FPGA IMPLEMENTATION OF BLOWFISH CRYPTOGRAPHIC ALGORITHM

A PROJECT REPORT

Submitted by

D.DEEPAK RAJAN	71206106015
N.POONKUMARAN	71206106035
K.PRABHU	71206106036
S.VIGNESH KUMARAN	71206106058

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2010

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"FPGA IMPLEMENTATION OF BLOWFISH CRYPTOGRAPHIC ALGORITHM"** is the bonafide work of **"D.DEEPAK RAJAN, N.POONKUMARAN, K.PRABHU, S.VIGNESH KUMARAN"** who carried out the project work under my supervision.

13/4/10

SIGNATURE

Dr.RAJESWARI MARIAPPAN, Ph.D.,
HEAD OF THE DEPARTMENT

Electronics and
Communication Engineering
Kumaraguru College of Technology
Coimbatore – 641006.

13/4/2010

SIGNATURE

Ms. S.NAGARATHINAM, M.E.,
SENIOR LECTURER
SUPERVISOR

Electronics and
Communication Engineering
Kumaraguru College of Technology
Coimbatore – 641006.

The candidates with university register numbers **71206106015, 71206106035, 71206106036, 71206106058** were examined by us in the project viva-voce examination held on .....15/04/2010.....

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We wish to express sincere thanks and deep sense of gratitude to our respected Director **Dr.J.SHANMUGAM,Ph.D,** for permitting us to undertake this project.

We are greatly indebted to our beloved Principal **Dr.S.RAMACHANDRAN,Ph.D.,** who has been the backbone of all our deeds.

We profusely thank **Dr.RAJESWARI MARIAPPAN, M.E., Ph.D., B.Tech.Ed., FIE., MISTE** Head of the Department, Department of Electronics and Communication Engineering, Kumaraguru College of Technology for leading a help hand in this project.

We are highly grateful to our beloved Project Coordinator **Ms.A.VASUKI M.E., Assistant Professor**, and project guide **Ms.S.NAGARATHINAM M.E., Senior Lecturer**, ECE department for their valuable guidance, timely helps, constant encouragement and advice rendered throughout the project period for complete successful completion of the project.

We are also grateful to the faculty members of Electronics and Communication Engineering Department, who have helped us in innumerable ways.

We also thank our parents without whom we could not have come so far and friends for their timely help that culminated as good in end.

CHAPTER NO.	TITLE	PAGE NO.

ABSTRACT

Cryptography is required to protect information in wireless network, nternet etc. from being intercepted and stolen by an unwanted third party vith the use of keys. The project provides a simple, robust implementation of Blowfish Cryptographic Algorithm in hardware. A hardware mplementation of Blowfish would be a powerful tool for any mobile device or any technology requiring strong encryption.

The Blowfish algorithm is conceptually simple, but its actual mplementation and use is complex. Blowfish has a fixed 64-bit block size. The cipher is a 16-round Feistel network which utilizes a structure which makes encryption and decryption very similar. Blowfish is among the fastest block ciphers available. The speed can be further increased at the expense of space and power by pipelining.

The algorithm has been implemented in Spartan 3E FPGA using Xilinx platform studio and ISE tools.

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

1.1 Cryptography

The art of science that conveys message from source to destination in a secured basis is Cryptography. There are two kinds of cryptosystems: symmetric and asymmetric. Symmetric cryptosystems use the same key (the secret key) to encrypt and decrypt a message, and asymmetric cryptosystems use one key (the public key) to encrypt a message and a different key (the private key) to decrypt it. Asymmetric cryptosystems are also called public key cryptosystems.

Need for security

Steps involved in secured communication:

1. Design an algorithm for performing the security related transformation such that the opponent cannot defeat its purpose.
2. Generate the secret information to be used with the algorithm.
3. Specify the protocol to be used by the two principles that make use of the security algorithm.

Threats in communication

Information access threat:

Modification of data without the knowledge of sender and then transmit the data.

Service threat:

Exploit the flaws in the services available in computer to inhibit the use by legitimate users.

Types of intruders

Masquerader: An unauthorized user, using a system's control to exploit legitimate user's account.

Misfeaser: A legitimate user accessing data, programs or resources for which access isn't authorized.

