Analysis and Implementation of DES Using FPGA

Thesis submitted in the partial fulfillment of requirement for the award of degree of

Master of Technology

in

VLSI Design & CAD

Submitted by:

Gaurav Sharma

Roll No. : 601061010

Under the guidance of:

Mr. Ajay Kakkar

Assistant Professor

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA – 147004 (PUNJAB)

June 2012

DECLARATION

I, Gaurav Sharma, hereby certify that the work which is being presented in this thesis entitled "Analysis and Implementation of DES Using FPGA" by me in partial fulfillment of the requirements for the award of degree of Master of Technology in VLSI Design & CAD from Thapar University (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Mr. Ajay Kakkar.

The matter presented in this thesis has not been submitted in any other University / Institute for the award of any other degree.

Date: 11/07/12

Gaurav Sharma
Roll No. 601061010

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Date: 11/07/12

Mr. Ajay Kakkar
Assistant Professor
ECED

Countersigned by:

(Dr. Rajesh Khanna)
Professor and Head ECED

Thapar University, Patiala
Date:

(Dr. S.K. Mohapatra)
Dean of Academic Affairs

Thapar University, Patiala
Date:

Acknowledgement

First of all, I would like to express my gratitude to **Mr. Ajay Kakkar, Assistant Professor**, Electronics and Communication Engineering Department, Thapar University, Patiala for his patient guidance and support throughout my work. I am truly very fortunate to have the opportunity to work with him. I found his guidance to be extremely valuable.

I am also thankful to our **Head of the Department, Dr. Rajesh Khanna,** Electronics and Communication Engineering Department, entire faculty and staff of Electronics and Communication Engineering Department. I would also like to thank my friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would like to thank my parents for their unconditional support and encouragement.

Gaurav Sharma

601061010

Abstract

Secure communication is the prime requirement of every organization. In today's world the security has became the major aspect of life. It can be achieved by various techniques such as password, cryptography and biometrics. A Field-Programmable Gate Array (FPGA) is a semiconductor device containing programmable logic components called logic blocks, and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or mathematical functions. FPGAs can be used for specific operational behavior, or general purpose CPU functionality depending on the complexity of the device. FPGA applications include DSP applications, imaging, speech recognition, data security, hardware emulation and for many other applications. In the work, the data has been initially converted into their equivalent numbers and are converted to their equivalent binary numbers in order to provide more secured data. Initially, we apply the test vectors to our design through the test bench and then compile, simulate the test bench with the help of model sim software. When the simulation has been completely done then we analyze the result with help of waveform generated by the Model-Sim software. The thesis deals with various parameters such as variable key length, key generation mechanism, etc. used in order to provide optimized results.

The classical DES takes 19 nanoseconds of encryption time for an input having data size of 64 bits. The algorithms has been implemented using VHDL, using their standard specifications, and has been implemented on Xilinx Spartan – 3E (XC3S500) FPGA kit. It has been observed while for input data size of 64 bits encryption time of 16 nanoseconds is achieved. There has been improvement of 15.78% in encryption time as compared to the classical DES.

Table of contents

List of Figures

List of Tables

List of Abbreviations

DES	Data Encryption Standard
TDES	Triple Data Encryption Standard
IDEA	International Data Encryption Algorithm
DSA	Digital Signature Algorithm

PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PKCS	Public Key Cryptographic Standards
ECB	Electronic Codebook
CBC	Cipher Block Chaining
CA	Certificate Authority
AES	Advance Encryption Standard
VHDL	Hardware Description Languages
FPGA	Field Programmable Gate Array
UD	uni-city distance
NLFSR	Non Linear Feedback Shift Register
IP	Initial Permutation
API	Application Programming Interface
COPACOBANA	Cost-Optimization Parallel Code Breaker
DPA	Differential Power Analysis

Chapter 1 Introduction

This chapter includes the introduction of various cryptography algorithms and approaches used for data security. Their design styles and applications have also been included in this chapter.

1. Cryptography

It is a technique used to avoid an unauthorized access of data. It helps to provide accountability fairness and accuracy and also provide confidentiality. Broadly, four different kinds of people contributed their efforts in this technique and are: (i) Military, (ii) The Diplomatic Corps, (iii) Diarists, and (iv) Communications System. Cryptography involves two basic operations and is named as encryption and key management. Information/data can be encrypted using a cryptographic algorithm by various keys. The security of cryptographic system is not only dependent on the encryption algorithm; it also depends upon the keys used for the encryption. These keys area always kept secured from the hacker and are known as secret key. Key plays an important role in encryption process, which is a main part of cryptography [31]. Due to channel impairments sometimes the transmitted data and/or key may get corrupted. If it is slightly changed or corrupted, the data will not be recovered; therefore, key needs to be transported over the secured channel. The frequency of use of a cryptographic key always has a direct correlation to how often the key should be changed. Encryption algorithms can be hacked by utilizing supercomputer which provides fast speed and allows the hacker to use more permutations and combinations in a specified time. Modern era depends upon wireless communication and almost all the electronic funds are being carried out online. In order to protect the same and maintain the privacy of the users; cryptography is the best solution due to their better response even in the presence of advisory. For better security, either more number of keys is used or the length of the existing keys is increased. In both the approaches the overheads are increased, therefore, the best idea is to use sub-keys. Sub-keys are used only for such nodes which are attacked by the hacker. The sub keys are always derived from the main key which helps in reducing the overheads. The basic cryptographic model has been shown in figure 1.1. It involves encryption and decryption section; initially, the data has been encrypted by the help of key and further transmitted over the web. Finally, it has been received and the encrypted data is decrypted with the help of same or different key. The key is any value and/or word and is used in both the sections for encryption and decryption purpose.

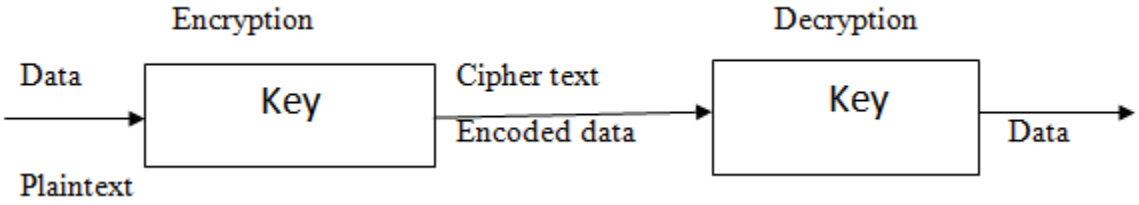


Figure 1.1: Block diagram of Cryptographic Model

There are two classes of key-based encryption algorithms: symmetric and asymmetric algorithms. Symmetric algorithms use the same key for encryption and decryption, whereas asymmetric algorithms use different keys for encryption and decryption.

2. Need of Cryptography

As almost all the organizations such as banks, railway, military, telecommunication, etc depends upon wireless approaches and are open to all the computer and the networks (LAN, MAN and WAN). Their transfer of funds, information and data all are carried out online. Secured funding and E-mails are the major requirement of all the above said organizations; therefore, it is highly essential to protect the data from the intruders. Electronic data transfer is used in all the present applications and it includes the security of ATM cards, computer passwords, and electronic commerce. Passwords are not good so far for the task [32] due to their short range; therefore, cryptography has wide future because this technique can be able to with stand against the various attacks.

3. Motivation

The need for encryption arises for the protection of private information. There are many cipher algorithms, which are used for this purpose such as DES, RC5, TDES, Blowfish, Two fish and IDEA. At present the speed of software used to break the system are very fast so there must be a need to develop a system in which key length is large to provide security, [31] but in doing so it does not result in error. For this task DES is used in which different keys are used in order to provide more security.

Key terms used in Cryptography

- **Encryption Algorithm** is a technique to convert the plain text into the cipher text with the help of symmetric and/or asymmetric keys. Ciphertext is a form which cannot be easily understood by unauthorized people.
- **Cryptosystem** is hardware or software implementation of cryptography is that transforms a message to ciphertext and back to plaintext. A cryptosystem consists of

three algorithms: one for key generation, one for encryption, [33] and one for decryption. The term cipher (sometimes cypher) is often used to refer to a pair of algorithms, one for encryption and one for decryption. Cryptosystem is most often used when the key generation algorithm is important.

- **Cryptanalysis** is a practice of obtaining plaintext from ciphertext without a key or breaking the encryption. Cryptanalysis refers to the study of ciphers, ciphertext, or cryptosystems with a view to finding weaknesses in them that will permit retrieval of the plaintext from the ciphertext, without necessarily knowing the key or the algorithm. This is known as breaking [31] the cipher, ciphertext, or cryptosystem.
- **Ciphertext** is the data in encrypted or unreadable format. Plaintext is what you have before encryption, and ciphertext is the encrypted result. The term cipher is sometimes used as a synonym for ciphertext, but it more properly means the method of encryption rather than the result. Cryptology is the study of both cryptography and cryptanalysis.
- **Encipher** is the act of transforming data into an unreadable format. **Decipher** is the act of transforming data into a readable format in such a way that the analysis of documents written in ancient languages, where the language is unknown, or knowledge of the language has been lost. Key Secret is a sequence of bits and instructions that governs the act of encryption and decryption. The keys need to be protected as they are being transmitted and while they are being stored on each workstation and server. The keys [33] need to be generated, destroyed, and recovered properly. Key management can be handled through manual or automatic processes.

The salient features of cryptography are confidentiality and authentication. Confidentiality is the protection against unauthorized disclosure of information. Confidentiality may be applied to whole messages, parts of messages, and even existence of messages. Confidentiality is the protection of transmitted data from passive attacks. The authentication service is concerned with assuring that a communication is authentic. It is the corroboration of the claimed source of a message. Authentication is of two types: (i) Peer entity, and (ii) Data origin. The integrity can apply to a stream of messages, a single message, or selected fields within a message. It assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Access control is the ability

to limit and control the access to host systems and applications via communications links. Each entity trying to gain access must first be identified and/or authenticated [32]. Data integrity can apply to a stream of messages, a single message, or selected fields within a message. It assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual. Non-repudiation prevents either sender or receiver from denying a transmitted message. When a message is sent, the receiver can prove that the alleged sender in fact sent the message.

4. Secured Communications

Secured communication against network is increasing significantly with time. Our communication media should also be secure and confidential. Cryptanalysis is the study used to describe the methods of code-breaking or cracking the code without using the security information, usually used by hackers. For this purpose, the following things can be done by the sender/receiver. [31]

- One can transmit the message secretly, so that it can be saved from hackers.
- The sender ensures that the message arrives to the desired destination.
- The receiver ensures that the received message is in its original form and coming from the authenticate person.

The confidentiality of information that cryptography can provide is useful not only for the legitimate purposes of preventing information crimes e.g. the theft of trade secrets or unauthorized disclosure of sensitive medical records but also for illegitimate purposes e.g., shielding from law enforcement officials a conversation between two terrorists planning to bomb a building. In order to achieve the same one can use two techniques, (i) one can use invisible ink for writing the message or can send the message through the confidential person, and (ii) use of scientific approach called “Cryptography”. The fundamental and classical task of cryptography is to provide confidentiality by encryption methods. It is used in applications present in technologically advanced societies; it includes the security of ATM cards, computer passwords, and electronic commerce. But the most recognized form of cryptography is its use encipher and

decipher information, thus keeping its contents guarded against unauthorized disclosure. There are two classes of key-based encryption algorithms: symmetric and asymmetric algorithms. Symmetric algorithms use the same key for encryption and decryption, whereas asymmetric algorithms use different keys for encryption and decryption. Ideally it is infeasible to compute the decryption key from the encryption key.

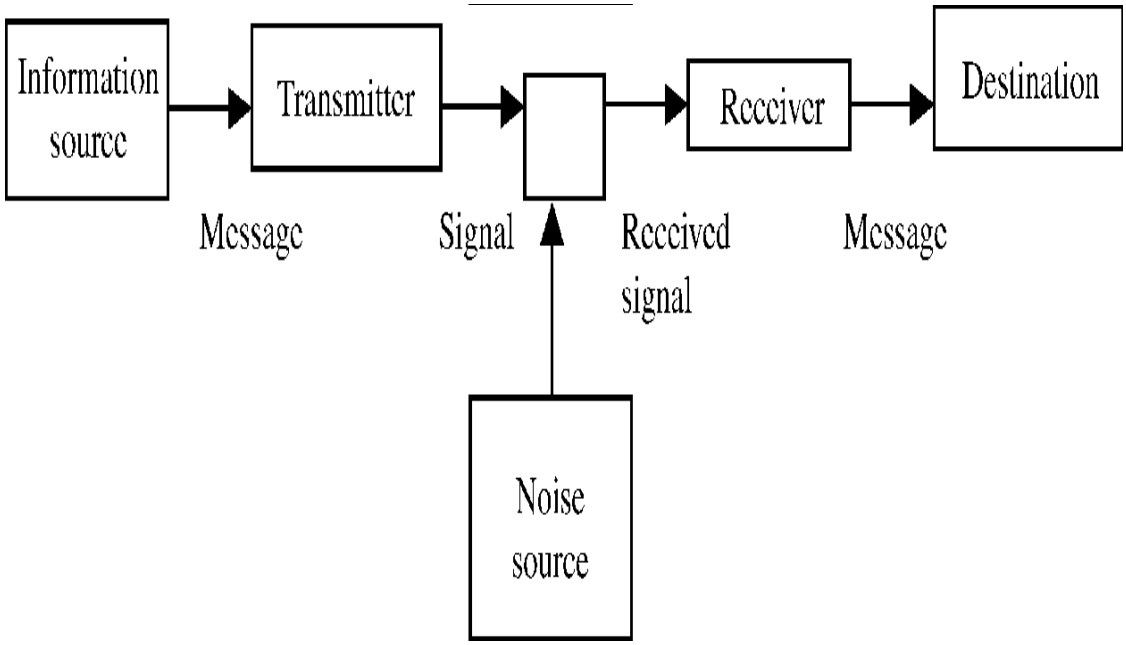


Figure 1.2 Block diagram of secured communication [35]

5. Objectives of the Work

The objectives of the work are

- To study various encryption algorithm used for data security.
- To analyze various tools used in data encryption standard.
- To implement DES in order to achieve security with minimum number of overheads.
- To compare the proposed work with classical DES.

6. Organization of Thesis

The proposed work has been organized into five chapters:

Chapter 1: It includes the introduction of various Cryptography Algorithms and various parameters used for data security. It also has design style and their applications in secured networks. Chapter 2: It gives the literature survey conducted by various author in the field

of cryptography and observation drawn from them. Chapter 3: It includes the introduction of various algorithms of encryption and decryption. Chapter 4: it describes the results obtain by the algorithm with the help of model-sim software. Chapter 5: Finally a conclusion and future scope is given in this chapter.

