption algorithm and it is commonly used as a fast solution to secure data. When designing VLSI systems, the task of balancing the area, power, and speed is a challenge; hardware encryption is no different. System requirements drive certain performance parameters to the forefront, identifying how to alter design implementations to meet performance requirements is not always apparent. Multiple resources in this research field have identified AES algorithm features of interest and discussed their impact a few of the design trade spaces, however, a single comparative analysis was lacking. This project explores six different AES features key size, mode specificity, round key storage, round unraveling, SBOX implementation, and pipelining. A summarized view of the resulting designs allows readers to quickly analyze how each of the six features impacts speed, power, area, latency and throughput.

### Proposed Improvements in Existing Works

The implementation done in higher level language is just for simulation or observation and has no relevancy to hardware design constraints in general. So we will first take c blocks of the proposed encryption technology. VIVADO has the capability to convert the c language code into a synthesizable Verilog code. Figure 2.3 is the screenshot of a general VIVADO

HLS interface. In the image a C language will be converted into a register transfer level Verilog code.



*Figure 2.2: Starting screen of VIVADO HLS to open or create a project*

### Summary

Unwavering quality, openness, mystery and secretly of correspondence are the primary angles that would be kept up in voice security. Shielding voice frameworks from disturbance and adjustments just as the illicit access is the primary objective of voice security. A

 productive and secure correspondence framework for voice flags that will base on AES open key cryptosystem is structured in this work. The introduced cryptosystem is executed and its

execution is assessed utilizing diverse voice quality measurements in both encryption and unscrambling forms.

## Chapter 3 System Design

### Introduction

Cryptography assumes an essential job in security of information transmission. The improvement of processing innovation forces more grounded prerequisites on the cryptography plans. In 2000, the Advanced Encryption Standard (AES) supplanted the DES to beat the expanding prerequisites for security. In cryptography, the AES, likewise called as

Rijndael, is a square figure embraced as an encryption standard by the US government, which indicates an encryption calculation fit for securing delicate data. This calculation is a

symmetric square figure which can encode and decode the data. Encryption changes over information into an ambiguous structure known as figure content. Decoding of the figure content believers the information again into its unique structure, that is called plaintext. The AES calculation is under-pins keys length of 128, 192, and 256 bits to scramble and decode information in squares of 128 bits, consequently the name progresses toward becoming AES-128, AES-192 and AES256 separately. The equipment execution of the AES calculation can give superior, minimal effort for explicit applications and dependability contrasted with its product partners.

* + 1. Purpose

Out proposed project is about implementation of Voice Encryption and Decryption over IP Networks using AES Algorithm. Encryption is an efficient method for protection of speech communications. Voice Encrypt and digitize the conversation at transmitter end and apply a cryptographic technique to the resulting bit-stream. For decrypting the speech correct encryption scheme must be used. Voice Encryption helps us in private and confidential manner. It is nearly impossible to decrypt voice into its original form again.

* + 1. System Overview

By giving a speech signal as input to an Analog to Digital Converter (ADC) at the transmitter end which will covert voice into its digitized form and then voice samples will be stored it on FPGA Block rams. The voice will be decrypted at transmitter end. After sending it through a wired communication channel (Ethernet cable), it will be received on second FPGA the data will be decrypted and later convert from binary to speech signal through Digital to Analog Converter (DAC). We will work on two approaches namely

* + - * Offline Method (recorded Voice Samples Stored in memory)
      * Real-Time Method (Taking input from microphone and listening to output on Speaker)

-

13

* + 1. Design Map



AD4C.



FPGA

FPGA



DAC

*Figure 3.1: Design Map*

### Design Considerations

Design considerations like intended security and key’s expected time were handled.

* + 1. Assumptions

The main assumption of AES that the numerical problems are too large to be solved in a sensible time by attackers utilizing advance computer technology.

* + 1. Constraints

Main constraint of AES Encryption and Decryption is that if number of computations exceeds from RAM storage or RAM storage is low then resource will also proceed from FPGA limits.

* + 1. Design Methodology

Following Steps are followed for design methodology.

Step 1- Taking a C language code of AES encryption and decryption. Step 2- Simulation

Step 3- Converting C language code to Verilog code.

Step 4- Comparing existing codes to new generating code. Step 5- Performance and execution time checking.

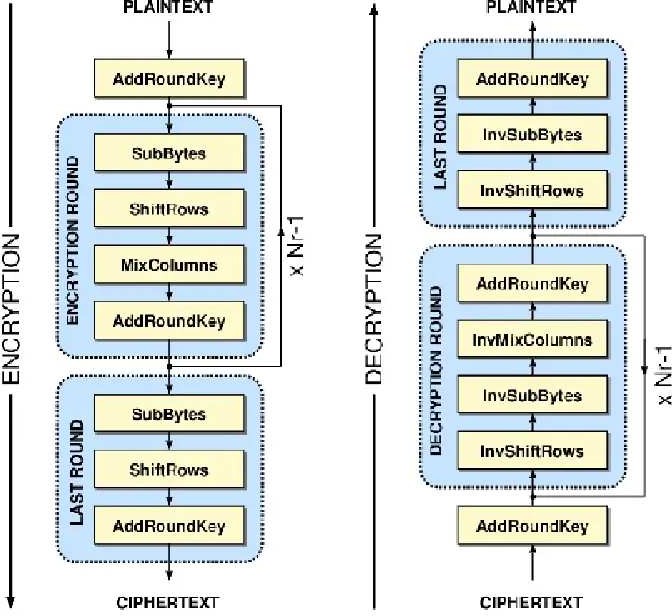
* + 1. Risks and Volatile Areas

 No risk and volatile areas are involved.