1.2 SYMMETRIC CIPHER MODEL

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the development of public-key encryption. The most widely used symmetric cipher is TDES

Plaintext: original message or data fed into the algorithm as input.

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

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

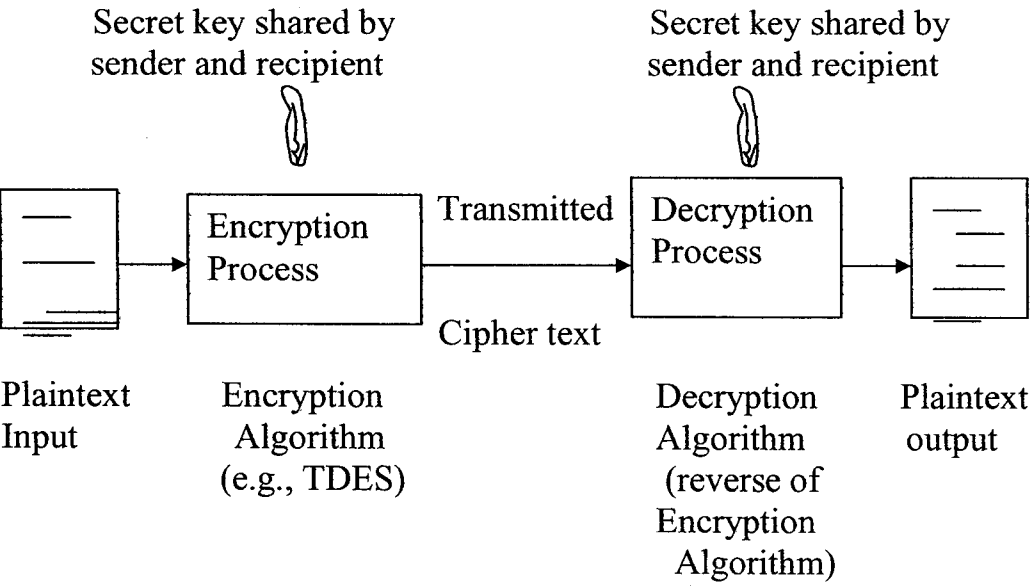


Fig-1.1: Simplified Model of Conventional Encryption

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

Decryption algorithm: The reverse process of encryption algorithm. It takes the ciphertext and secret keys and produces the original plaintext.