Chapter 2 Literature Survey

This chapter has the survey of the work done by various researchers in the field of encryption algorithm. The possibility of improvement in the work and motivation has been stated clearly. There is also a brief description of some of the encryption algorithms which are used for encrypting and decrypting in different systems.

2.1 Introduction

Data Encryption Standard (DES) is the most well-known cryptographic mechanism in history [1]. It begins with the work of Feistel at IBM in the early 1970s and culminating in 1977 with the adoption as a U.S. Federal Information Processing Standard for encrypting unclassified information. The most striking development in the history of cryptography came in 1976 when Diffie and Hellman published a transaction [2]. Before the modern era, cryptography was concerned solely with message confidentiality i.e. encryption conversion of messages from the comprehensible form into an incomprehensible one and back again at the other end, rendering it unreadable without secret knowledge. In recent decades, the field has expanded beyond confidentiality concerns to include techniques for authentication, digital signatures, interactive proofs, and secure computation.

Ruth M. Davis [3] provides a hardware-implementable algorithm for enciphering data, which has been adopted as a Federal standard to provide a high level of cryptographic protection against various attacks.

Whitfleld Diffie et. al [4] describes cryptographic technology, which examines the forces driving public development of cryptography. The paper describes how one can secure the message over the telephone lines.

Ingrid Verbauwhede [5] described Security and Performance Optimization of a New DES Data Encryption Chip. Novel CAD tools are used at different steps in the design process for simulation. The result is a single chip of 25 mm in 3-pm double-metal CMOS. Functionality tests show that a clock of 16.7 MHz can be applied, which means that a 32-Mbit/s data rate can be

achieved for all eight byte modes.

James E. Katz [6] provides Social Aspects of Telecommunications Security Policy that describes a system that offers a variety of convenient and powerful services while meeting the legitimate needs of the individual for privacy and of society for security.

H. Bonnenbergt [7] described the VLSI implementation of a new block cipher. The chip that runs with a maximum clock frequency of 33 MHz permitting a data conversion rate of more than 55 Mbits/s performs data encryption and decryption in a single hardware unit.

K.H. Mundt [8] presented superscript ASIC technology that facilitated a new device family for data encryption in which semi-custom cell-based ASIC technology is described to get 100Mbits/s encryption speed on silicon applying 1 micron design rules.

C. Boyd [9] provides the modern data encryption in which proposed standard for digital signatures based on RSA were introduced. A. Curigert describes VINCI: VLSI Implementation of the new secret-key block Cipher IDEA. VINCI's IDEA premier silicon realization, integrates high-speed encryption and decryption, comprehensive key management functions, and all standardized cipher modes of operation in their ordinary and high-speed adapted versions.

R. Zimmermann et. al. [10] provides a 177 Mb/s VLSI implementation of the International Data Encryption Algorithm in which the VLSI chip implements data encryption and decryption in a single hardware unit. All-important standardized modes of operation of block ciphers, such as ECB, CBC, CFB, OFB, and MAC, are supported. Moreover, with a system clock frequency of 25 MHz the device permits a data conversion rate of more than 177 Mb/s.

Stefan Wolter [11] provides the implementation of the IDEA architecture that includes a concurrent self-test based on a mod3 residue code self-checking system. It allows the detection of permanent and temporary single and multiple-bit errors in the IDEA data-path. Hence it provides the secure prevention of faulty encrypted or decrypted data.

Seung-Jo Han [12] describes the improved DES algorithm in which a 96-bit data block is divided into three 32-bit sub-blocks to increase the Unicity Distance (UD) by performing different f-functions on each of the sub-block.

Hassina Guendouz et. al. [13] describes rapid prototype of a fast data encryption standard with integrity processing for cryptographic applications. It implements the DES algorithm in the same silicon area with high-speed performance based on VHDL specifications K. Wong provides a single-chip FPGA implementation of the Data Encryption Standard (DES) algorithm describing the DES algorithm is secured in a single chip FPGA by loading the configuration data into the chip during the initialization cycle. Once the key is loaded, it is secured inside the chip.

K. Wong [14] performed transform domain analysis of DES algorithm by using tool. DES can be regarded as Non Linear Feedback Shift Register (NLFSR) with input and for pseudo-random sequence analysis were applied to S-Boxes in DES. They analyzed the properties of S-Boxes of DES under different transforms. They showed that out of 32 functions from GF(26) to GF(2) associated with eight S-Boxes, around two-third of them had maximal linear span of 63 and the remaining one-third had linear spans greater than or equal to 57. They have also shown that for each of the 32 functions, extended hadamard transform spectra have the same distributions as that of hadamard transform spectra of that function.

M.P. Leong [15] described a bit-serial implementation of the International Data Encryption Algorithm (IDEA) using a novel bit-serial architecture to perform multiplication modulo 216 by having minimal amount of hardware. They also proposed new criteria that can be considered for the design of block-cipher algorithms: i) larger linear span for each component function, ii) the same spectral distribution for all extended Hadamard transforms as for the Hadamard transforms.

R. G. Sixel et. al. [16] describes a high level language implementation of the DES and bit-slice architecture. This implementation had two objectives: (i) testing the whole algorithm prior to a VHDL description for future synthesis, and (ii) by making DES available for other applications requiring a software implementation.

Teo Pock Chueng [17] provides implementation of pipelined DES using Alter CPLD. The architecture contains of three main parts, DES module, pipeline module and control unit module. Four segments pipeline is used in this architecture to burst the throughput of DES and Alter Hardware Description Language (AHDL) is used to implement the pipelined DES design for better output. It allows dynamic circuit specializations which are based on a specific key and mode. When these are combined with a speed efficient layout, the result is a throughput of over 10 Gbits per second.

Yeong-kang lai et. al. [18] represented the VLSI architecture design and implementation for two fish block cipher. In this paper the security of the two fish encryption algorithm was increased by using loop-folding techniques with efficient hardware mapping. Touria arich provides hardware implementation of the data encryption standard in electronic code book mode (ECB) using hardware description language VHDL.

Toby Schaffer et. al. [19] describes an integrated design of Advanced Encryption Standard (AES). This method is used to make it very low complexity architecture and helps in saving the hardware resource in the implementation of AES. It is the techniques using Random number generator using the recurrence matrices and a quadruple vector. It provides data encryption at two levels and hence security against crypto analysis is achieved at relatively low computational overhead using the mod function

Cameron Patterson [20] provides high performance DES encryption in Vertex FPGAs using Jbits. Jbits provides a Java-based Application Programming Interface (API) for the run-time creation and also for the modification of the configuration bit-stream. The author also provided a performance comparison of data encryption algorithms in which various algorithms were compared and it was found that Blowfish algorithm is the best algorithm in view of processing time and security.

Jingmei Liu [21] provides an AES S-Box to increase complexity and cryptographic analysis. An improved AES S-box is presented to improve the complexity of AES S-Box algebraic expression

with terms increasing from 9 to 255 and algebraic degree invariable. The improved AES S-box also has better properties of Boolean functions in SAC and balance, and is capable of attacking against differential cryptanalysis with high reliable security.

Pui-Lam Siu et. al. [22] presented about the A Fault Attack on pairing based Cryptography Current fault attacks against public key cryptography focus on traditional schemes, such as RSA and ECC, and, to a lesser extent, on primitives such as XTR. The work presents the security of pairing-based cryptography against side-channel attack. However, bilinear maps, or pairings, have presented theorists with a new and increasingly popular way of constructing cryptographic protocols.

N. Sklavos et. al. [23] explained the brief description of an operation centred approach to fault detection in symmetric cryptography ciphers one of the most effective ways of attacking a cryptographic device is by deliberate fault injection during computation, which allows retrieving the secret key with a small number of attempts. Several attacks on symmetric and public-key cryptosystems have been described in the literature and some dedicated error-detection techniques have been proposed to foil them. The proposed techniques are ad hoc ones and exploit specific properties of the cryptographic algorithms. In this paper, we propose a general framework for error detection in symmetric ciphers based on an operation-centred approach. We first enumerate the arithmetic and logic operations included in the cipher and analyze the efficacy and hardware complexity of several error-detecting codes for each such operation.

Ahnet Eskicioglu et. al. [24] presented that cryptanalysis with Copacobana. Cryptanalysis of ciphers usually involves massive computations. The security parameters of cryptographic algorithms are commonly chosen so that attacks are infeasible with available computing resources. Thus, in the absence of mathematical breakthroughs to a crypt analytical problem, a promising way for tackling the computations involved is to build special-purpose hardware exhibiting a (much) better performance-cost ratio than off-the-shelf computers. This contribution presents a variety of crypt analytical applications utilizing the Cost-Optimized Parallel Code Breaker (COPACOBANA) machine, which is a high-performance low-cost, cluster consisting of 120 field-programmable gate arrays (FPGAs). COPACOBANA appears to be the only such

reconfigurable parallel FPGA machine optimized for code breaking tasks reported in the open literature. On up to 120 low-cost FPGAs, COPACOBANA is able to perform cryptographic operations simultaneously and in parallel for applications with high computational but low memory and communication requirements.

Dragan Jankovi et. al. [25] explained about the Optimization of polynomial Expressions by Using the Extended Dual Polarity Reed-Muller expressions and their various extensions and generalizations for binary and multiple-valued logic functions are an important class of discrete function representations that are often used in practical applications. These expressions can be uniformly viewed as discrete polynomial expressions over finite fields GF (2) and GF (q) or the field of rational numbers in the case of expressions with integer-valued coefficients. The optimization of them in the number of product terms count is performed by selecting either positive or negative literals for variables in the functions to be represented. Since there are no ways to select in advance the polarity for variables that will result in most compact expression for a given function, all possible expressions have to be generated and the simplest of them selected.

Massimo Alioto et. al. [26] performed differential power analysis attacks to precharged buses which is a general analysis for symmetric-key cryptographic algorithms. They have discussed the general model of multibit Differential Power Analysis (DPA) attacks to precharged buses that provided the deeper insight into the dependence of DPA effectiveness on the parameters which define the attack, the algorithm and the processor architecture in which the algorithm has been implemented. They have proposed the novel figure of merit to measure DPA effectiveness of multibit attacks.

Chin-Chung Lu et. al. [27] presented that Differential Fault Analysis against AES-192 and AES-256 with Minimal Faults The naive implementation of AES is known to be vulnerable to Differential Fault Analysis (DFA). We can find the key of AES-128 AES with 128-bit key with one pair of correct and faulty ciphertext. Due to the longer key size and the characteristic of AES key schedule, we need subtle caution in attacking AES-192 and AES-256. We propose new DFA against AES with 192 and 256-bit key. They could retrieve AES-192 key with two pairs of

correct and faulty ciphertext. With three pairs we could succeed in finding the key of AES-256. These are the minimal faults among the existing methods. He proposed new differential fault analysis against AES with 192 and 256-bit keys. We could retrieve AES-192 key with two pairs of correct and faulty ciphertext. With three pairs we could deduce the secret key of AES-256 within a practical time.

G. Catalini [28] provides Modified Two fish Algorithm for increasing Security and Efficiency in the Encryption of video signals. The proposed system was tested on H 263+ coded signals. The hybrid binary-ternary number system provides both short representations and small density. In this paper, we present three novel algorithms for both single and double scalar multiplication three novel algorithms have been proposed and thoroughly analyzed.

Aamer Nadeem [29] has given the efficient uses of FPGAs for implementations of DES and its experimental linear cryptanalysis. Linear cryptanalysis is a well known plaintext attack which uses a linear relation of input and output bits along with the key bits of encryption algorithm. The FPGA implementation of DES encryption/decryption core is given which can work at rates up to 333MHz. In their design, the plaintext, the key and the mode can be changed without any dead cycles. Also, an FPGA implementation of the linear cryptanalysis of DES is given and the design was deployed on eight FPGAs and found 12+1 key bits in about 2.3 hours. In addition with the new Xilinx FPGA Xilinx VIRTEX-II XC2V8000, the same attack can be carried out in about 1 hour, using only one FPGA board.

M. McLoone [30] provides high-performance FPGA implementation of DES using a novel method for implementing the key schedule based on Xilinx Vertex technology. Utilizing the novel method, a 16-stage pipelined DES design is achieved, which can run at an encryption rate of 3.87 Gbits/s. The author explained about that Hybrid Binary-Ternary Number System for Elliptic Curve Cryptosystems Single and double scalar multiplications are the most computational intensive operations in elliptic curve based cryptosystems. Improving the performance of these operations is generally achieved by means of integer recoding techniques, which aim at minimizing the scalars' density of nonzero digits. The first one, called w-HBTF is a family of algorithms for single scalar multiplication. It combines the hybrid binary-ternary

number system with widely used windowing methods. The other two algorithms, namely, HBTJF and RHBTJF, are for double scalar multiplication. Which algorithm should be used for the implementation of an elliptic curve protocol depends on several parameters: the amount of memory that is available to store the precomputed points, the size of the finite field, and the type of elliptic curve. For elliptic curves with fast tripling, like tripling-oriented DPA curves, our algorithms are likely to provide significant improvements.

The following table section consists of the comparisons of various parameters of the different types of algorithms as elaborated below.

Parameters/Algorithms	DES	TDES	AES	Two fish	Blowfish
Key Size (Bits)	56	128	128,192,256	256	448
Block Size (Bits)	64	64	128	128	64
Processing Time (Seconds)	14	42	21	15	11
Bytes/Sec	7988	2633	5320	10167	10167
Reliability	Yes	Yes	Much Strong	Strong	Strong

Table 2.1 Comparison of Various Algorithms

2.2 Observation from the Related Work

The following observations have been drawn from the literature survey

- To achieve data security with minimum number of overheads. For this the encryption algorithm needs to be quite simple and to achieve high degree of security key should be large.
- It is clear that key plays an important role in encryption process, which is a main part of cryptography. If the key is slightly changed then almost all the data is wasted so the key management must be good enough to provide accurate result.

- It has been observed that the speed of operation can be enhanced by using more hardware, as a result the heat dissipation and size of the board is increased.
- Hybrid Binary-Ternary Number System for Elliptic Curve Cryptosystems Single and double scalar multiplications are the most computational intensive operations in elliptic curve based cryptosystems but it is not enough to overcome such problem. This can be achieved with the help of tripling-oriented DPA curves.

2.3 Need of Work

After the study of various encryption algorithms, the concept of keys has been successfully observed. From the observation it has been seen that better secured system can be achieved by increasing the key length. Longer key lengths consume more power and dissipate more heat. Basically it is a tradeoff, between security and overheads. In order to achieve more secured system continuous efforts are required. An efficient encryption algorithm should consist of two factors – fast response and reduced complexity. By keeping the utility of encryption algorithm in secure communication it is desirable to optimize and/or improve the encryption techniques, so security overheads remains under control.

Chapter 3 Encryption Algorithms

This chapter includes the basics of the FPGA implementation and various types of cryptography algorithms are elaborated. It also includes the key management and rules for key generation with the help of tool.