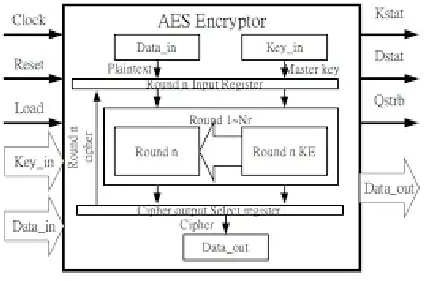
### Architecture

The architecture gives the high level design view of a system and gives a foundation for more specific design work.

* + 1. Oerview

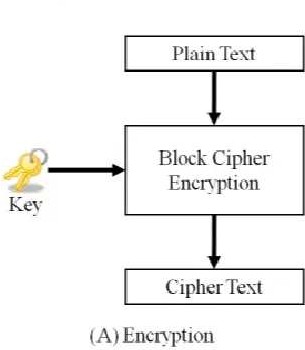


*Figure 3.2: AES Architecture*



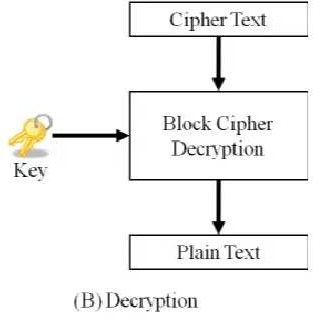
*Figure 3.3: AES Encryption Description*

* + 1. Components, Subsystems or Modules 1 …N

*Encryption Block*

*Figure 3.4: Encryption Block*

*Decryption Block*



*Figure 3.5: Decryption Block*

* + 1. Strategy 1…N

Followings are AES key generation algorithm steps:

* + - 1. From cipher key obtain the set of round keys. 2. Initialize the state array with the plaintext.

3. Add initial round key to the starting state array. 4. Perform nine rounds of state control.

1. Perform the tenth and final round of state manipulation.
2. Duplicate the final state array out as the encrypted data (cipher text).

Each round of the encryption procedure requires a series of steps to adjust the state array. These steps involve four operations:

* Sub Bytes
* Shift Rows
* Mix Columns
* XOR Round Key

AES Encryption algorithm:

1. Plain text M has been defined.
2. Cipher text C can be created using operations of state array.

AES Decryption algorithm:

1. Cipher text C has been received.



1. Plain text can obtain by performing nine full decryption rounds.
   * XOR Round Key
   * Inverse Mix Column
   * Inverse Shift Rows
   * Inverse Sub Bytes

Perform final XOR Round Key

The same round keys are used in the same order.

### AES Modes of Operation

A mode of operation describes how o repeaedly apply a cipher's single-block operation o securely ransform amouns of daa larger han a block. Five modes of operation of he AES algorihm were sandardized: ECB (Elecronic Code Book), CBC (Cipher Block Chaining), CFB (Cipher Feed Back), OFB (Oupu Feed Back) and CTR (Couner).

ECB (Electronic Code Book)

The ECB (Elecronic Code Book) mode of operation is he simples of all. A block scheme of

his mode is presened in Figure 3.6.

As i can be seen from Fig. 3.6, he plainex message is divided in blocks (P1, P2, PN), where each block is encryped separaely wih he same key (K). The resuls of he encryption are

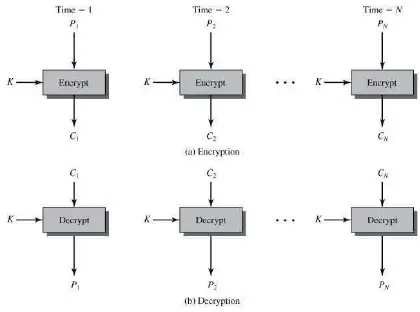
he encryped messages C1, C2 and CN respectively.

If he size of he message is larger han n blocks, he las block is lled wih padding. In his mode, if an error occurs in one of he blocks, i is no propagaed o he oher blocks, which is why decryption is possible in he blocks ha don’ conain an error.

The encryption in his mode is deerministic, because identical P blocks will produce identical C blocks, which is why identical plainex blocks or a message wih he same beginning are

easily recognizable. Also, he ordering of he C blocks can be changed wihou he receiver noticing. In general, his mode is no recommended for encryption of daa ha is larger han one block.

*Figure 3.6: Scheme of ECB*



CBC (Cipher Block Chaining)

In order o provide crypographic securiy, every encryption of he same plain ex should resul wih a dieren cipher ex. The CBC (Cipher Block Chaining) mode of operation (Fig. 3.7) provides his by using an initialization vecor labelled as IV. The IV has he same size as

he block ha is encryped.

Fig. 3.7 presens he encryption process. Firs, an XOR operation is applied o he plainex

block (P1) wih he IV, and hen an encryption wih he key (K) is performed. Then, he resuls of he encryption performed on each block (C1, C2, … , CN-1) is used in an XOR operation of

he nex plainex block PN which resuls in CN. In his way, when identical plainex blocks are encryped, a dieren resul is obained. Also, using a dieren IV for each new encryption, an identical message will always be encryped dierenly. I should be emphasized ha he same key K is used in each of he encryption blocks.

An error in one of he plainex block will propagae in all he following blocks and will be manifesed in he process of he description. Specications is recommended ha he Padding mehod 2 is used in case padding is needed wih he CBC mode of operation because i provides proection from some of he known PA (Padding Attacks).