MODEL OF CONVENTIONAL CRYPTOSYSTEM

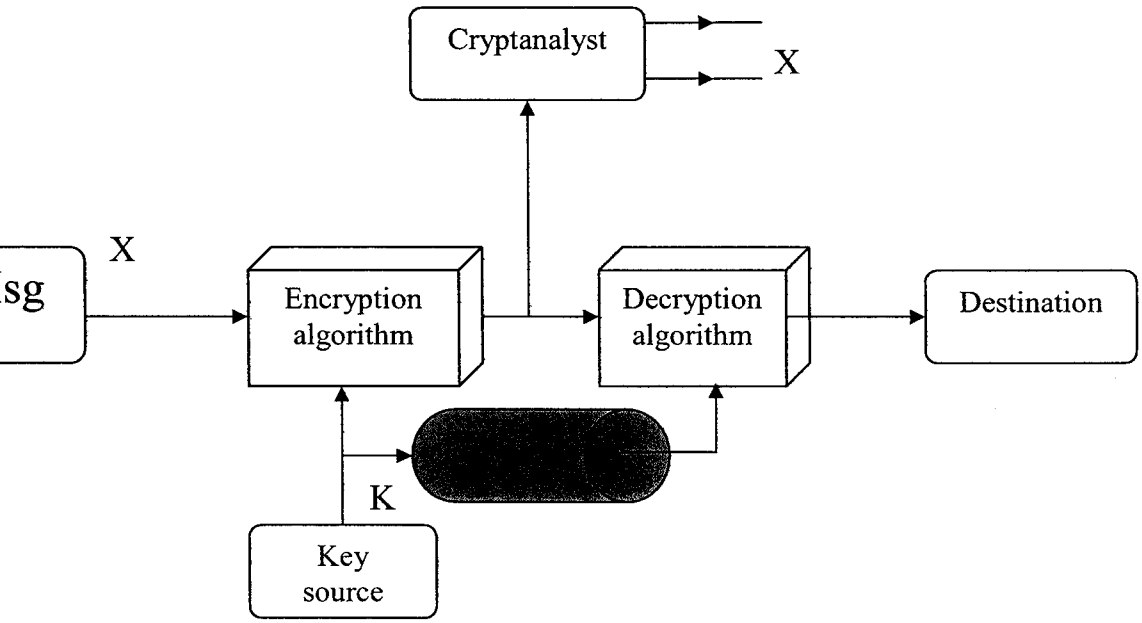


Fig-1.2: Model of conventional cryptosystem

1.3 Model of conventional cryptosystem

A source produces a message in plaintext, X=[X1, X2, X3, ... , XM].The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet {0,1} is typically used. For encryption, a key of the form K=[K1, K2, K3, ..., KJ] is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext Y=[Y1, Y2, Y3, ... , YN].

$$Y=EK(X)$$

The above notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K. The intended receiver, in possession of the key, is able to invert the transformation:

$$X=DK(Y)$$

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms. If the opponent is interested in only this particular message, then focus of the effort is to recover X by generating a plaintext estimate X. Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate K.

1.4 FUNDAMENTALS OF CRYPTOGARPHY

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient. Messages transmitted across the

nternet are susceptible to eavesdropping attacks via any path along the ransmission of a message.

Cryptography is required to protect information from being ntercepted and stolen by an unwanted third party. While cryptography is the cience of securing data, cryptanalysis is the science of analyzing and reaking secure communication. Classical cryptanalysis involves an nteresting combination of analytical reasoning, application of mathematical ools, pattern finding, patience, determination, and luck. Cryptanalysts are lso called attackers. Cryptology embraces both cryptography and ryptanalysis.

.4.1 Symmetrical Cryptography

In Symmetrical cryptography, also called secret-key or conventional-ey encryption, one key is used both for encryption and decryption. The Data Encryption Standard (DES) is an example of a conventional ryptosystem that is widely employed by the Federal Government.

Conventional encryption is very fast. However, conventional ncryption alone as a means for transmitting secure data can be quite xpensive simply due to the difficulty of secure key distribution. Anyone vho overhears or intercepts the key in transit can later read, modify, and orge all information encrypted or authenticated with that key. The problems f key distribution are solved by public key cryptography.

.4.2 Asymmetric or Public Key cryptography

Public key cryptography is an asymmetric scheme that uses a pair of keys for encryption: as shown in Figure 1-1 public key, which encrypts data, and a corresponding private, or secret key for decryption. The public key is published to the world while private key is kept secret.

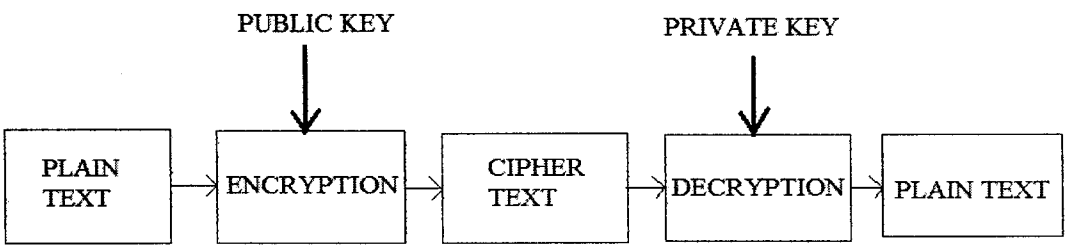


Fig 1.3: Public key cryptography

The deduction of the private key from the public key is computationally infeasible. A user with a public key can encrypt information but cannot decrypt it. Only the user with the corresponding private key can decrypt the information.

.5 CRYPTOGRAPHIC ALGORITHMS:

'here are various cryptographic algorithms. Examples of some ryptographic algorithm are as follows,

.5.1 RSA(Rivest-Shamir-Adelman)

RSA is a public-key cryptosystem developed by MIT professors Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman in 1977, in an ffort to help ensure Internet security. RSA uses modular arithmetic and lementary number theory as the basis for encryption computation. The RSA s a solid algorithm that produces strong cipher text. It has been broken, sing brute force and other crypto-analysis techniques. RSA has never been roven to be secure but it is based on the difficulty of factoring large prime umbers. Dramatic advances in factoring large integers would make RSA ulnerable.