1. Basic cryptography

The term cryptography is defined as the conversion of plain text into cipher text with help of key is known as cryptography. There are two main processes in the cryptography, named as encryption and decryption. In the encryption process the key has been used to convert the plain text into cipher text [31]. The key may be any word or value and is known to only sender and receiver. While in the decryption process the key has been used to convert the cipher text into plain text.

There are basically three types of Cryptography:

- Symmetric (Private) Cryptography
- Asymmetric (Public) Cryptography
- Modern Cryptography

Symmetric (Private) Cryptography

The term symmetric key cryptography is referred as if both sender and receiver both use same key or secret key is known as symmetric encryption. It is also known as 'conventional encryption' or 'single key' encryption.

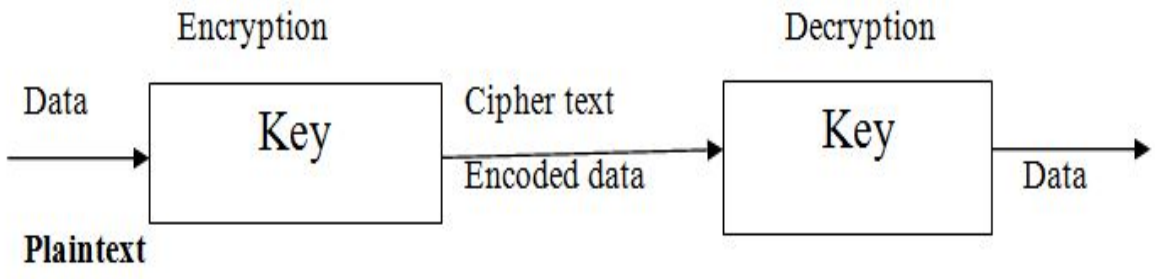


Figure 3.1: Simplified model of symmetric key cryptography

This section consists of basic advantages and disadvantages of symmetric-key cryptography are explained below and also describe how symmetric key cryptography is helpful in today's communication world of electronic media.

a. **Advantages of symmetric-key cryptography**

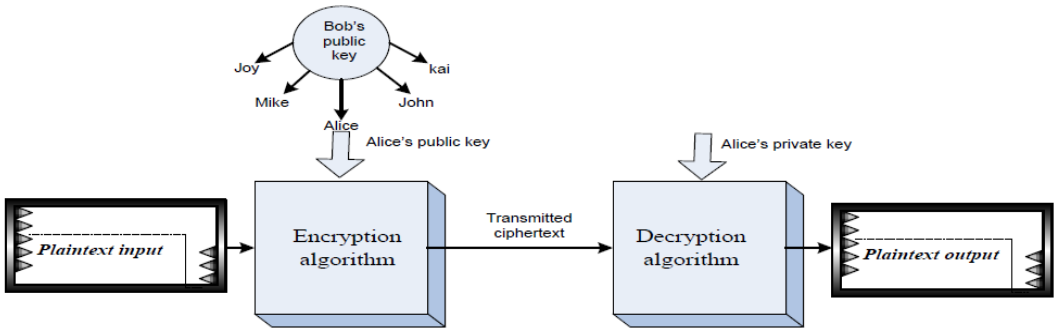
- Symmetric-key ciphers can be designed to have high rates of data throughput.
- Keys for symmetric-key ciphers are relatively short.
- Symmetric-key ciphers can be employed as primitives to construct various cryptographic mechanisms including pseudorandom number generators, hash functions and computationally efficient digital signature schemes, to name just a few.
- Symmetric-key ciphers can be composed to produce stronger ciphers. Simple transformations which are easy to analyze, but on their own weak, can be used to construct strong product ciphers.

b. **Disadvantages of symmetric key cryptography**

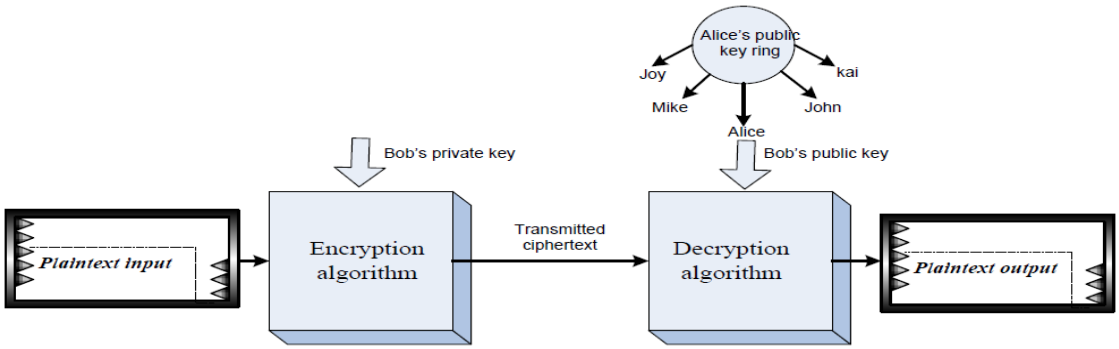
- In a two-party communication, the key must remain secret at both ends.
- In a large network, there are many key pairs to be managed. Consequently, effective key management requires the use of an unconditionally trusted TTP.
- In a two-party communication between entities A and B, sound cryptographic practice dictates that the key be changed frequently and perhaps for each communication session.
- Digital signature mechanisms arising from symmetric-key encryption typically require either large keys for the public verification function or the use of a TTP.

Asymmetric (Public) Cryptography

The term asymmetric key encryption is referred as if both sender and receiver both use different key or two key is known as asymmetric encryption [33]. It is also known as public key cryptography.



(a)



(b)

Figure 3.2: Simplified Model of Asymmetric Key Cryptography [31]

This section consists of basic advantages and disadvantages of asymmetric-key cryptography are explained below and also describe how asymmetric key cryptography is helpful in today's communication world of electronic media

Advantages of public-key cryptography

- Only the private key must be kept secret.
- Depending on the mode of usage, a private key/public key pair may remain unchanged for considerable periods of time, e.g., many sessions.
- Many public-key schemes yield relatively efficient digital signature mechanisms. The key used to describe the public verification function is typically much smaller than for the symmetric-key counterpart.
- In a large network, the number of keys necessary may be considerably smaller than in the symmetric-key scenario.

Disadvantages of public-key encryption

- Throughput rates for the most popular public-key encryption methods are several orders of magnitude slower than the best-known symmetric-key schemes.
- Key sizes are typically much larger than those required for symmetric-key encryption, and the size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques.

Modern Cryptography

In the modern cryptography a combination of both public-key and traditional symmetric cryptography is used in modern cryptographic systems [32]. The reason for this is that public-key encryption schemes are computationally intensive versus their symmetric key counterparts. Because symmetric key cryptography is much faster for encrypting bulk data, modern cryptography systems typically use public-key cryptography to solve the key distribution problem first, and then symmetric key cryptography is used to encrypt the bulk data.

2. Encryption Techniques

The process of converting plain text to cipher text is known as encryption and the algorithm which encrypts the data is known as encryption algorithm [31]. The encryption techniques are described below.

1. Data Encryption Standard

Data Encryption Standard (DES) is a cryptographic standard that was proposed as the algorithm for secure and secret items in 1970 and was adopted as an American federal standard by National Bureau of Standards (NBS) in 1973. It is one of the most widely accepted, publicly available cryptographic systems today [34]. It was developed by IBM in the 1970s but was later adopted by the US government as a national standard DES is a block cipher, which means that during the encryption process, the plain-text is broken into fixed length blocksand each block is encrypted at the same time. Basically it takes a 64 bit input plain text and a key of 64-bits (only 56 bits are used for conversion purpose and rest bits are used for parity checking) and produces a 64 bit cipher text by encryption and which can be decrypted again to get the message using the same key [29]. The simplified DES is shown in Fig. The algorithm employs three different

types of operations: Permutations, Rotations, and Substitutions. Between the initial and final transpositions, the algorithm performs 16 iterations of a function.

Working measures of DES

DES uses a 56-bit key. In fact, the 56-bit key is divided into eight 7-bit blocks and an 8th odd parity bit is added to each block i.e., a "0" or "1" is added to the block so that there is an odd number of '1' bit in each 8-bit block [34]. By using the 8 parity bits for rudimentary error detection, a DES key is actually 64 bits in length for computational purposes although it only has 56 bits worth of randomness, or entropy. DES then acts on 64-bit blocks of the plaintext, invoking 16 rounds of permutations, swaps, and substitutes, as shown in Figure. The standard includes tables describing all of the selection, permutation, and expansion operations mentioned below; these aspects of the algorithm are not secrets [31]. The basic DES steps are

- The 64-bit block to be encrypted undergoes an initial permutation (IP), where each bit is moved to a new bit position; e.g., the 1st, 2nd, and 3rd bits are moved to the 58th, 50th, and 42nd position, respectively.
- The 64-bit permuted input is divided into two 32-bit blocks, called left and right, respectively. The initial values of the left and right blocks are denoted L0 and R0.
- There are then 16 rounds of operation on the L and R blocks. During each iteration (where n ranges from 1 to 16), the following formulae apply:

$$= \quad _{-1}$$

$$= \quad _{-1} \qquad (\quad _{-1}, \quad)$$

At any given step in the process, the new L block value is merely taken from the prior R block value. The new R block is calculated by taking the bit-by-bit exclusive-OR (XOR) of the prior L block with the results of applying the DES cipher function, *f*, to the prior R block and K_n. K_n is a 48-bit value derived from the 64-bit DES key. Each round uses a different 48 bits according to the standard's Key Schedule algorithm.

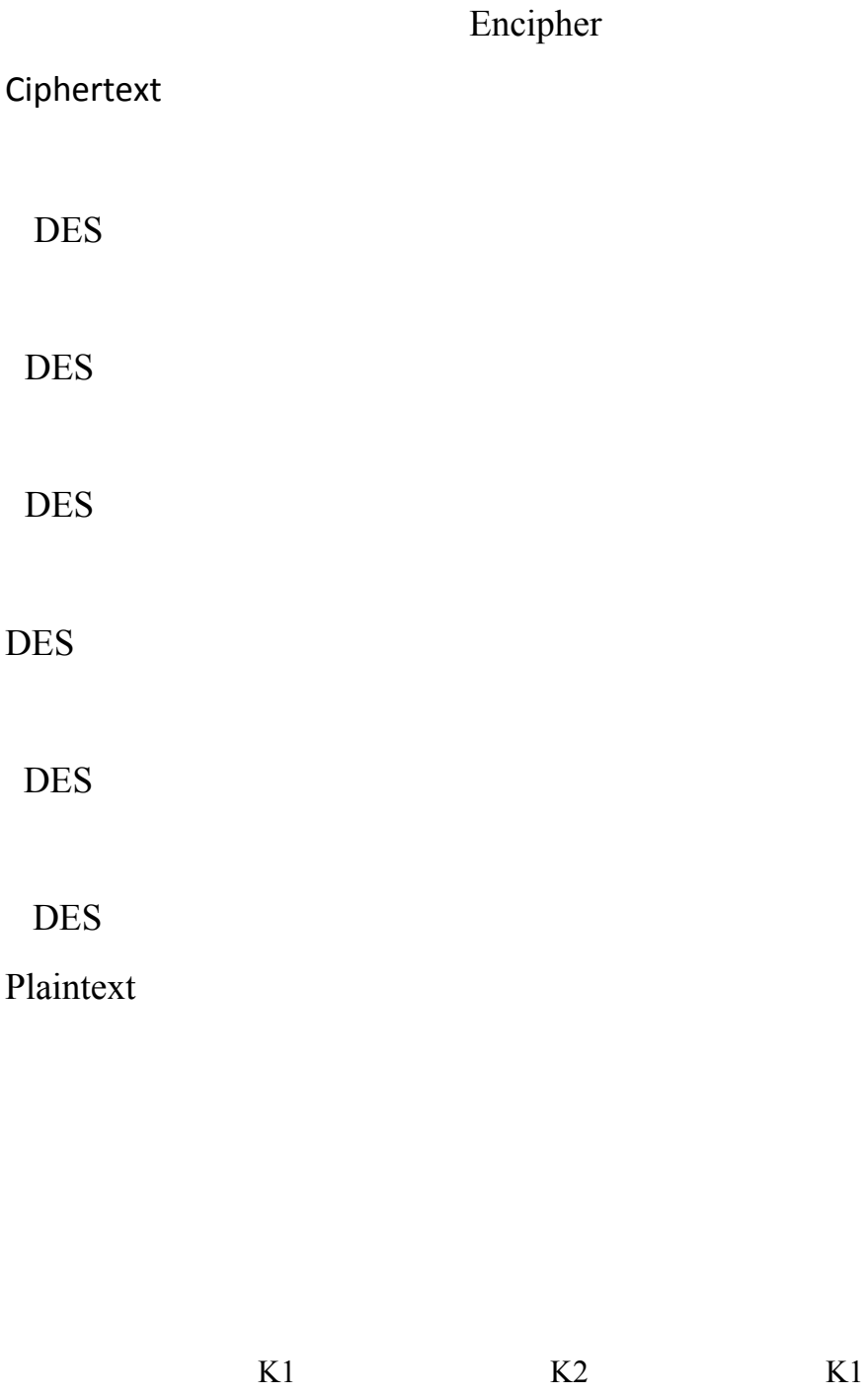
The cipher function, f, combines the 32-bit R block value and the 48-bit sub key in the following way. First, the 32 bits in the R block are expanded to 48 bits by an expansion function (E); the extra 16 bits are found by repeating the bits in 16 predefined positions. The 48-bit expanded R-block is then XORed with the 48-bit sub key [32]. The result is a 48-bit value that is then divided into eight 6-bit blocks. These are fed as input into 8 selection (S) boxes, denoted S1, S2,…,

S8. Each 6-bit input yields a 4-bit output using a table lookup based on the 64 possible inputs; this results in a 32-bit output from the S-box. The 32 bits are then rearranged by a permutation function (P), producing the results from the cipher function.

(4) The results from the final DES round — i.e., L16 and R16 — are recombined into a 64-bit value and fed into an inverse initial permutation (IP-1). At this step, the bits are rearranged into their original positions, so that the 58th, 50th, and 42nd bits, for example, are moved back into the 1st, 2nd, and 3rd positions, respectively. The output from IP-1 is the 64-bit cipher text block.