There are complex CBC attacks for which an unpredicable value of IV is needed in order o overcome hem. In i is emphasized ha he CBC mode of operation is safe from CPA (Chosen

Plainex Attack) attacks (attacks in which he attacker chooses a se of plainexs and is able

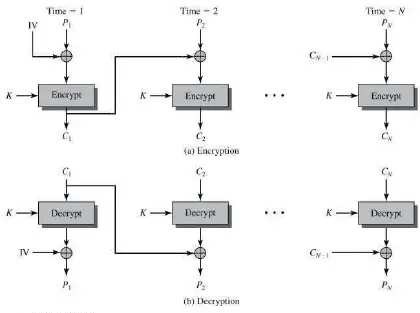
o obain respective cipher exs) only if he IV has a random value, bu no if he IV is a nonce (a number ha is no repeaed). The CBC mode of operation, besides is vulnerabiliy o PA attacks, is also easily susceptible o CCA (Chosen Cipher ex Attack) attacks (where he attacker chooses a se of cipher exs and is able o obain respective plainexs). According o [3], he encryption key has o be changed whenever condition holds:





In his equation, q is he number of blocks ha should be encryped and n is he number of bis in he encryption blocks.

In order o provide proection from CCA attacks in his mode of operation, i is necessary o use АЕ (Auhenticaed Encryption), where, besides he encryption, auhentication is also performed.



*Figure 3.7: Scheme of CBC*

CFB (Cipher Feed Back)

The CFB (Cipher Feedback) mode of operation allows he block encrypor be used as a sream cipher. The scheme of he CFB mode of operation is given in Fig. 3.8. As can be seen in Fig. 3, in he CFB mode of operation, a he beginning (a he rs block) he encryption (uses an encrypor denoed wih Encryp) is performed by using an IV and an encryption key K. Aer

ha, he XOR operation beween he encryption resul (he oupu form he encrypor) and

he plainex block (P1) is performed. For all he oher blocks, he encryption is performed over he resul of he encryption of he previous blocks accordingly (C1, C2, ). Then an XOR

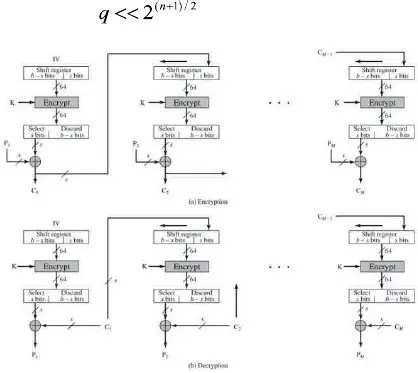
is performed wih he corresponding plainex block (P2, P3, ). In he beginning, he IV is

placed in a shi regiser, he size of which can be e.g. 64 bis. The resul of he encryption of

he IV is again 64 bis. Bu, he XOR is applied o only a few bis (for example, s=8) of he encryped IV wih also s bis from he plainex P1. The leas signican bis from he IV ha will no be used are discarded. The resul C1 from he XOR operation is hen placed a he righmos position in he shi regiser from he nex block, and he operation is repeaed in

he same manner. The encryption and decryption operations in he CFB mode of operation are he same operations. Also, an error in one block will propagae o he nex block, which is

manifesed in he process of decryption.

In i is poined ou ha he IV should be a unique identier, e.g. a couner, whereas in i is saed ha he value of he IV should be a nonce. The CFB mode of operation is safe form CPA attacks only if he IV has a random value, and is no safe if he IV is a nonce. Moreover, his mode of operation is no safe from CCA attacks. According o, he encryption key needs o be

changed each time condition holds.

*Figure 3.8: Scheme of CFB*

OFB (Output Feed Back)

The OFB (Oupu Feedback) mode of operation (Fig 3.9) also enables a block encrypor o be

used as a sream encrypor. As shown in Fig 3.9, he dierence beween he CFB and OFB

mode is such ha, in he case of an OFB, as an inpu for he shi regiser from he nex block,

he oupu from he encrypor (Encryp) from he previous block is chosen. A he same time,

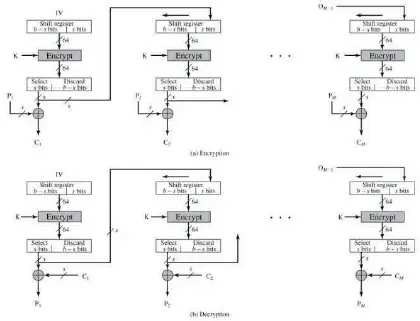
he XOR operation wih he s-bis of plain ex P uses only s bis from he encrypor. Encryption and decryption are he same operation. If here is an error in a block during he

encryption, while performing he decryption, i will inuence only a par of he plain ex ha will resul from ha block, i.e. here is a limied propagation of error. Therefore, his mode of operation is oen used in communication hrough media ha carry noise (for example, saellie communications).

According o, he IV should be a nonce. The guidelines given in sugges ha he IV should be

chosen randomly and used only once wih he given encryption key K. securiy does no exis if he IV is a nonce, bu he sequence generaed by some couner is accepable. The CFB

mode of operation is vulnerable o attacks performed by modication of bis in he encryped sream. To provide securiy in he OFB mode of operation, he encryption key K should be changed for every 2n/2 encryption blocks, where n is he number of bis in he block. Bu, OFB does no oer securiy from CCA attacks.



*Figure 3.9: Scheme of OFB*

CTR (Counter)

 A he CTR (Couner) mode of operation, shown on Fig. 3.10, as an inpu block o he encrypor (Encryp), i.e. as an IV, he value of a couner (Couner, Couner + 1, … , Couner + N 1) is used. The couner has he same size as he used block. As shown in Fig. 3.10, he XOR operation wih he block of plain ex (P1, P2, … , PN) is performed on he oupu block from

he encrypor. All encryption blocks use he same encryption key K. If he las block of clear

ex PN has he number of bis smaller han he number of bis in he block, hen only he s mos signican bis for he XOR operation on he block PN are used from he oupu block of

he encrypor. The remaining bis are discarded. Hence, as i is poined ou in [4], here is no

need for adding bis (padding) o he las block. Values of he couners are independen from

he oupu of he previous block; herefore, here is no propagation of error from one block

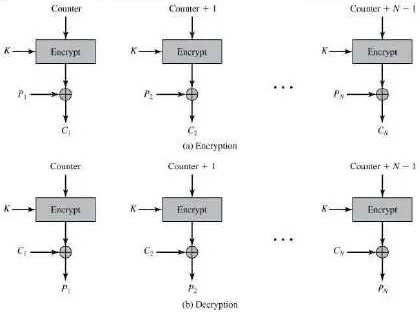
o anoher. Considering he independence of he blocks, his fac allows for parallelism in he encryption and he decryption, and here is also he possibiliy of preprocessing he values of

he encrypors, which speeds up he process.