.5.2 DES(Data Encryption Standard)

The DES uses the DEA algorithm and executes very quickly. It was he first official U.S. government cipher intended for commercial use and vas the most widely used cryptosystem in the world. The DES can also be sed for single-user encryption, such as to store files on a hard disk in an ncrypted form. In a multi-user environment, secure key distribution may be ifficult; public-key cryptography provides an ideal solution to this problem.

DES is the most widely used symmetric algorithm in the world, espite claims that the key length is too short. Ever since DES was first

announced, controversy has raged about whether 56 bits is long enough to guarantee security. DES is relatively small 56-bit key which was becoming vulnerable to brute force attacks. DES was designed for hardware and is relatively slow when implemented in software.

1.5.3 Triple DES:

Triple DES avoids the problem of a small key size. But it is also very slow software and is unsuitable for limited resource platform. Triple DES Algorithm is a block cipher that transforms 64 bit data blocks under three 56 bit secret keys, by means of permutation and substitution. Triple DES Algorithm uses three consecutive DES algorithm, therefore security level of the coding is drastically increased.

The cryptography DES Algorithm transforms a 64 bit binary value into a unique 64 bit binary value based on a 56-bit variable.

1.5.4 AES (Advanced Encryption Standard)

In November 2001, AES designated Rijndael algorithm as the standard algorithm. Therefore, the terms AES algorithm and Rijndael algorithm are used interchangeably. There are several algorithms proposed for AES, including RC6, skipjack etc. However, Rijndael is selected because of its flexibility and simplicity. Rijndael is a symmetric block cipher with a block size of 128 bits. The AES specifies the algorithm to support variable key sizes: 128, 192, and 256 bits.

1.5.5 ORIGIN OF BLOWFISH ALGORITHM:

Blowfish is a keyed, symmetric block cipher, designed in 1993 by Bruce Schneier and included in a large number of cipher suites and encryption products. No effective cryptanalysis of Blowfish has been found to date.

Schneier designed Blowfish as a general-purpose algorithm, intended as a replacement for the aging DES and free of the problems associated with other algorithms. At the time, many other designs were proprietary, encumbered by patents or kept as government secrets. Schneier has stated that, "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone." Notable features of the design include key-dependent S-boxes and a highly complex key schedule.

1.6 FUNDAMENTALS OF BLOWFISH CRYPTOGRAPHIC ALGORITHM:

The Blowfish encryption scheme was designed by Bruce Schneier in 1993 to replace Data Encryption Standard (DES), which was the Federal Information Processing Standard Cryptography (FIPS Crypto). The intent was to create a cryptographic algorithm which did not possess the limitations and issues common in other crypto algorithms and to provide an open, readily available crypto for users rather than the common patented or classified crypto algorithms being used contemporaneously. Due to its standing as a crypto algorithm, blowfish is now part of the Linux kernel.

Blowfish continues to attain its lofty goals of secure, open encryption that is realizable in software and hardware. The Blowfish algorithm is conceptually simple, but its actual implementation and use is complex. Blowfish has a fixed 64-bit block size. The key length of Blowfish is anywhere upto 448 bits.

The cipher is a 16-round Feistel network and uses password-dependent S-boxes. A Feistel network is one that utilizes a structure which makes encryption and decryption very similar through the use of the following elements:

- Pboxes (permutation boxes; these perform bit shuffling)
- S-boxes (substitution boxes, simple nonlinear functions)
- XORing to achieve Linear Mixing

Feistel ciphers are a special class of iterated block ciphers where the cipher text is calculated from the plaintext by repeated application of the same transformation or round function.

Blowfish encapsulates all these elements into an efficient and powerful algorithm. The action of Blowfish can be seen below.