2. Triple Data Encryption Standard (TDES)

DES uses a 56 bit key and is not sufficient to encrypt sensitive data [33]. While triple data encryption standard goes through 3 iteration of DES, effectively encrypting data with a168-bit key strong enough to secure sensitive information. The data is first encrypted using 56-bit key, decrypted with another 56-bit DES key, and finally encrypted again with originally 56-bit key. Because triple DES contains several levels of encryption, it can better protected from the attacks [31]. DES is very fast algorithm, and through triple des is a little slower, it is still faster than some other symmetric algorithm. The biggest advantage for triple DES, through is that is compatible with all software and hardware that support DES. Triple DES runs three times slower than DES, but is much more secure if used properly. The procedure for decrypting something is the same as the procedure for encryption, except it is executed in reverse. Like DES, data is encrypted and decrypted in 64-bit chunks. Although the input key for DES is 64 bits long; the actual key used by DES is only 56 bits in length [34]. The least significant rightmost bit in each byte is a parity bit, and should be set so that there are always an odd number of 1s in every byte. These parity bits are ignored, so only the seven most significant bits of each byte are used, resulting in a key length of 56 bits. This means that the effective key strength for Triple DES is actually 168 bits because each of the three keys contains 8 parity bits that are not used during the encryption process. Triple DES has always been regarded with some Suspicion since the original algorithm was never designed to be used in this way, but no serious flaws have been uncovered in its design, and it is today available cryptosystem used in a number of Internet protocols.


Encipher
Ciphertext
DES
DES
DES
DES
DES
DES
Plaintext
K1
K2
K1

Decipher

Figure 3.3: Structure of triple DES

3. **International Data Encryption Algorithm (IDEA)**

International Data Encryption Algorithm (IDEA) was originally developed in 1990 as the Proposed Encrypted Standard (PES). In 1992, it was renamed IDEA. IDEA is a block cipher that uses 64-bit data blocks and 128-bit key. Even though some consider some this is stronger algorithm than Triple DES, it has not gained wide acceptance and usage in the market. It is a block cipher and operates on 64-bit blocks of data. The key is 128 bits long. The 64-bit data block is divided into 16 smaller blocks and each has eight rounds of mathematical functions performed on it [37]. It offers different modes similar to the modes described in the DES section, but it is much harder to break than DES. IDEA is used in the PGP encryption software. It was thought to replace DES, but it is patented, meaning that licensing fees would have to be paid to use it.

4. **Blowfish Algorithm**

It is designed in 1993 by Bruce schneier. It is a symmetric block cipher that can be used for encryption and providing security for the data. Basically it takes variable length key from 32 bits to 448 bits and it is unpatented, license-free and easily available for user [31]. It works on a Feistel network and iterating in 16 times (horst fiestal). It consist of 64 bit block size as input and have scalable key 32 to 256 bits and have simple operations as Exclusive-or, addition, table look up modular- multiplication. In the encryption operation it consist of two parts-

a. Key-expansion part

b. Data-encryption part the key expansion convert key of 448 bits into a sub key array totaling 4168 bytes. Data encryption occurs via 16- rounds of fiestal network and each round consists of a key dependent substitution. It uses the large number of sub keys and may be computed before any data encryption or decryption. It is a symmetric block cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data. It is unpatented and license-free, and is available free for all uses. This Algorithm is a Feistel Network, iterating a simple encryption

function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors [33]. It is a variable-length key block cipher. It is suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches.It has a 64-bit block size and a variable key length from 32 up to 448 bits. It Flow is shown in Fig. 3.3. Blowfish exhibits fast and versatile performance across most platforms; it performs well both in hardware and in memory-constrained environments [35]. Blowfish can be optimized for speed, key setup, memory, code size in software, or space in hardware. Encrypt data in less than 500 clock cycles per block on an Intel Pentium, Pentium Pro, and Pentium II, for a fully optimized version of the algorithm.

5. Pretty Good Privacy (PGP)

In the pretty good privacy the online privacy is a rather complex issue then the other one. The electronic conversation between the staff and the user should be secured so that no third party can be occupied that what text has been transmitted. The parties of the conversation should also to verify that they are communicating with the current recipient. This required other thing robust cryptographic protocols, which need a reliable method of distributing the encryption keys of the recipient inside (and outside) the organization. Also to be ever actually used the organizations security and privacy policies should take into account and encourage the use of more secure e-mail practices.

The most popular and publicly spread e-mail encryption software utilizing public key cryptography was written in 1990 by Philip R. Zimmermann. The software, called pretty good Privacy or PGP, provides both encryption and signing operations along with some Utility services such as data compression and character set encoding. PGP features hybrid encryption where the messages transmitted are first conventionally encrypted with a symmetric key [2, 5, 20]. This key is then encrypted or signed or both with a public key algorithm. It is possible, and often very desirable, to encrypt a single message with multiple public keys. The encrypted message can then be distributed on public channels and every member of the group whose public

keys were used can use his secret key to decrypt the message. PGP succeeds by taking the most conceptually difficult part of its operation (deciding who should be trusted and who shouldn't), and making it the responsibility of the human being who uses PGP rather than the program itself. It makes life very much easier for the implementer of the PGP program; it is also in some sense the right thing to do, as the program itself lacks any knowledge which it could use to make a rational decision to trust someone or not. However, this approach also has a drawback: it places an additional burden on the user, and more significantly, it requires that a human being actually be present when the PGP program is run [33]. This latter point is not a big problem when PGP is used to secure personal e-mail; the human being has to be there to understand the contents of the message, so they might as well help out with the security operations while they're at it. The requirement for a human to be present becomes more burdensome when it is desired to extend the application area of PGP include things such as electronic commerce or remote access to databases. In these new application areas, it makes sense to have a computer running completely unattended and acting on messages received. It is unacceptably expensive to require a computer operator to be present just to support the security, if they wouldn't otherwise be needed. In addition, when PGP is used for the official business of an organization (as opposed to personal use), there is the issue that the interests of the organization and the interests of the person employed to run the program might not be entirely the same, and as a result the operator might deliberately make a bad decision.

6. Public Key Infrastructure (PKI)

In the PKI the public keys of the users are wrapped in certificates by appointed trusted third parties, Certificate Authorities (CA). There are different kinds of certificates, some for identifying persons and some for identification of computers and so services. When a user starts using, for e.g., encrypted mail, he first requests a personal certificate from his local CA [2, 4, 21]. A root CA has appointed the local CA and it has been given the rights to generate new certificates. The CA generates a public key key-pair similar to PGP and wraps them along with the users name and other contact information into a certificate. The CA signs the certificate with its own secret key that in turn is signed by a higher level (root) CA. The public key is placed into the CAs repository. The user gets his certificate and installs it into his e-mail client [34]. When the user starts to send encrypted e-mail to a recipient, the e-mail client fetches the recipient’s

public key from the CA. If the CA doesn't have the key, it queries another CA higher in the hierarchy all the way up to the root CAs, until the key is found.

7. Diffie-Hellman Algorithm

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. First, we define a primitive root of a prime number p as one whose powers modulo p generate all the integers from 1 to p 1. That is, if a is a primitive root of the prime number p, then the numbers a mod p, a2 mod p,..., ap1 mod p are distinct and consist of the integers from 1 through p 1 in some permutation. For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that b ai (mod p) where 0 i (p 1).

The exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as dloga,p (b). The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret values.

NODE B

NODE A

A

BStep1

Step1

PUBLIC VALUE

PUBLIC VALUE

A

B Step2

Step2

PRIVATE VALUE

PRIVATE VALUE

Step3

Step3

PRIVATE VALUE A COMBINED WITH PUBLIC VALUE B

PRIVATE VALUE B COMBINED WITH PUBLIC VALUE A

A B

SECRET SHARED KEY

SECRET SHARED KEY

Step4

Step4

Figure: 3.4 Structure of the Diffie-Hellman algorithm [31]

3. Types of Ciphers

- **Substitution Cipher**

It is the one in which the letter of plaintext are replaced by other letter or by numbers or symbols such process are known as substitution techniques [35]. There is various substitution techniques are as follows:

Caser Cipher: It has been discovered by Julius Caser. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.

Plain text: My Name is Gaurav

Cipher text: - Ob Qdoh lv Jdxudy

One can be able create their own algorithm by selecting the key. Example, same data has been encrypted by using R5 (shifting of data by 5 bits towards right)

Plain text: My Name is Gaurav

Cipher text: - Rd Serj nx Lezwea

Mono alphabetic Cipher:

In mono alphabetic cipher the cipher text can be any permutation of the 26 alphabetic.

Strength

The main strength is that it is using 26! Keys for encrypting a single word of a plain text i.e. $4*10^{26}$ possible keys.

Limitation

- If you know the nature of plaintext, any substitution cipher, regardless of the size of the key space, can be broken easily with a statistical attack.
- When the plaintext is plain English, a simple form of statistical attack consists measuring the frequency distribution for single characters, for pairs of characters, for triples of characters, etc., and comparing those with similar statistics for English.

The easiest of these ciphers is where the letters are offset by a fixed amount. For instance if the offset value is three, then a letter A in the plaintext becomes a D in the cipher text. At the end of

the alphabet the letters wrap around to the beginning [37]. It will require the student to reset to the beginning of the alphabet when a value runs off the end. The plaintext can have punctuation and spaces already removed or the program can be required to do that.

Plaintext=> A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Cipher text=> D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Play Cipher

The play fair cipher has been discovered by the LORD PETER WIMSEY .it is also known as the best known multiple letter encryption cipher. The play fair algorithm is based on the use of a 5×5 matrix of letter constructed using a keyword. The Keyword is Monarchy

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	W	W	X	Z

Table 3.1: the play cipher approach for encryption

G	A	U	R	O	V
B	C	D	E	F	H
I	J	K	L	M	N
P	Q	S	T	W	X
Y	Z	0	1	2	3
4	5	6	7	8	9

Table 3.2: 6×6 Play Cipher for Encryption of Data

The above mentioned technique has 6×6 cells which cop up the problem of representation of numeric values. Here the Keyword is Gaurov.

- **Transposition Cipher**

In the transposition techniques are very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred as transposition

ciphers. The simplest such cipher is the rail fence techniques in which plaintext is written in diagonal and then read off as a sequence of rows [35]. In a **transposition cipher**, permutation is used, meaning that letters are scrambled. The key determines the positions that the characters are moved to, as illustrated. This is a simplistic example of a transposition cipher and only shows one way of performing transposition. When introduced with complex mathematical functions, transpositions can become quite sophisticated and difficult to break. Most ciphers used today use long sequences of complicated substitutions and permutations together on messages. The key value is inputted into the algorithm and the result is the sequence of operations (substitutions and permutations) that are performed on the plaintext [31]. Simple substitution and transposition ciphers are vulnerable to attacks that perform ***frequency analysis***. In every language, there are words and patterns that are used more often than others. For instance, in the English language, the words "the," "and," "that," and "is" are very frequent patterns of letters used in messages and conversation. The beginning of messages usually starts "Hello" or "Dear" and ends with "Sincerely" or "Goodbye." These patterns help attackers figure out the transformation between plaintext to cipher text, which enables them to figure out the key that was used to perform the transformation.

It is important for cryptosystems to not reveal these patterns. More complex algorithms usually use more than one alphabet for substitution and permutation, which reduces the vulnerability to frequency analysis.

4. Digital Signatures

Public key cryptography gives a major benefit by providing a method for employment of digital signatures. Digital signatures and hand-written signatures both rely on the fact that it is very hard to find two people with the same signature. The authentication and data integrity are the two features of digital signatures by providing a seal over a document or a handwritten signature. Anyone with access to the public key of the signer may verify the signature. For example, in the field of E-commerce, an instruction to your bank to transfer money can be authenticated with a digital signature [31]. Digital signatures cannot be copied to another document.

Figure: 3.5 Steps for digital signature generation [31]

The steps for the digital signature generation are as follows:

1) The message digest of the plain-text is computed, i.e., what type of data is - text, video, image, etc. and what is the length of the data is, etc.

2) This message digest will be encrypted using key techniques and a digital signature is also attached before sending to the receiver.

3) The encrypted plain-text (including plain-text and digital signature) is sent to the receiver.

Digital Signature Verification:

The steps for the digital signature verification are as follows:

1) The message digest of the file is evaluated for the verification purpose.

2) The encrypted file, i.e., cipher-text is decrypted using the key techniques.

3) After then the decrypted file and original file are compared for the verification. If two files are same then the decrypted file will be accepted, otherwise, it will be rejected.

5. Key Management

Cryptography can be used as a security mechanism to provide confidentiality, integrity, and authentication, but not if the keys are compromised in any way. The keys have to be distributed to the right entities and updated continuously [31, 35]. The keys need to be protected as they are being transmitted and while they are being stored on each workstation and server. The keys need to be generated, destroyed, and recovered properly. Key management can be handled through manual or automatic processes. The frequency of use of a cryptographic key can have a direct correlation to how often the key should be changed. The more a key is used, the more likely it is to be captured and compromised. Keeping keys secret is a challenging task. Keys should not be in clear-text outside the cryptography device

Rules for keys generation and their handling:

1. The key length should be of variable size for the highly secure communication.
2. Keys should be randomly selected by using the full spectrum of available key-space.
3. Multiple use of keys leads to short lifetime.
4. Keys should be properly destroyed when their lifetime is over.
5. For the secure communication, the keys are to be kept secret

6. Introduction to FPGA

The Xilinx Spartan – 3E (XC3S500) FPGA kit is used for implementation of encryption algorithms [37]. The Xilinx ISE 9.2i tool is used for the design. Some specifications of Spartan – 3E kit are as following:

- Up to 232 user – I/O pins
- Over 10,000 logic cells
- 2 – line, 16 – character LCD screen
- PS/2 mouse or keyboard port
- VGA display port
- Two 9 – pin RS – 232 ports (DTE/DCE)
- 50 MHz clock oscillator
- Chip scope Soft Touch debugging port
- Eight discrete LEDs
- Four slide switches
- Four push-button switches
- Speed Grade -4
- FG320 package

A field programmable Gate Array (FPGA) is a reprogrammable logic device used as a reprogrammable alternative to ASIC chip devices [36]. FPGAs can be used for specific operational behavior, or general purpose CPU functionality depending on the complexity of the device. FPGA applications include DSP applications, imaging, speech recognition, cryptography, hardware emulation and for many other application specific uses. Many FPGAs are implementing shared general purpose CPUs on chip for shorter latency times between operations resulting in higher performance of the overall system. FPGA is an integrated circuit (IC) that includes a two-dimensional array of-

- Configurable Logic Blocks (CLBs)
- Configurable I/O Blocks (IOB)