The encryption and decryption operations a he CTR mode of operation are he same. The couner sequence should be dieren for every block. On he oher hand, i is suggesed ha

he same value of he couner (Couner, Couner + 1, Couner + N - 1) and he same key K should no be used in he encryption of more han one block of daa. If his requiremen is

no upheld, he plainex can be revealed by performing he XOR operation on he wo blocks of ex encryped by he same se of parameers. In ha case, here is a complee breach of privacy. Usually he couner is initialized o some value, and hen i is incremened by one for every block. Couner is a non-repeaable number on he order of 96 bis. The oher 32 bis are zero a he beginning of he process, and hen heir values are incremened by 1 for every block. The guidelines found in recommend using a unique value for he Couner, chosen in a random manner. In order o ensure securiy wih he CTR, he encryption key K should be changed for every 2n/2 blocks of encryption where n is he number of bis in a block [3]. In summation on he modes of operation in, he CTR mode is marked as he bes choice among all he ohers.



*Figure 3.10: Scheme of CTR*

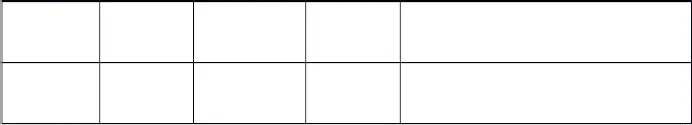
### Database Schema

* + - 1. New Fields(s)



Table Name Field Name Data Type

Allow Nulls Field Description



Plain text

Plaintext

128 bit number yes

128 bit number that will be given to

encryption block

Key

Private k ey

128 b it n umber yes

128 bit private number that will be given to

cipher decryption block/

*Table 3.1: Table of Fields*

* + - 1. Fields Change(s)

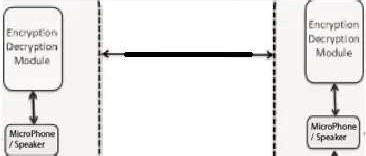
|  |  |  |
| --- | --- | --- |
| Table Name | Field Name | What they do? |
| Sbox value | Sub\_bytes | The changing input sequence after comparing it with the sbox 2 D |
| Shift Rows | ShiftRows | matrix  Shift 11 b yte i n 2nd row, 2 bytes in 3rd row and 3 bytes in 4th row |
| Mix Colums | encMixCols | Each column obtained from Shift Rows is multiplied with Rijndael’s |
|  |  | Galois Field |
| Round Key | AddRoundKey | Obtained state from Mix Columns with the round key using a special |
|  |  | type of arithmetic. |

*Table.3.2: Table of Variables*

3.4.2 Data Migration

Existing data is migrating in new tables by analyzing of sub bytes and shift rows.

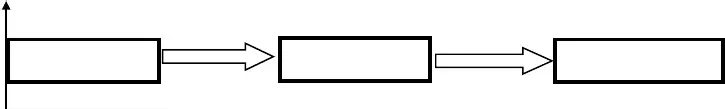
### High Level Design



*Figure 3.6: High Level Design*

* 1. Low Level Design

*Encryption*



Voice Signal

  ADC

Cipher Tex



Public Key

*Figure 3.7: Low Level Design*

At encryption block, a speech /voice signal is given as input to an Analog to Digital convertor (ADC) which will converted into binary and then voice samples will be stored it on FPGA Block RAMS then encrypted with public key. Then the cipher text it is sent through a medium.

* + 1. Module 1…N

### Module 1

SBOX

Main purpose of SBOX is 128 bit numbers is applied to each block by separated to 1

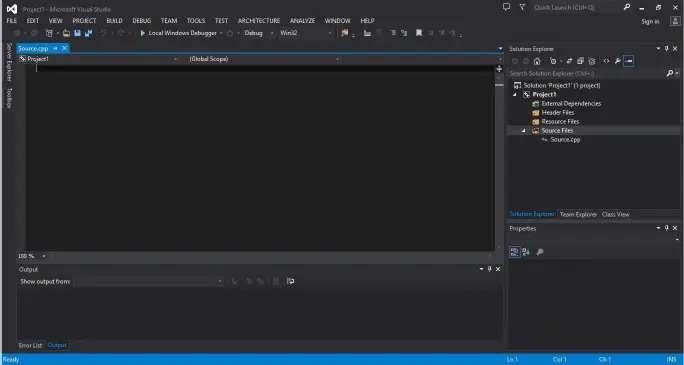
 byte. The range of S-BOX is from 00-FF. Each byte will be computed by 2D table.

### User Interface Design

At the GUI the process is generating the User Public key.

* + 1. Screenshots 1… N

Major user interface screens the user will experience will be a C/C++ console using Microsoft Visual Studio.



*Figure 3.8: Visual Studio Interface*

### Summary

This structure accomplishes tradeoff between the speed and zone even without utilizing the BRAM. The information way is kept up to be 128 bits. The entire structure is performed with the assistance of Xilinx ISE and VIVADO HLS orchestrated with its apparatuses. The recreation and combination is finished focusing on the Spartan 6 FPGA. From the got exhibitions, we can reason that our proposed AES Architecture is appropriate to be utilized in asset obliged frameworks.