Tab: 1.1 Speed comparisons of various block ciphers **(Bruce Schneier 1996)**

Speed Comparisons of Block Ciphers on a Pentium				
Algorithm	**Clock cycles per round**	**# of rounds**	**# of clock cycles per byte encrypted**	**Notes**
Blowfish	9	16	18	Free, unpatented
Khufu/Khafre	5	32	20	Patented by Xerox
RC5	12	16	23	Patented by RSA Data Security
DES	18	16	45	56-bit key
IDEA	50	8	50	patented by Ascom-Systec
Triple-DES	18	48	108	

Blowfish has been also identified as a powerful cryptographic algorithm since it can satisfy two basic requirements: high immunity to attacks and relative low algorithm complexity. These two characteristics are essential for implementation of robust and fast electronic cryptographic systems. Although a complex initialization phase is required before any encryption can take place, the actual encryption of data is very efficient on hardware coprocessor devices. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

The algorithm consists of two parts: a key expansion part and a data encryption part. Key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes. It is suitable for applications where the key does not often change.

CHAPTER – 2

DESCRIPTION OF BLOWFISH ALGORITHM

2.1Blowfish Algorithm

Blowfish a variable-length key, 64-bit block cipher consists of two parts

- A key-expansion part.
- A data- encryption part.

Key expansion converts a key of at most 448 bits into several sub-key arrays totaling 4168 bytes.

Data encryption occurs via a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Blowfish in practice

Blowfish is one of the fastest block ciphers in widespread use, except when changing keys. Each new key requires pre-processing equivalent to encrypting about 4 kilobytes of text, which is very slow compared to other block ciphers. This prevents its use in certain applications, but is not a problem in others. The idea is that the extra computational effort required gives protection against dictionary attacks. In some implementations, Blowfish has a relatively large memory footprint of just over 4 kilobytes of RAM. This is not a problem even for older smaller desktop and laptop computers, but it does prevent use in the smallest embedded systems such as early smart cards.

2.2 Sub-keys

Blowfish uses a large number of sub-keys. These keys must be precomputed before any data encryption or decryption.

1.The P-array consists of 18 32-bit sub-keys: P1, P2,..., P18.

2. There are four 32-bit S-boxes with 256 entries each:

S1,0, S1,1,..., S1,255;

S2,0, S2,1,..,, S2,255;

S3,0, S3,1,..., S3,255;

S4,0, S4,1,..,, S4,255.

The exact method used to calculate these sub-keys will be described later.

2.3 Encryption

Blowfish is a Feistel network consisting of 16 rounds (Fig 3.2). The input is a 64-bit data element, x.

Divide x into two 32-bit halves: xL, xR

For i = 1 to 16:

xL = xL XOR Pi

xR = F(xL) XOR xR

Swap xL and xR

Swap xL and xR (Undo the last swap.)