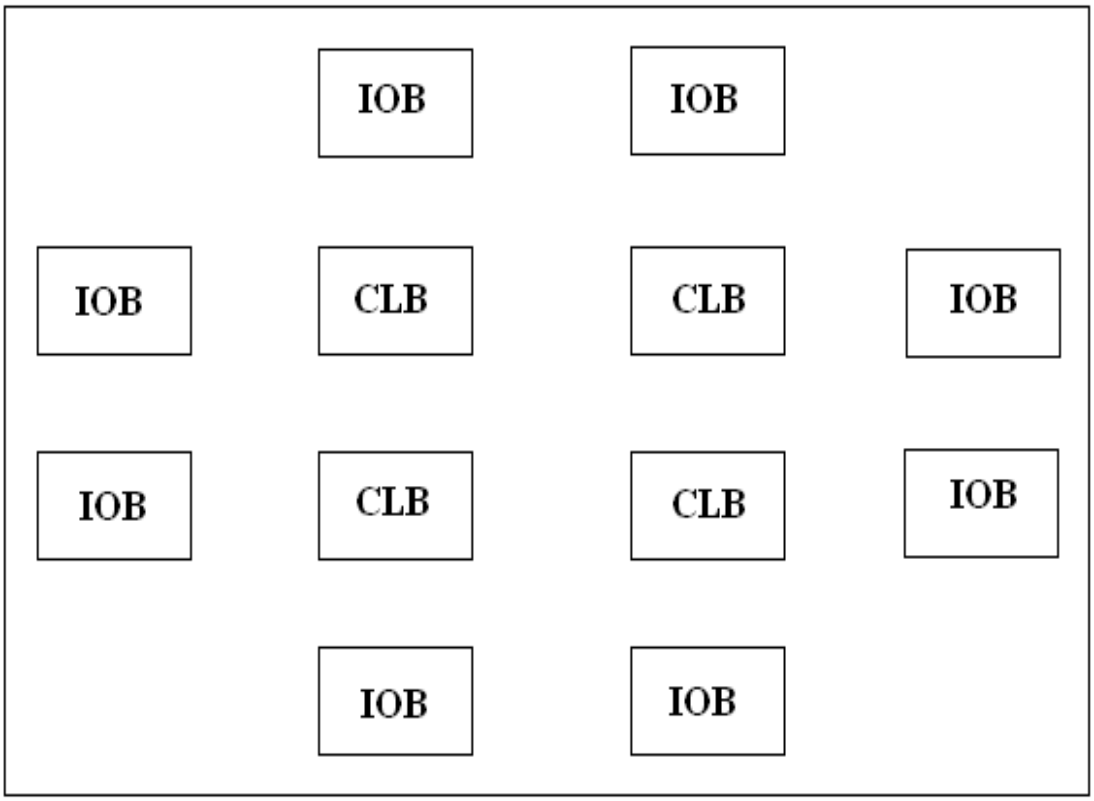


Fig 3.6 FPGA Architecture

7. Design Flow

The design flow of the proposed scheme in model sim and Xilinx 9.1e has been detailed and shown in the following sections with the help of data flow graph.

1. Design Entity

The HDL languages – Verilog or VHDL are used to design the architecture of the system. A VHDL design can be processed by its entity and architecture declarations [37]. The architecture declaration can be of behavioral, dataflow and structural modeling style.

2. Behavioral Simulation

By using behavioral simulation we can verify the functionality of the code designed for our system [36]. Various modifications can be done in case of getting errors in the code.

3. Synthesis

The synthesis process includes the compilation of the sub-modules in the main module and the analysis of the hierarchy of the design [36]. It checks the syntax of the design gives the output in the netlist format which is saved to a Native Generic Circuit (NGC).

4. Design Implementation

It includes the following three processes:

- Translation
- Mapping
- Place & Route

Translation process compiles all the input netlist and constraints to a logic design file, namely – Native Generic Database (NGD) file having extension node. In this process, the ports are assigned to physical elements such as pins, switches, buttons, etc. This information is stored in the UCF (User Constraint file) file.

Mapping process divides the whole circuit into sub-blocks so that they can fit into FPGA blocks. It includes the mapping of input NGD file logic into targeted FPGA elements such as CLBs and IOBs

Place and Route process places the sub-blocks from map process into the logic blocks according to the constraints and connects the logic blocks. This can be done by using Place and Route (PAR) program.

The various pins, input/output ports and their functions have been described in table 3.1.

Pin name	**I/O**	**Function**
CLK	I	Core clock signal
RESET	I	Core reset signal
CEN	I	Synchronous enable signal. When LOW the core ignores all its inputs and all its outputs must be ignored
START	I	When goes HIGH, a cryptographic operation is started
E/D	I	Encrypt – 1 / Decrypt – 0
MODE [1:0]	I	DES Mode 0 – Single DES 1 – Double DES

Table 3.3 Describes the input and output port of kit

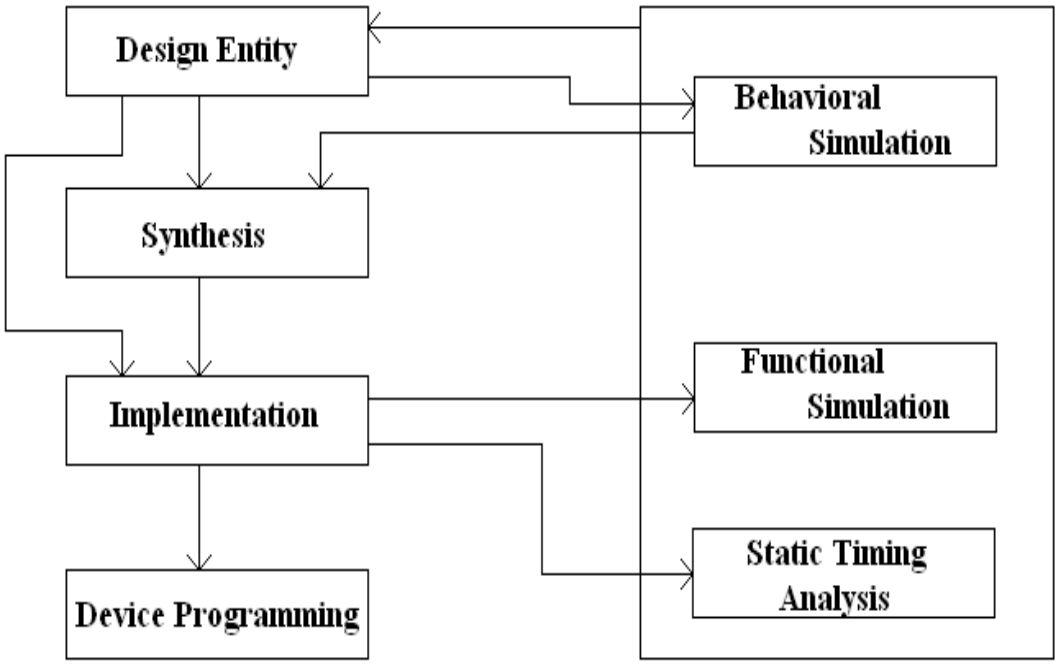


Figure 3.7: Design Flow of FPGA

This chapter discusses various types of techniques of encryption algorithm as well as decryption algorithm. If the two persons want to communicate with each other, it is necessary that they should have private key communication. On the other if the data is transfer on large scale (electronic mail), the organizer as well as the user both should have their own keys. This will solve the problem of communication on the large scale. The main task in cryptography is to how to handle the keys. Keys must be managed by keeping an eye on the encryption technique. To illuminate we have a trusted third party, this will provide the duplicate key to the receiver. But this method reduces the security because one more person knows the key. In the encryption process each word the data stream must be converted into binary so that it is X-ored with the key. The data flow of the proposed scheme has been implemented successfully.

Chapter 4 Results and Discussions

This chapter contains the architecture of the modified DES algorithm implemented which is faster than the conventional DES and is found to be more resource efficient. The encryption technique based on this Polybius's table and anagrams is made in this thesis work. In the program the data or information containing the letters is first converted into their equivalent numbers then they are converted to their equivalent binary numbers in order to provide more safety. For example if our information is SPRING then it is converted to 100 011 011 101 100 010 010 100 011 011 010 010 (43 35 42 24 33 22). To achieve more security this sequence is X-ored with other sequence which act as key. For the above example let us take SAFETY as a key, the equivalent binary number of this 100 011 001 001 010 001 001 101 100 100 101 100 (43 11 21 15 44 54), after Xoring the result comes out to be:

SPRING	100 011 011 101 100 010 010 100 011 011 010 010
SAFETY	100 011 001 001 010 001 001 101 100 100 101 100
X-ored	000 000 010 100 110 011 011 001 111 111 111 110

Then this X-ored sequence is transmitted so that, if anyone hacks this sequence then he/she would not able to understand the data. At the receiver section this sequence is Xored with the same key, this will give the original information. This process is known as decryption.

- **Simulation Result**

Initially we apply the test vectors to our design through the test bench and then compile, simulate our test bench with the help of model-sim software. When the simulation is completely performed then we analysis the result with help of waveform generated by the model-sim. After analyzing the wave forms, we found that output is obtained after every 15 clock cycles. The obtained waveforms are shown in following given figures:

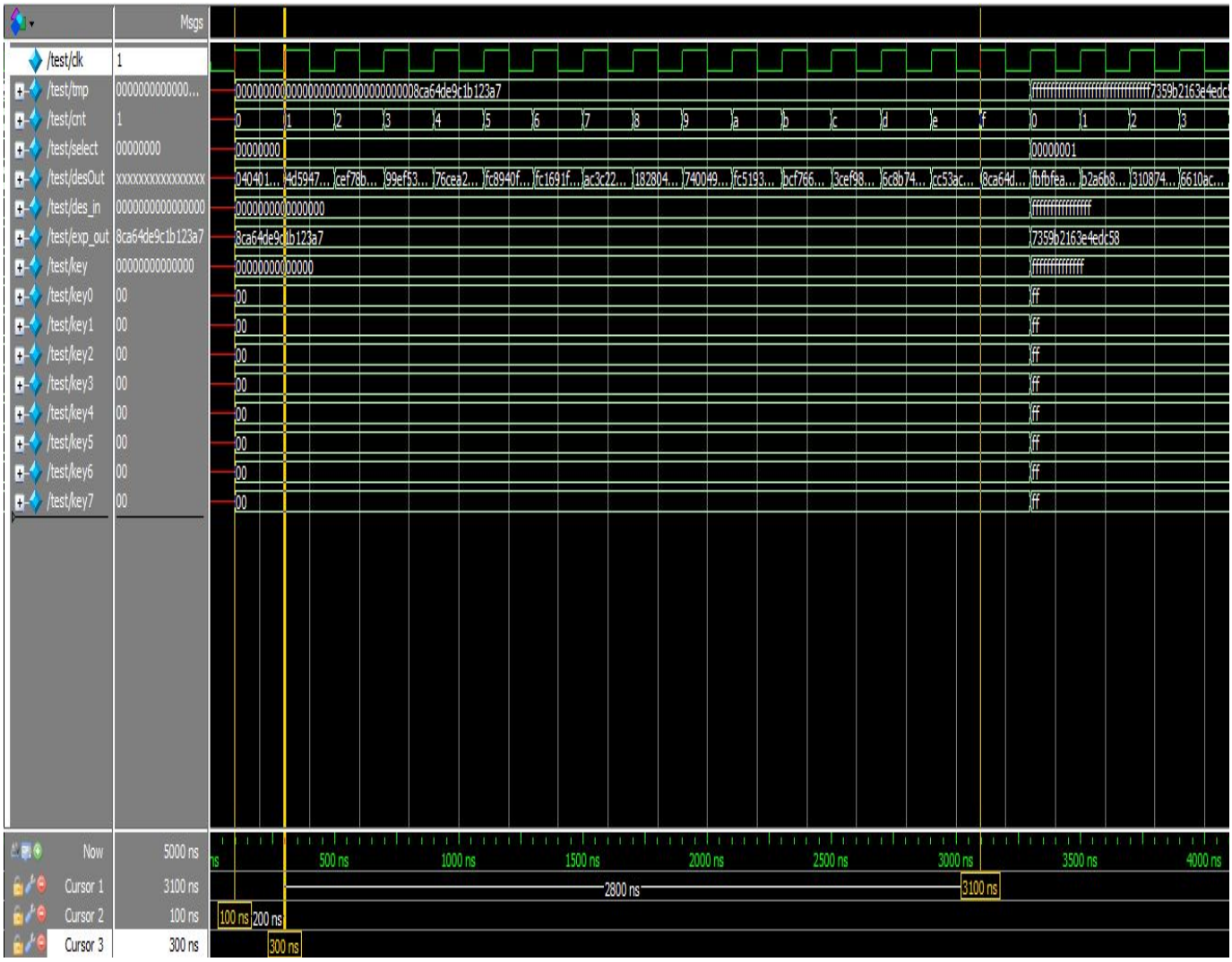


Figure 4.1: Simulation Result of implemented algorithm with 1st input

It has been shown in figure 4.1 that with a given data input of 64 bits and the key used to encrypt data of 56 bits. At 100 ns, the data has been input to encrypt and after 15 clock pulses, the out data has been received at the data out pin. Each clock cycle is of 200 ns and thus the output is produced at 3100 ns.

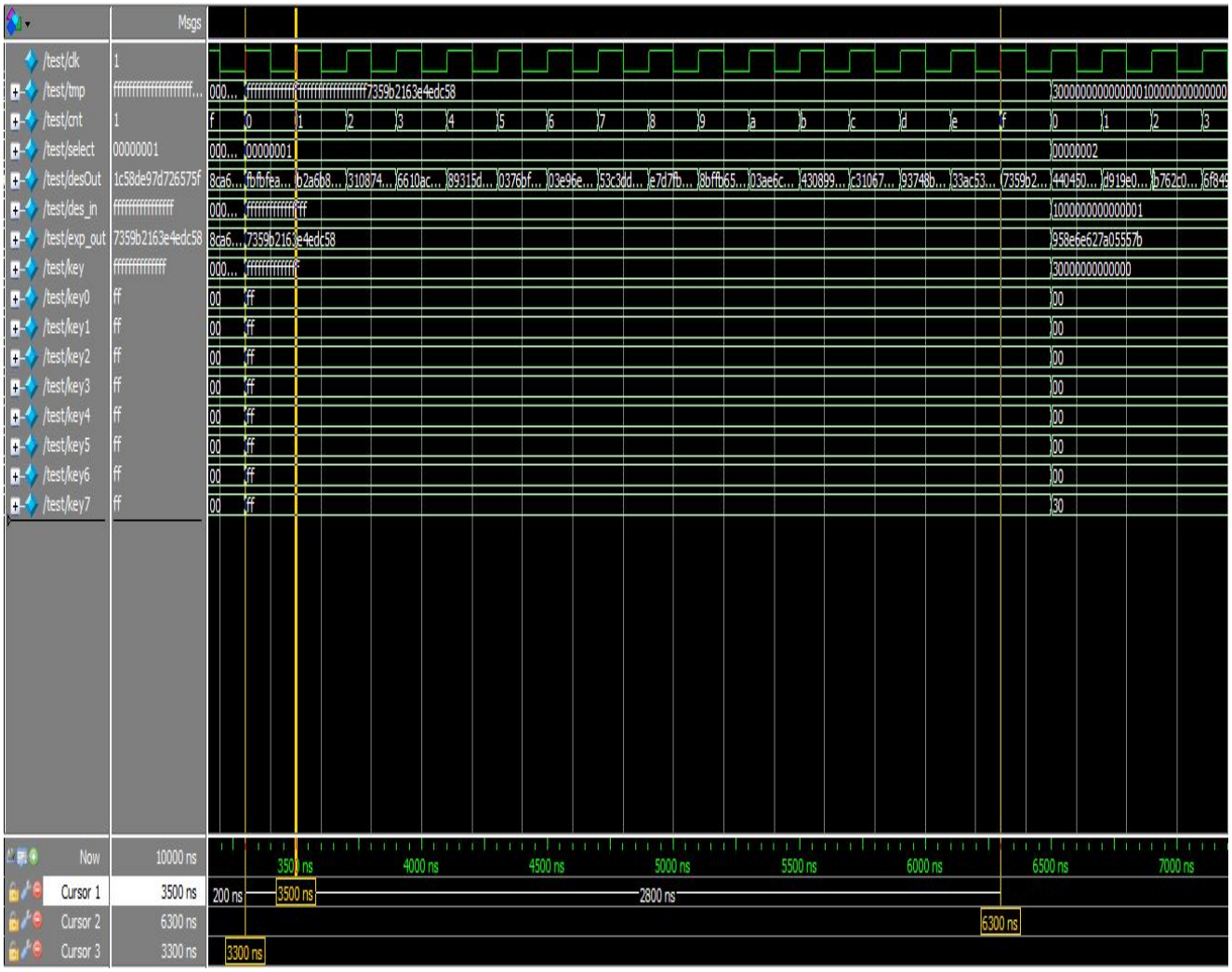


Figure 4.2: Simulation Result of implemented algorithm with 2nd input

The simulated results of the simple DES algorithm with two different 64 bit data input and a 56 bit keys have been shown in Figure 4.1 and figure 4.2. In figure 4.2, the encryption of the 2nd input data has been described. The key used to encrypt data again has 56 bits and the output data is received at the data out pin after 15 clock pulses. The encrypted data has been received at 6300 ns when the input data was given at 3300 ns. The clock cycle has been observed to be 200 ns in this case also.