## Chapter 4 Implementation

### Discussion

Most of the challenging situations we faced during the implementation required finding the

 proper stability between flexibility, effectiveness and usability, while not compromising safety or security.

### Development Methodologies

Existing Voice AES encryption techniques are implemented in C/C++ and Verilog. We can use these existing techniques for our research and to convert the existing techniques C/C++ code in Verilog then verification and comparison to existing Verilog and C/C++ techniques

 by speed testing, complexity and security.

Modules in C++ Code:

aes\_ctr128\_axis:

The module for the complete AES-128 CTR mode encryptor / decryptor, including key expansion. In this module has a 128-bit input data, 128-bit key and 256-bit of initialization vector. This module will XOR ECB (Electronic Code Book) output with

 plaintext for encryption and with cipher text for decryption.

aes\_ecb128:

It is a complete AES-128 ECB mode encryptor, including key expansion. There are 10 rounds required for AES-128. It performs process of sub Bytes, shift Rows, mix Columns and add round key.

encMixCol:

Every input and output of different stages in this module are of 8 bits. In round 1 the four

numbers of one column are modulo multiplied in Rijndael’s Galois filed by a given matrix. The MixColumns step along with the ShiftRows steps is the primary source of diffusion in Rijndael.

keyExpEngine:

Key expansion range in this module is 32 bits.  This module calculates the next round key for AES key expansion.

nonceCount:

 Nonce and counter for AES counter mode. Initialization Vector will be loaded at power-

on/reset or at first 16byte word (message block) of new plaintext/cipher text message. The last word of current plaintext/cipher text message is indicated by "last".

At the first word of a new message, the counter and nonce will load from init\_vect. This allows the counter to be started at a non-zero value chosen by the user. The sizes (in bits) of the counter and nonce variables can be modified but the sum of their lengths must be 128.

rCon:

RCON lookup table for AES key expansion. Expansion of the given Cipher key into 11

 partial keys, used in initial round, the 9 main rounds and the final round. The expanded key can be seen as an array of 32-bit columns numbers from 0 to 43. The first four columns are filled with the given Cipher key.

rotWord:

Word rotate function for AES key expansion.

shiftRows:

Input and output range in this module is 8 bits. It rotates over 1st byte, 2nd byte and 3rd byte.

subBytes:

In the Sub Bytes step, each byte in the matrix is updated using an 8-bit substitution box, the Rijndael S-box. This module provides the non-linearity in the cipher. The S-box used

is derived from the multiplicative inverse over GF (28), known to have good non-linearity

 properties. To avoid attacks based on simple algebraic properties, the S-box is constructed

 by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points, and also any opposite fixed points.

subWord:

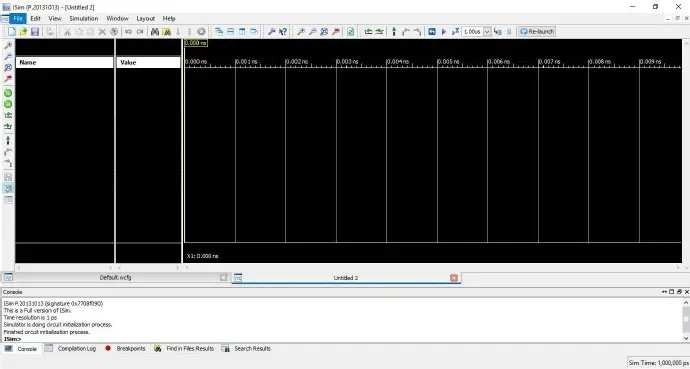
SBOX function for key expansion.

### Implementation Tools and Technologies

Microsoft Visual Studio was used to analyze the existing techniques; The C implementation will be synthesized into its equivalent Verilog implementation using higher level synthesis tool. VIVADO HLS can convert into a higher language code into its parameterized Verilog modules. A Xilinx ISE test tool was used for testing, verification and comparison of new

generated technique with existing techniques.

*Figure 4.1: Vivado HLS Interface*



*Figure 4.2: ISIM Interface*

* 1. Summary

In this project we have implemented AES technique in C/C++ and C implementation will

 be synthesized into its equivalent Verilog implementation using higher level synthesis tool. We use existing techniques forverification and comparison to implemented Verilog and C/C++ techniques by speed testing, complexity and security.

## Chapter 5 Testing

### Testing Techniques Employed for This Project

Major testing technique employed for this project is ISIM. Xilinx ISIM is a Hardware Description Language (HDL) simulator that permits to perform purposeful and timing simulations for VHDL, Verilog and mixed VHDL/Verilog designs.

### Test Cases

There must be a defined methodology to ensure an implementation meets the published standard or not. In simple words, one should be able to validate that given a plaintext string and a specific key, the output of the encryption algorithm is the correct one.

Such a methodology does exist. So, in this project testing cases includes Input values and key management. The data we gave is “d4e0b81e27bfb44111985d52aef1e530” in Hexadecimal and the program returned correct value.

### Test Results

The test results that are obtained by testing elements (Input values and key) using Xilinx ISIM testing techniques is correct input/output values. Test results from test cases are given

 below:

In the Mix Columns step, the four bytes of every section of the state are joined utilizing an invertible direct change. The Mix Columns work accepts four bytes as information and four

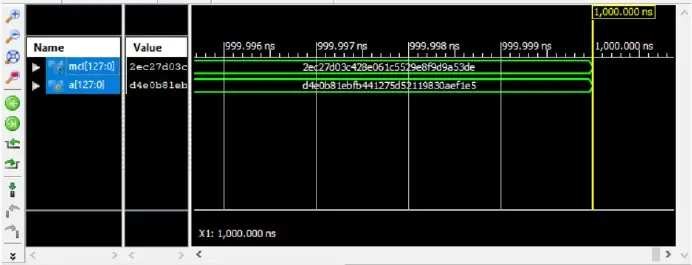
 bytes for output, where each info byte influences every one of the four output bytes. Together with Shift Rows, Mix Columns gives assemble in the cipher.