Simulation results for the small DES algorithm with 2 valid data outputs

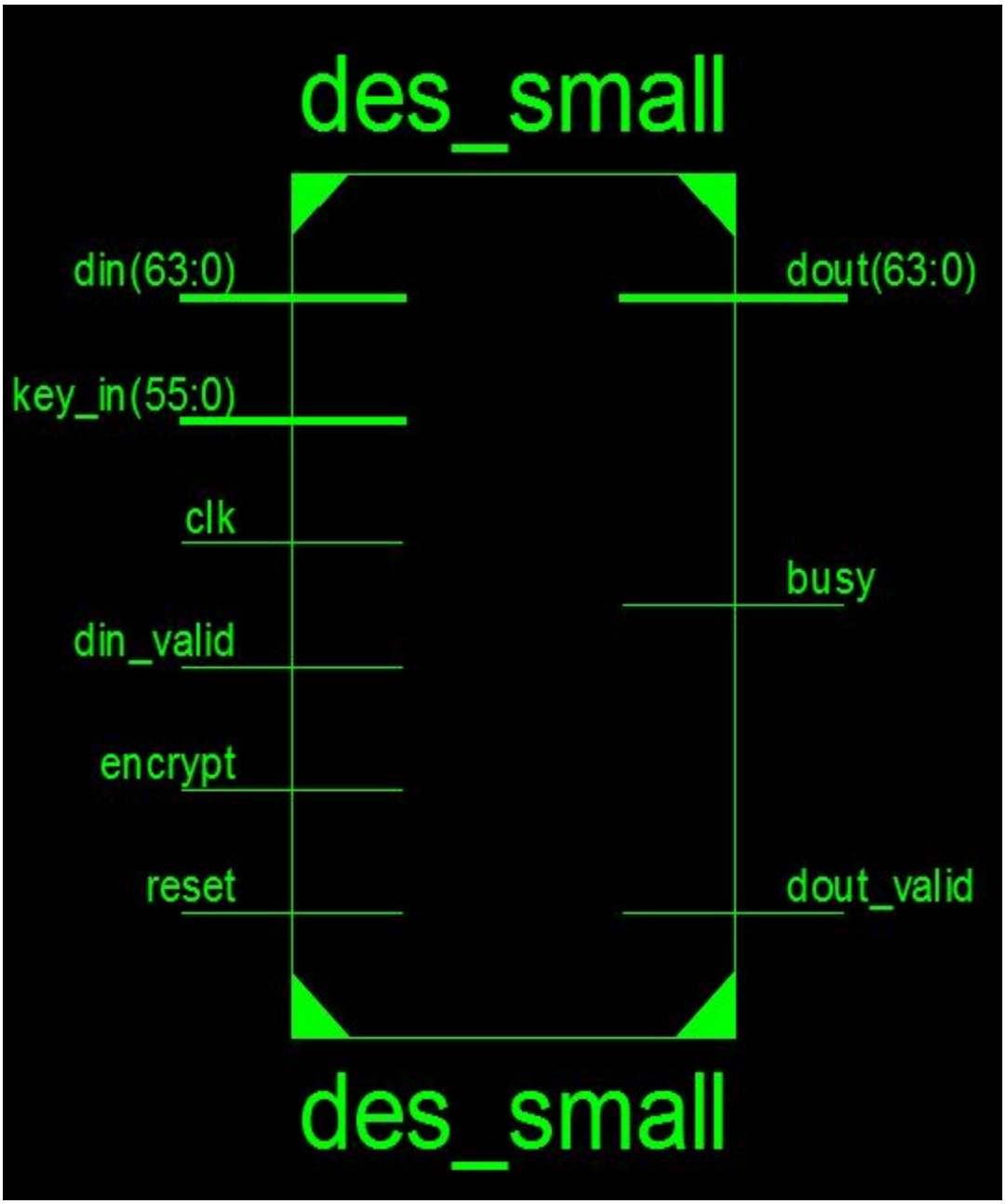


Figure 4.3: Synthesized module of DES technique

The module of the synthesized DES encryption technique is shown in figure 4.3. It has 64 bit input pin labeled as din and 64 bit output pin labeled as dout. The 56 bit key has been given through key_in port and the clock has been given through clk input port. Also, two valid pins have been provided which depicts the valid input data (din_valid) and valid output data dout_valid).

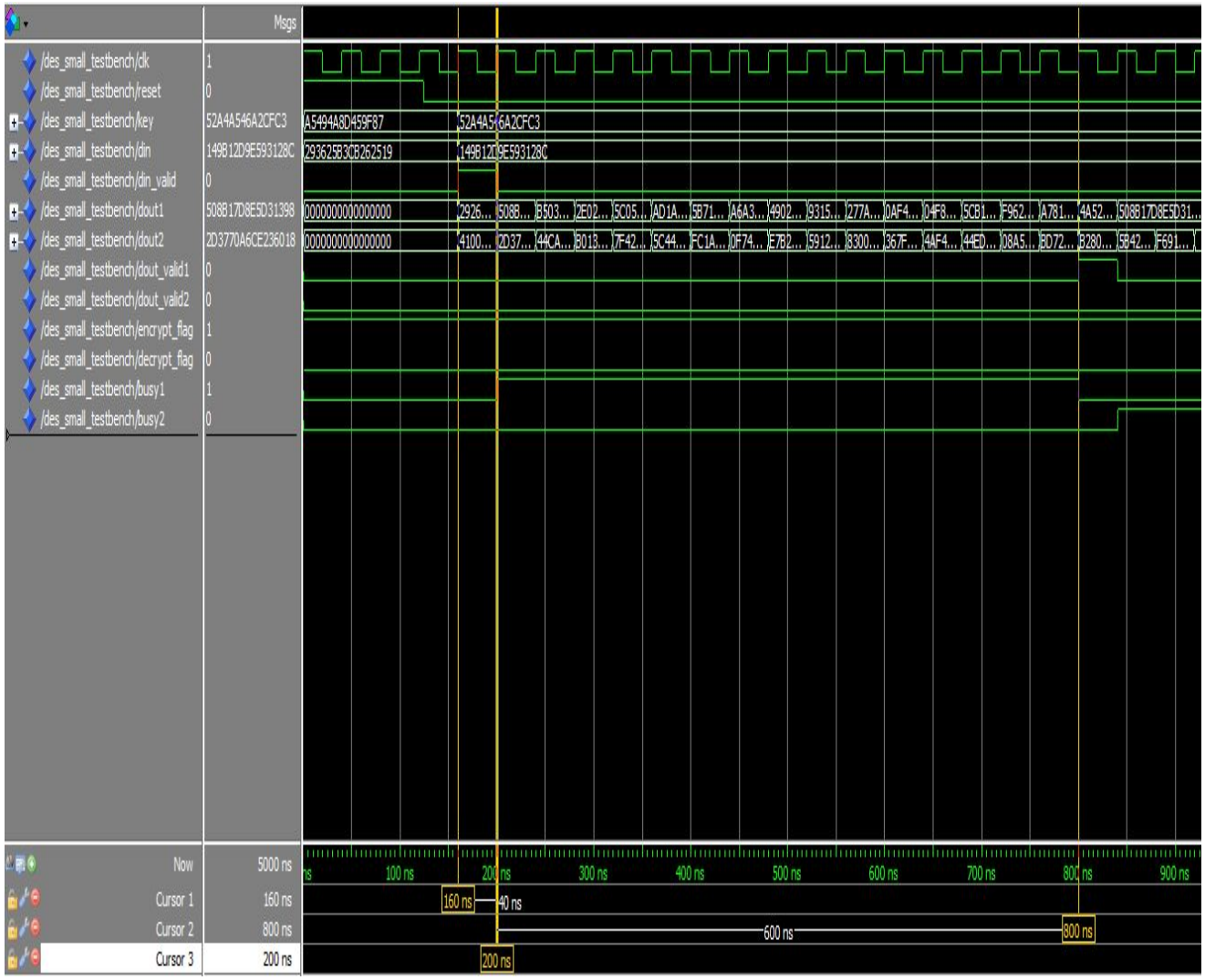


Figure 4.4: Simulation results of DES with 1st input

The DES algorithm for a 56 bit key and 64 bit data has been simulated and analyzed for two different data inputs. The encryption has been achieved with 56 bit key and valid pin has been provided for signaling the valid input and valid output data. In figure 4.4, at 160 ns the din valid pin goes high and data is latched at the input and the encrypted output has been received at 800 ns after 16 clock pulses. Each clock pulse in this implementation has been observed to be 40 ns.

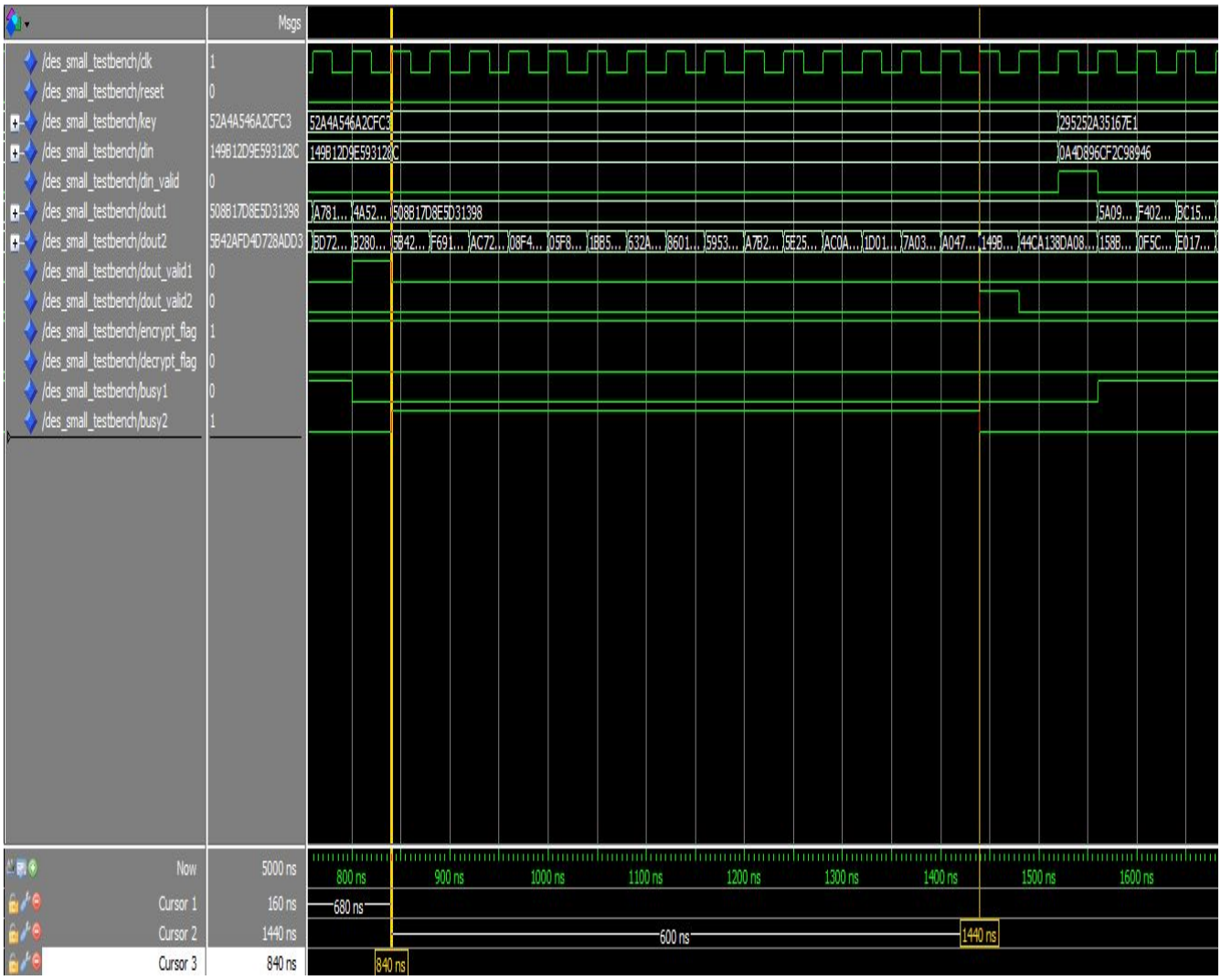


Figure 4.5: Simulation result of DES with 2nd input

The DES algorithm for a 56 bit key and 64 bit data has been simulated and analyzed for two different data inputs. The encryption has been achieved with 56 bit key and valid pin has been provided for signaling the valid input and valid output data. The result for the second input sequence has been shown in figure 4.5 where the valid pin again goes high at 800 ns and the output is received at 1440 ns where dout_valid2 has been observed to be high. The clock period was same as that in case of the 1st input pattern.

Synthesis Report of small DES

Component Name	**Number of components used**
ROMs	9
16x10-bit ROM	1
64x4-bit ROM	8

Counters	1
4-Bit Up Counter	1
Registers	182
Flip-Flops	182
Xors	2
32-Bit Xor	1
48-Bit Xor	1

Table 4.1: Synthesis report showing number of components used

The synthesis report for the implemented DES technique has been given in table 4.1. It can be seen that 182 registers have been utilized for the implementation of the encryption technique. One 4-bit up counter has been used along with one 32 bit xor and one 48 bit xor. It has been observed that the Read Only Memories (ROM) utilized for the implementation was 9 which contained one 16x10-bit ROM and eight 64x4-bit ROM.

Timing Summary of implemented small DES

Parameter	**Value**
Minimum Period	8.027 ns
Maximum Frequency	124.573 MHz
Minimum Input Arrival Time before Clock (Setup Time)	8.430 ns
Maximum Output Time After Clock (Hold Time)	4.694 ns

Table 4.2: Timing Report of small DES technique

Simulation results of DES algorithm implemented for fast response

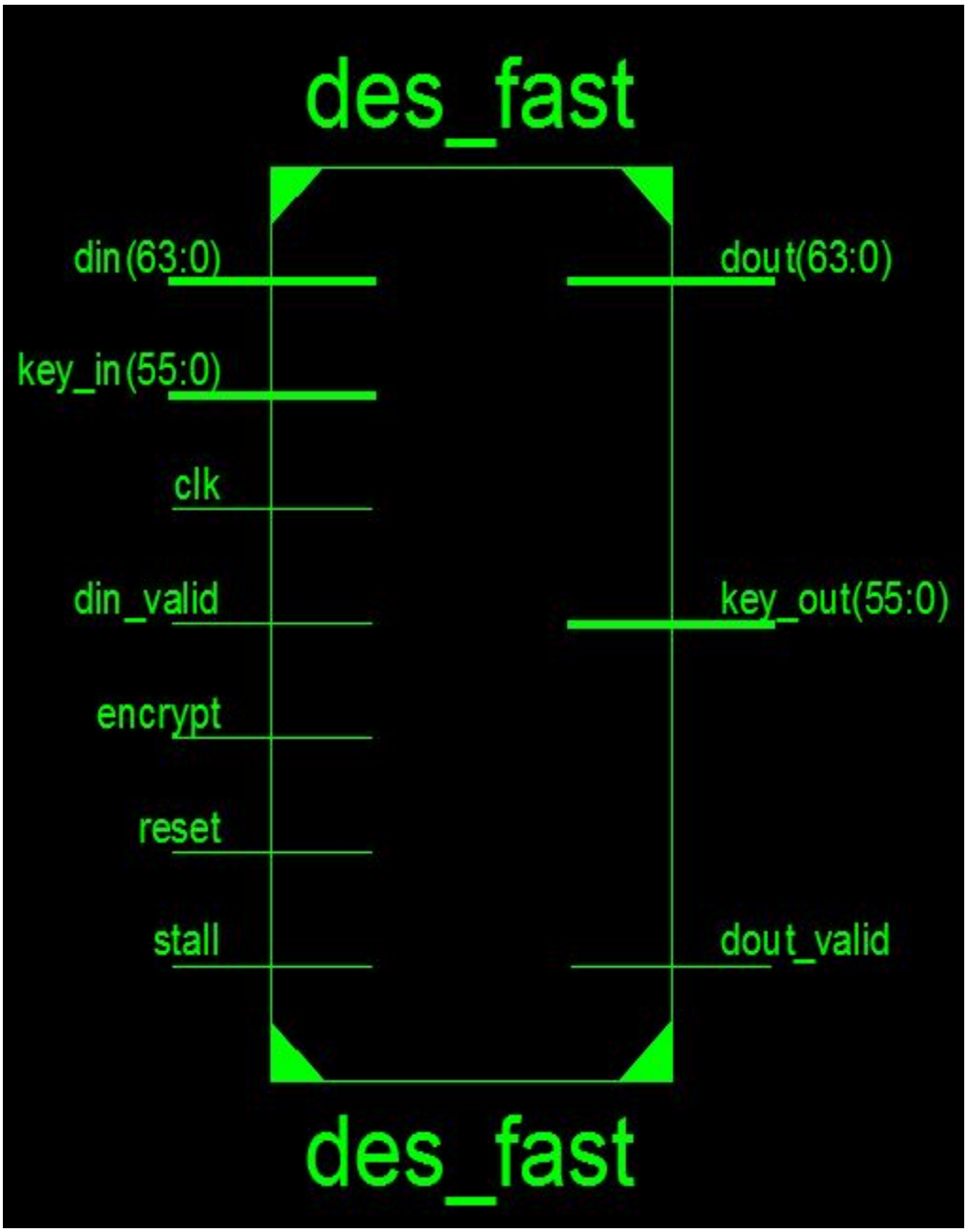


Figure 4.6: Pin Configuration of implemented hardware

The above pin configuration is of the hardware implementing DES algorithm for faster response. The input and output pins are of 64 bits with the key length used to be 56 bits. Valid pins are provided for depicting valid input and output data. The clock input is given through clk pin.

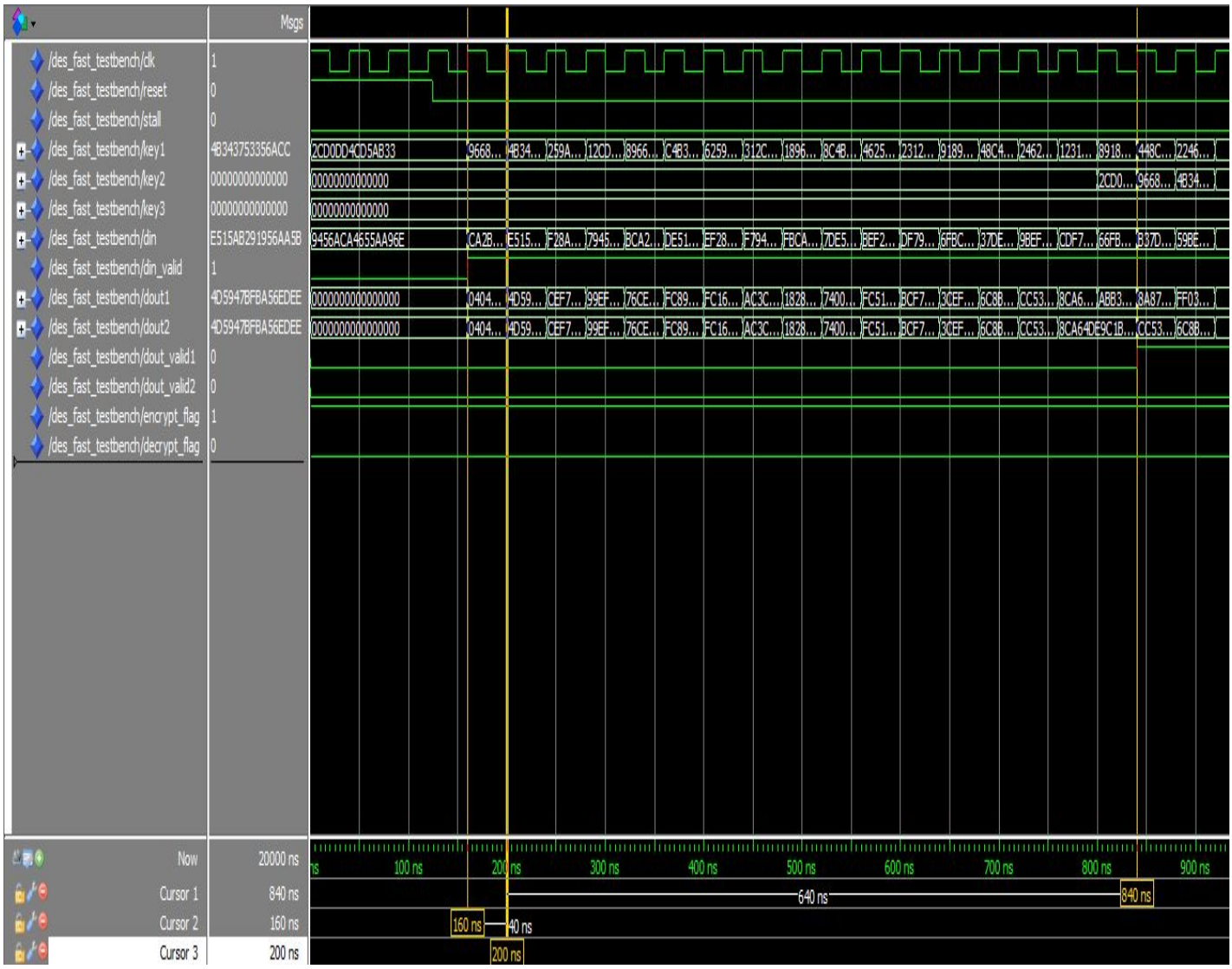


Figure 4.7: Simulated result of DES implementation with 1st key

In figure 4.7, the simulated result for the module described in figure 4.6 using the key 1. The input data became valid at 160 ns and valid output data was received at 840 ns. The clock period is of 40 ns and it took 17 clock cycles to give the valid data output. The radix of input and output data is in hexadecimal.

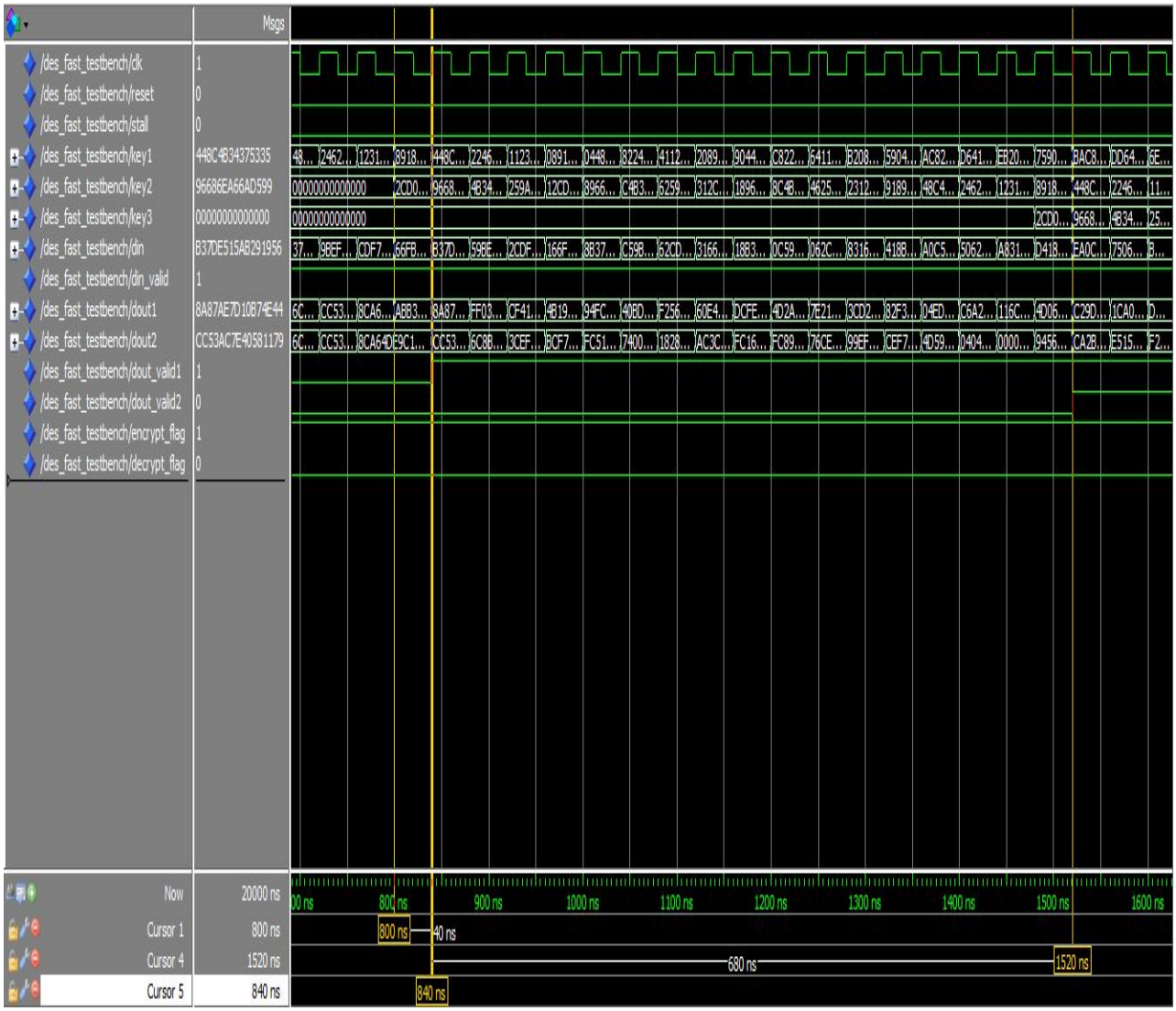


Figure 4.8: Simulated result of DES implementation with 2nd key

Figure 4.8 shows the encryption achieved with the 2nd key for the faster implementation of DES algorithm. At 1520 ns, the dout2_valid pin goes high and the output after 2nd encryption is obtained. The input data remained valid for whole duration and the output data after 1st encryption became valid at 840 ns and the data after 2nd encryption became valid at 1520 ns. With the usage of two keys for encrypting the data, the security has been increased and also the faster response has been observed.

Synthesis Report of DES

Component Name	Number of components used
ROMs	128

16x10-bit ROM	1
Registers	2073
Flip-Flops	2073
Shift Registers	1
16-bit Shift Registers	1
Xors	32
32-Bit Xor-2	16
48-Bit Xor-2	16

Table 4.3: Synthesis report showing number of components used in fast DES

Timing Summary of implemented DES

Parameter	**Value**
Minimum Period	4.067 ns
Maximum Frequency	245.866 MHz
Minimum Input Arrival Time before Clock (Setup Time)	5.569 ns
Maximum Output Time After Clock (Hold Time)	4.677 ns

Table 4.4: Timing Report of fast DES technique

From the synthesis report it has been clear that there is an increase in the number of flip flops used but there has been no counters utilized in this implementation. From the timing report the setup and hold time has been observed which has been found to be lesser than other implementations.

Observations drawn from the results shown in previous sections

From the table 4.2 and table 4.4, it has been seen that the maximum operating frequency for the fast DES implementation is higher than other implementation. The setup time and the hold time have also been found lesser than the previous implementation. From the results it has been clear that the implementations vary in terms of number of components used for their implementations. It has been observed that the faster implementation require more number of flip flops as the security is enhanced and the frequency of operation has also been increased. Only 19% of the

available flip flops have been utilized in the faster implementation and only 1% of the available FPGA flip flops have been utilized for the small DES implementation.