During this activity, every section is multiplied by the known matrix that is for the 128-bit key is

The multiplication activity is characterized as: multiplication by 1 implies no change, multiplication by 2 implies moving to the left side, and multiplication by 3 implies moving to the left side and afterward performing XOR with the underlying un-shifted values. After moving, a conditional XOR with 0x1B will be performed if the shifted value is bigger than 0xFF.

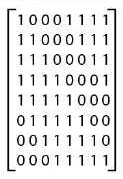
33

Every section is treated as a polynomial over Galois Field (2^8) and is then multiplied modulo x4+1 with a fixed polynomial c(x) = 0x03 · x3 + x2 + x + 0x02. The coefficients are shown in their hexadecimal equivalent of the binary representation of bit polynomials from Galois Field (2)[x].



*Figure 5.1: Result of Mix Columns Block Wave Diagram*

The S-Box is produced by deciding the inverse multiplication of a given number in Rijndael's Galois Field. The inverse multiplication is then changed using the following matrix know as affine transformation matrix:



This relative change can likewise be determined by the accompanying calculation:

* + 1. Store the inverse multiplication of the info number in two 8-bit unsigned impermanent factors: s and x
    2. Rotate the one-bit s value to the left side; if the estimation of s had a high piece (eight

 bit from the left) of one, make the lower bit of s one; generally, the low bit of s  is zero.

* + 1. XOR the estimation of x with the estimation of s, storing estimated value in x

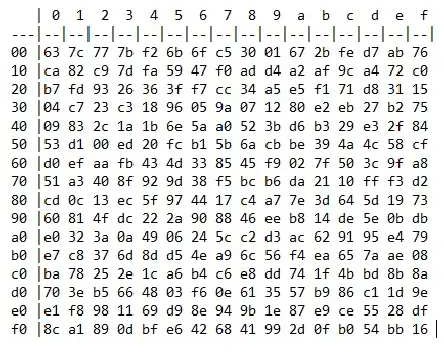
34

* + 1. For three additional cycles, repeated stages two and three; stages two and three are completed a sum of multiple times.
    2. The estimation of ''x'' will currently have the changed value.

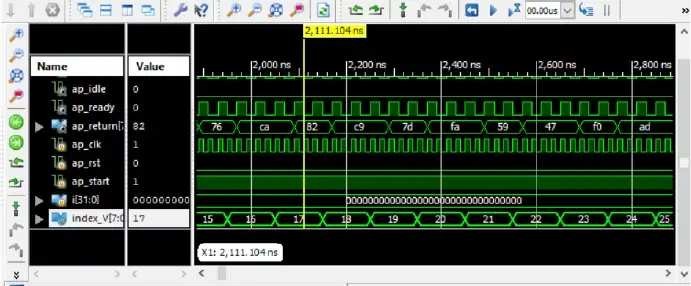
After the relative change is done, XOR the received value with the decimal number 99 (the hexadecimal number).

Here is the code that performs above algorithms.

This code will generate the AES S-box, which is represented below with hexadecimal form.



35

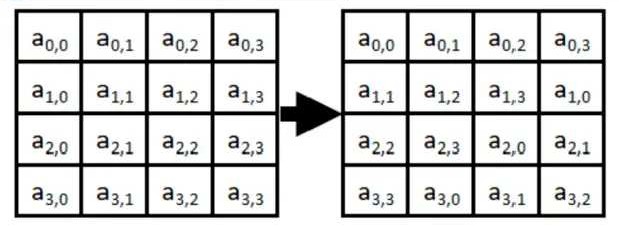


#### *Figure 5.2: Result of S-Box Block Wave Diagram*

In the Shift Rows period of AES, each column of the 128-bit interior state of the cipher is moved. The column in this stage refers to the standard representation of the inward state

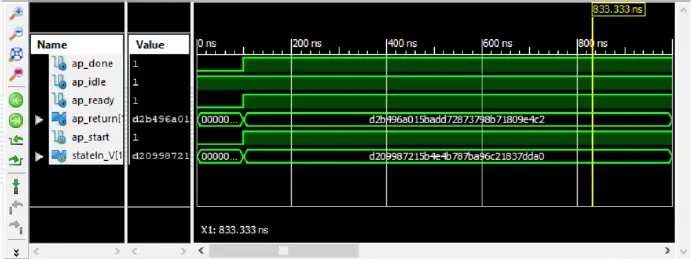
in AES, which is a 4x4 matrix where every cell contains a byte. Bytes of the inner state are set in the matrix row wise over lines from left to right and down columns.

In the Shift Rows activity, every one of these lines is moved to left side by a set sum: their column number beginning with zero. The top column isn't moved in any way, the following line is moved by one, etc. This is outlined in the Figure underneath.



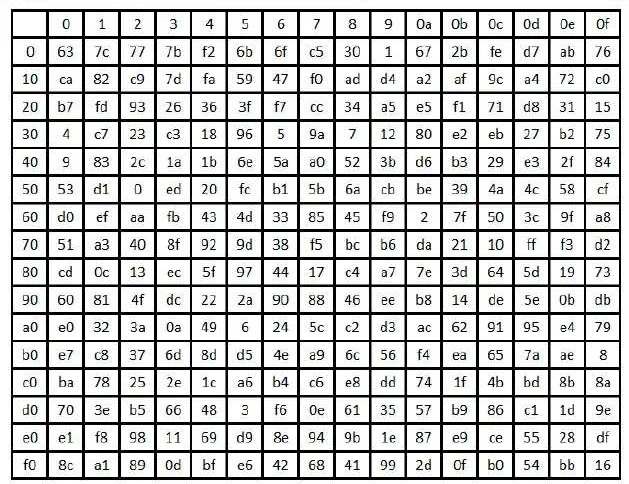
In the Figure, the main number in every cell alludes to the line number and the second alludes to the segment. The highest line (row 0) does not move by any means, push 1 moves left by one, and up to so on.

36



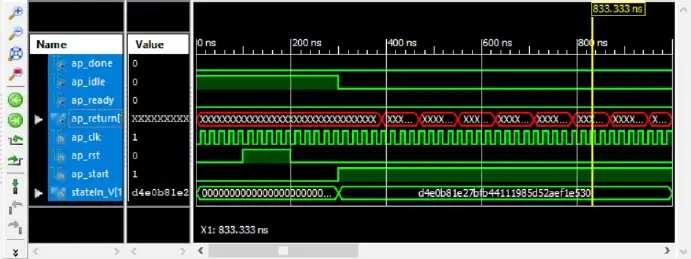
#### *Figure 5.3: Result of Shift Rows Block Wave Diagram*

In Sub Bytes period of AES includes the contribution to bytes and going each through a Substitution Box or S-Box. In contrast to DES, AES utilizes a similar S-Box for all bytes. The AES S-Box actualizes reverse augmentation in Galois Field 2^8. The AES S-Box is

appeared in the Table underneath.

37

To read above Table, the byte input is divided into two 4-bit parts. The first half decides the row and the second half decides the column. For instance, the S-Box change of 35 or 0x23 can be found in the cell at the crossing point of the row named 20 and the column named 03. In this manner decimal 35 ends up 0x26.



*Figure 5.4: Result of Sub Bytes Block Wave Diagram*

### Summary

The summary of this chapter tells us about test results in ISIM by testing input of 128

 bits’ plain text, key and their wave diagrams.

38

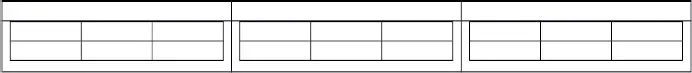
## Chapter 6 Data Analysis and Results

### The Empirical Study Methodology

We implemented AES algorithm in C language, and generating C code into Verilog using VIVADO HLS and testing and comparing it by existing implemented codes.

### Metrics for Result Comparison

We take 128-bit in both generated and existing code. By evaluating resulting we analyze and compare their values are equal or not.

Proposed Verilog implementation

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | LUT |  |  | FF |  |  | IOB’s |  |
| Used | Avail | % | Used | Avail | % | Used | Avail | % |
| 2001 | 53200 | 3% | 1604 | 3183 | 50% | 520 | 200 | 260% |

*Table 6.1: Table of C-Based Code*

Reference AES Design module

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LUT | | FF | | | IOB’s | | | |
| Used | Avail | % | Used | Avail | % | Used | Avail | % |
| 9387 | 53200 | 17% | 0 | 9387 | 0% | 384 | 200 | 192% |

*Table 6.2: Table of Verilog-Based Code*

### Performance Evaluation

By comparing and testing the values of our generated and existing code we can say that our hardware is efficient co it uses on 3% of hardware. Existing codes block circuits are combinational and our generated code blocks circuits are sequential because they’re using Flip Flop.

### Statistical Evaluation

By analyzing and comparing lookup tables of existing and generated techniques we can find that existing technique uses 17% of lookup table and generated technique uses 3% of lookup tables.

40

### Summary

The summary of the chapter tells us about the testing techniques that are being used during the completion of project. From the scratch till the end of the project from the existing works to the comparison with the results generated from newly work-done. Test cases basically tells us about the Lookup table, flip-flops and IOB’s that are being used in the project their

 previous results and new results are being compared in order to differentiate.

41

## Chapter 7 Conclusions and Future Work

### Contributions

* + 1. Contribution 1

Our first contribution in this project was to find the issues of conversion of floating point to fix point implementation. Usually the code in C is processors based which is usually used by the floating-point.

On the other hand, FPGA’s have fixed point implementation. If they are converted into floating point, then they consume a lot of resources. This phenomenon was observed in the project. We used ZYNQ-702 architecture as it supports floating point numbers.

* + 1. Contribution 2

Analysis of clock without predefined pipelines had an effect in the latency. The

 performance of the circuit was checked with VIVADO HLS directives which were responsible for the creation of pipelines.

### Findings

Overall project was a good learning curve-one of the key findings of the project was the VIVADO capability to optimize the hardware. VIVADO takes an array and process it in the form of shift registers. Furthermore, the synthesis tool employed usage for dedicated DSP units. DSP unit have multiple accumulators which VIVADO mapped them not so efficiently as we would have liked.

We observed in our finding was much better than existing techniques in manner of hardware utilization and time complexity.

### Future Work

The AES algorithm is most widely used algorithm for different security based applications. Security of the AES algorithm can be expanded by utilizing biometric framework for creating key.

As a future work, we are going to continue this research in order generating more secure key

 by biometric implementation and to get the maximum encryption speed in limited implementation area. Moreover, some artificial intelligence techniques can be used for simulating the AES encryption and crypto analysis processes.

43

* + 1. Improvements in the existing System

Existing system used 128-bit scheme. It can be further improved by increases 128 bits to 192 and 256-bit scheme. System performance can be further increased by applying pipe lines stages in between modules.

* + 1. Further System Designs

We need to have an improved mechanism for floating point to fixed point implementation. We haven’t implemented any parallel processing. For parallel processing we can use the concept of loop unrolling. Loop unrolling identifies independent sections in for loop. By using this concept, we can further increase the performance of the system.