Chapter 5 Conclusions and Future Scope

The information security can easily be achieved by cryptography algorithm techniques a large number of encryption algorithm have been developed for securing confidential data from the cyberpunks. The aim of current Cryptography is to prevent data from hackers. The strength of the system is dependent on the length of the key. But to achieve this a large computational time is required, giving a large delay which can be harmful to us. The use of FPGAs can help us to improve this limitation because FPGAs can give enhanced speed. This is due to fact that the hardware implementation of most encryption algorithms can be done on FPGA.

The proposed scheme for DES algorithm has been optimised on the time required to generate the keys or decode data. The algorithm and coding has been implemented on Model-Sim software with the help of VHDL language. The synthesis has been done on Xilinx FPGA (Xilinx 9.1e) and the faster clock frequency has been observed in comparison with classical DES.

It has been observed that it takes 19 nanoseconds for the input data of size 8 bytes while the technique by which the algorithm has been implemented in this thesis using VHDL has reduced the time to 16 nanoseconds. The improvement has been observed to be 15.78% as compared to classical DES technique.

The work has been extended in order to increase the security for more severe attacks since in the thesis only the encryption time has been reduced. The complexity and severity of attacks need a lot of theoretical calculations. There has been seen the scope to further optimize the utilization of resources. The implementation has been further improved so as to get the more efficient usage of the resources and increase in the maximum clock frequency. The key length can be reduced, keeping the same security, in order to optimize the utilization of resources. The few gaps have been covered but still a lot of work can be done for the increase in security of the data along with the optimization of resources.

List of Publications

- Gaurav Sharma and Ajay Kakkar, “Cryptography Algorithms and Approaches used for Data Security”, International Journal of Scientific & Engineering Research, 2012, Vol. 3, No. 6, pp. 1-6.
- Gaurav Sharma and Ajay Kakkar, "Cryptography Algorithms and Approaches used for DES algorithms", International Conference on Advancements in Computing and Communication, Vol. 2, 2012, pp. 427-432.

References

1. D. Kahn: The Code breakers: the story of secret writing, MacMillan publishing, 1996.
2. W. Diffie and M. Hellman, "New Directions in Cryptography", IEEE Transaction on Information Theory, Vol. IT-22**,** 1976, pp. 644-654**.**
3. Ruth M. Davis, "The Data Encryption Standard", Proceedings of Conference on Computer Security and the Data Encryption Standard, National Bureau of Standards, Gaithersburg, MD, 1977, NBS Special Publication 500-527, pp 5-9.
4. Whitfleld Diffie, "Cryptographic Technology: Fifteen Year Forecast" Reprinted by permission AAAS, 1981 from Secure Communications and Asymmetric Crypto Systems. AAAS Selecte8 Symposia. Editor: C.J. Simmons. Vol. 69, West view Press, Boulder, Colorado, pp 38-57.
5. Ingrid Verbauwhede, "Security and Performance Optimization of a New DES Data Encryption Chip", IEEE journal of Solid-State Circuits, Vol. 23, No. 3.1988, pp 647-656.
6. James E. Katz, "Social Aspects of Telecommunications Security Policy", IEEE journal Technology and Society Magazine, 1990, pp 16-24.
7. H. Bonnenbergt, "VLSI Implementation of a New Block Cipher", IEEE journal on Information Theory 1991, pp 510-513.
8. K.H. Mundt, "SUPERCRYPT, ASIC Technology facilitates a new Device Family for Data Encryption", IEEE journal on cloud computing 1992, pp 356-359.
9. C. Boyd. "Modern Data Encryption," Electronics & Communication Engineering Journal on data security and neural networks 1993, Vol. 5, pp 271-278.
10. R. Zimmermann, "A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm", IEEE Journal of Solid-State Circuits. Vol. 29, No. 3, 1994, pp 303-307.
11. Stefan Wolter "On the VLSI Implementation of the International Data encryption Algorithm IDEA", IEEE journal on computer system and data security 1995, pp 397-400.
12. Seung-Jo Han, "The Improved Data Encryption Standard (DES) Algorithm" IEEE journal on information system 1996, Vol. 3, pp 1310-1314.
13. Hassina Guendouz, "Rapid Prototype of a Fast Data Encryption Standard with Integrity Processing for Cryptographic Applications", IEEE transaction on data originations 1998, pp 434-437.

14. K. Wong, “A Single-Chip FPGA Implementation of the Data Encryption Standard (DES) Algorithm” Global Telecommunications Conference, 1998. GLOBECOM 98, IEEE, Vol. 2, pp. 827-832.
15. M.P. Leong, “A Bit-Serial Implementation of the International Data Encryption Algorithm IDEA”, 2000 IEEE conference Symposium on Field-Programmable Custom Computing Machines, pp 122-131.
16. R. G. Sixel, “A High Level Language Implementation of the Data Encryption Standard and a Bit-Slice Architecture”, Roc 43rd IEEE Midwest Symposium on Circuits and Systems, Lansing MI, 2000, pp 266-269.
17. Teo Pock Chueng, “Implementation of Pipelined Data Encryption Standard (DES) Using Altera CPLD”, TENCON 2000 Proceedings, Vol. 3, IEEE 2000, pp 17-21.
18. Yeong-Kang Lai, “A Novel VLSI Architecture for a Variable-Length Key, 64-Bit Blowfish Block Cipher”, Signal Processing Systems, 1999 IEEE Workshop, pp 568-577.
19. Toby Schaffer, “A Flip-Chip Implementation of the Data Encryption Standard (DES)”, IEEE1997, pp 13-17.
20. Cameron Patterson, “High Performance DES Encryption in Vertex FPGAs using Jbits”, IEEE journal Symposium on FPGA 2000, pp 113-121.
21. Jingmei liu, “Improved DES Algorithm based on Irrational Numbers”, IEEE Int. Conference Neural Networks & Signal Processing Zhenjiang, China, 2008, pp 632-635.
22. Pui-Lam Siu, “A Low Power Asynchronous DES”, Circuits and Systems, ISCAS 2001 IEEE International conference Symposium, Vol. 4, pp 538-541.
23. N. Sklavos, “Asynchronous Low Power VLSI Implementation of the International Data Encryption Algorithm”, Electronics Circuits and Systems ICECS 2001, 8th IEEE International Conference Vol. 3, pp 1425-1428.
24. Ahmet Eskicioglu, “Cryptography”, IEEE journal on Symposium and Potentials 2001, pp 36-38.
25. Dragon Jankovi, “An Efficient and Scalable VLSI Implementation of DES”, ASIC 2001 Proceedings 4th International Conference IEEE on Symposium 2001, pp 341-343.
26. Massimo Aloito, “Hardware implementations of the Data Encryption Standard”, IEEE journal on neural networks 2002, pp 100-103.

27. Chih-Chung Lu, “Integrated Design of AES (Advanced Encryption Standard) Encryptor and Decrypted”, Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP’02).
28. G. Catalini, “Modified Two fish Algorithm for increasing Security and Efficiency in the Encryption of Video signals”, IEEE 2003, pp 525-528.
29. Aamer Nadeem, “A Performance Comparison of Data Encryption Algorithms”, IEEE on Application-Specific Systems, Architectures, and Processors 2005, pp 84-89.
30. M. McLoone, “High-performance FPGA implementation of DES using a novel method for implementing the key schedule”, IEEE Proc.-Circuits Devices Syst., Vol. 150, No. 5, pp 373-378.
31. William Stallings, “Cryptography and Network Security Principles and Practices”, Fourth Edition, 2005, Prentice Hall.
32. Andrew S. Tanenbaum: “Computer Networks” by Prentice Hall.
33. Jerome Burke John McDonald Todd Austin “Architectural Support for Fast Symmetric-Key Cryptography” Advanced Computer Architecture Laboratory
34. Alan G. Konheim. "Cryptography: A Primer", John Wiley & Sons.
35. Dominic Welsh, "Codes and Cryptography", Oxford University Press.
36. Jari Nurmi, "Processor Design: System-On-Chip Computing for ASICs and FPGAs", Springer.
37. Pong P. Chu "FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version", Wiley Interscience.

Appendix

library ieee;

```
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

package destest_lib is

  component des_fast_testbench
  end component;

  component des_small_testbench
  end component;

  function random32 (din:std_logic_vector(31 downto 0))
                return std_logic_vector;
  function random56 (din:std_logic_vector(55 downto 0))
                return std_logic_vector;
  function random64 (din:std_logic_vector(63 downto 0))
                return std_logic_vector;
end destest_lib;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
library work;
use work.destest_lib.all;
package body destest_lib is
  function random32 (din:std_logic_vector(31 downto 0))
return std_logic_vector is
    variable val :std_logic_vector (31 downto 0);
  begin
```

```
  val := (din(31) xor din(6) xor din(4) xor din(2) xor din(1)) & din(31 downto 1);
  return (val);
 end random32;

 function random56 (din:std_logic_vector(55 downto 0))
                return std_logic_vector is
  variable val :std_logic_vector (55 downto 0);
 begin
  val := (din(55) xor din(6) xor din(3) xor din(1)) & din(55 downto 1);
  return (val);
 end random56;

 function random64 (din:std_logic_vector(63 downto 0))
                return std_logic_vector is
  variable val :std_logic_vector (63 downto 0);
 begin
  val := (din(63) xor din(3) xor din(2) xor din(0)) & din(63 downto 1);
  return (val);
 end random64;
end destest_lib;
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
library work;
use work.des_lib.all;
use work.destest_lib.all;

entity des_fast_testbench is
end des_fast_testbench;
architecture arch_des_fast_testbench of des_fast_testbench is
```

```
  signal clk              :std_logic := '1';
  signal reset            :std_logic := '1';
  signal stall            :std_logic := '0';
  signal key1             :std_logic_vector          (55          downto          0)
"00000000000000000000000000000000000000000000000000000000";
  signal key2             :std_logic_vector          (55          downto          0)
"00000000000000000000000000000000000000000000000000000000";
  signal key3             :std_logic_vector          (55          downto          0)
"00000000000000000000000000000000000000000000000000000000";
  signal din              :std_logic_vector          (63          downto          0)
"0000000000000000000000000000000000000000000000000000000000000000";
  signal din_valid        :std_logic := '0';
  signal dout1            :std_logic_vector          (63          downto          0)
"0000000000000000000000000000000000000000000000000000000000000000";
  signal dout2            :std_logic_vector          (63          downto          0)
"0000000000000000000000000000000000000000000000000000000000000000";
  signal dout_valid1      :std_logic := '0';
  signal dout_valid2      :std_logic := '0';
  signal encrypt_flag     :std_logic := '1';
  signal decrypt_flag     :std_logic := '0';
begin
  process (clk)
  begin
    if clk='1' then
      clk <= '0' after 20ns, '1' after 40ns;
    end if;
  end process;

  reset <= '1' after 0ns, '0' after 125ns;

  -- Generate the random key's and random data
```

```
  process (reset, clk)
  begin
    if reset='1' then
      din_valid <= '0';
      din <= "1001010001010110101011001010010001100101010110101010100101101110";
      key1 <= "0010110011010000110111010100110011010101101010110011001l";
    elsif clk'event and clk='1' then
      key1 <= random56(key1);
      din  <= random64(din);
      din_valid <= '1';
    end if;
  end process;


  -- Do the DES encryption/decryption blocks
  encrypt_flag <= '1';
  decrypt_flag <= '0';
     DES0:   des_fast port map (clk, reset, stall, encrypt_flag, key1, din, din_valid, dout1,
dout_valid1, key2);
    DES1:   des_fast port map (clk, reset, stall, decrypt_flag, key2, dout1, dout_valid1, dout2,
dout_valid2, key3);


  -- Verify the unencrypted output
  process (reset, clk, dout2, dout_valid2)
    variable check  :std_logic_vector (63 downto 0);
  begin
    if reset='1' then
      check := "1001010001010110101011001010010001100101010110101010100101101110";
    elsif clk'event and clk='1' then
      if dout_valid2='1' then
```

```
      check := random64(check);

      if check/=dout2 then
        assert 1=0
          report "Simulation Ended”
          severity failure;
      end if;
    end if;
  end if;
 end process;

end arch_des_fast_testbench;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
library work;
use work.des_lib.all;
use work.destest_lib.all;

entity des_small_testbench is
--  port();
end des_small_testbench;

architecture arch_des_small_testbench of des_small_testbench is
  signal clk              :std_logic := '1';
  signal reset            :std_logic := '1';
  signal key              :std_logic_vector          (55          downto          0)
"00000000000000000000000000000000000000000000000000000000";
```

```
  signal din              :std_logic_vector       (63          downto          0)
"0000000000000000000000000000000000000000000000000000000000000000";
  signal din_valid        :std_logic := '0';
  signal dout1            :std_logic_vector       (63          downto          0)
"0000000000000000000000000000000000000000000000000000000000000000";
  signal dout2            :std_logic_vector       (63          downto          0)
"0000000000000000000000000000000000000000000000000000000000000000";
  signal dout_valid1      :std_logic := '0';
  signal dout_valid2      :std_logic := '0';
  signal encrypt_flag     :std_logic := '1';
  signal decrypt_flag     :std_logic := '0';
  signal busy1            :std_logic := '1';
  signal busy2            :std_logic := '1';

begin
  process (clk)
  begin
    if clk='1' then
      clk <= '0' after 20ns, '1' after 40ns;
    end if;
  end process;

  reset <= '1' after 0ns, '0' after 125ns;

  -- Generate the random key's and random data
  process (reset, clk, busy1, busy2)
  begin
    if reset='1' then
      din_valid <= '0';
      din <= "0010100100110110001001011011001111001011001001100010010100011001";
      key <= "1010010101001001010010101000110101000101100111111000011 1";
```

```
  elsif clk'event and clk='1' then
    --if dout_valid2='1' then
    if din_valid='0' and busy1='0' and busy2='0' and dout_valid1='0' and dout_valid2='0' then
      din_valid <= '1';
      din <= random64(din);
      key <= random56(key);
    else
      din_valid <= '0';
    end if;
  end if;
end process;


-- Do the DES encryption/decryption blocks
encrypt_flag <= '1';
decrypt_flag <= '0';
 DES0:  des_small port map (clk, reset, encrypt_flag, key, din,  din_valid,  busy1, dout1, dout_valid1);
 DES1:  des_small port map (clk, reset, decrypt_flag, key, dout1, dout_valid1, busy2, dout2, dout_valid2);


-- Verify the unencrypted output
process (clk, din, dout_valid2)
  variable check  :std_logic_vector (63 downto 0);
begin
  if clk'event and clk='1' then
    if dout_valid2='1' then
      if din/=dout2 then
        assert 1=0
          report "Simulation Ended"
```

```
            severity failure;
        end if;
      end if;
    end if;
  end process;

end arch_des_small_testbench;
```