44

## References

1. Abdullah, A. M., & Aziz, R. H. H. (2016, June). New Approaches o Encryp and Decryp Daa in Image using Crypography and Seganography Algorihm., Inernational Journal of Compuer Applications, Vol. 143, No.4 (pp. 11-17).
2. Singh, G. (2013). A sudy of encryption algorihms (RSA, DES, 3DES and AES) for information securiy. Inernational Journal of Compuer Applications, 67(19).
3. Gaj, K., & Chodowiec, P. (2001, April). Fas implemenation and fair comparison of he nal candidaes for Advanced Encryption Sandard using Field Programmable Gae Arrays. In Crypographers’ Track a he RSA Conference (pp. 84-99). Springer Berlin Heidelberg.
4. Sallings, W. (2006). Crypography and nework securiy: principles and practices. Pearson Education India.
5. Yenuguvanilanka, J., & Elkeelany, O. (2008, April). Performance evaluation of hardware models of Advanced Encryption Sandard (AES) algorihm. In Souheascon, 2008. IEEE (pp. 222-225).
6. Lu, C. C., & Tseng, S. Y. (2002). Inegraed design of AES (Advanced Encryption Sandard) encryper and decryper. In Application-Specic Sysems, Archiecures and Processors, 2002. Proceedings. The IEEE Inernational Conference on (pp. 277-285).
7. Mohamed, A. A., & Madian, A. H. (2010, December). A Modied Rijndael Algorihm and i's Implemenation using FPGA. In Elecronics, Circuis, and Sysems (ICECS), 2010 17h IEEE Inernational Conference on (pp. 335-338).
8. Pramsaller, N., Gurkaynak, F. K., Haene, S., Kaeslin, H., Felber, N., & Fichner, W. (2004, Sepember). Towards an AES crypo-chip resisan o dierential power analysis. In Solid-Sae Circuis Conference, 2004. ESSCIRC 2004. Proceeding of he 30h European IEEE (pp. 307-310).
9. Deshpande, H. S., Karande, K. J., & Mulani, A. O. (2014, April). Ecien implemenation of AES algorihm on FPGA. In Communications and Signal Processing (ICCSP), 2014 IEEE Inernational Conference on (pp. 1895-1899).
10. Nadeem, H (2006). A performance comparison of daa encryption algorihms," IEEE Information and Communication Technologies, (pp. 84-89).
11. Diaa, S., E, Haem M. A. K., & Mohiy M. H. (2010, May) Evaluating he Performance of Symmeric Encryption Algorihms. Inernational Journal of Nework Securiy, Vol.10, No.3, (pp.213-219).

45

1. Jain, R., Jejurkar, R., Chopade, S., Vaidya, S., & Sanap, M. (2014). AES Algorihm Using 512 Bi Key Implemenation for Secure Communication. Inernational journal of innovative Research in Compuer and Communication Engineering, 2(3).
2. Selmane, N., Guilley, S., & Danger, J. L. (2008, May). Practical seup time violation attacks on AES. In Dependable Computing Conference, 2008. EDCC 2008. Sevenh European (pp. 91-96).

IEEE.

1. Beren, A. (2013). Advanced Encryption Sandard by Example. Documen available a URL [http://www.](http://www/) neworkdls. com/Articles/AESbyExample. pdf (April 1 2007) Accessed: June.
2. Benvenuo, C. J. (2012). Galois eld in crypography. Universiy of Washingon.
3. Lee, H., Lee, K., & Shin, Y. (2009). Aes implemenation and performance evaluation on 8-bi microconrollers. arXiv preprin arXiv:0911.0482.
4. Padae, R., & Pael, A. (2014). Encryption and decryption of ex using AES algorihm. Inernational Journal of Emerging Technology and Advanced Engineering, 4(5), 54-9.
5. Reddy, M. S., & Babu, Y. A. (2013). Evaluation of Microblaze and Implemenation of AES Algorihm using Sparan-3E. Inernational Journal of Advanced Research in Elecrical,

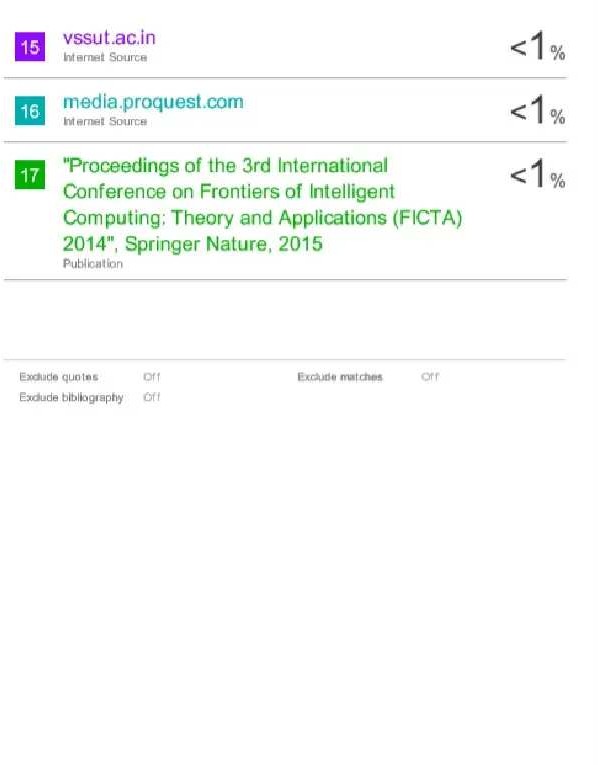
Elecronics and Insrumenation Engineering, 2(7), 3341-3347.

46





47



48