xR = xR XOR P17

xL = xL XOR P18

Recombine xL and xR

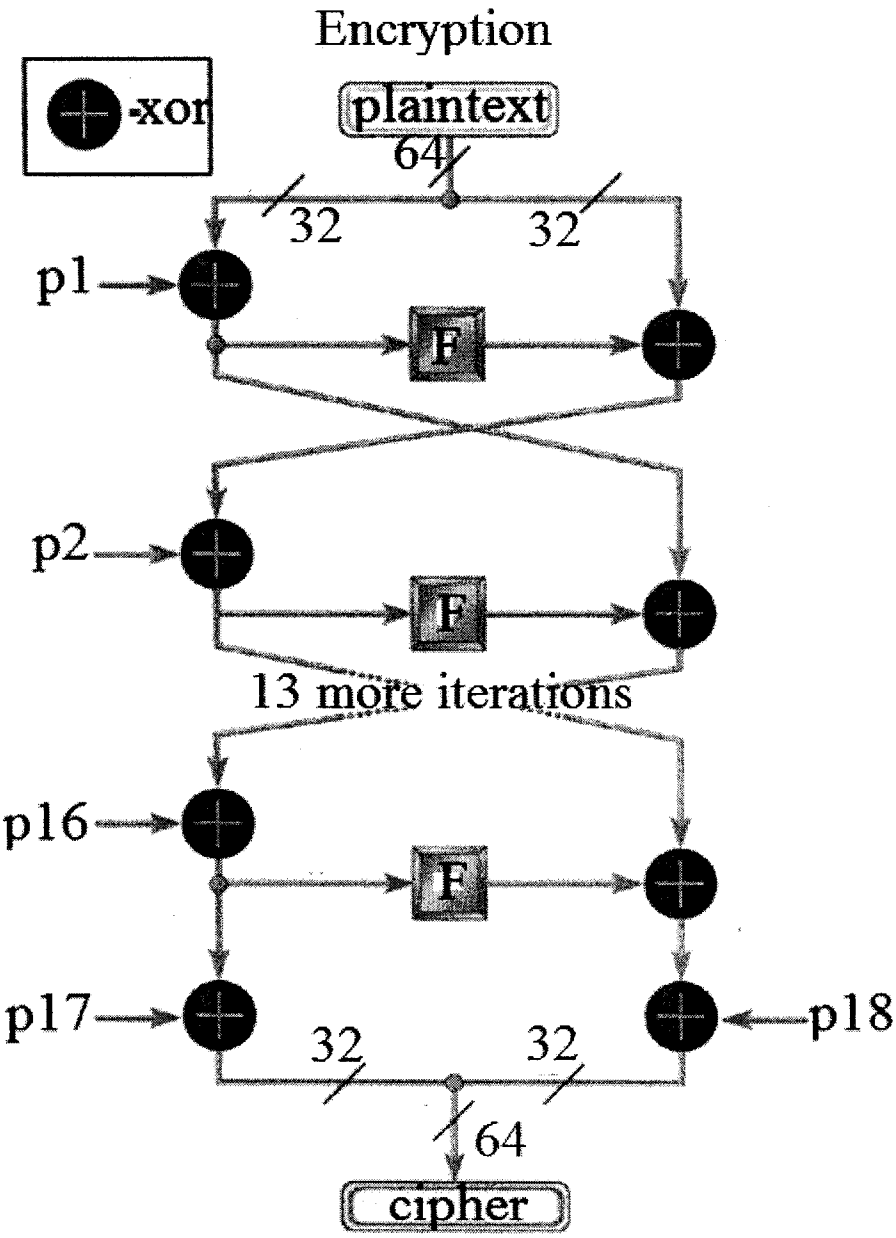


Fig: 2.1 Blowfish Encryption Algorithm.

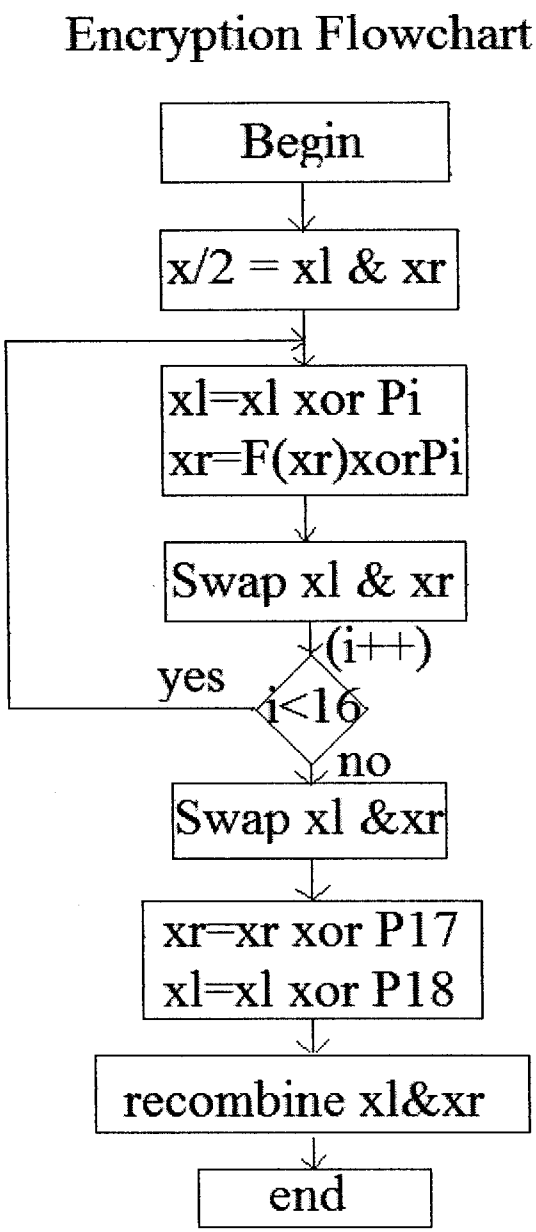


Fig 2.2 Blowfish algorithm- flowchart

.4 Function F

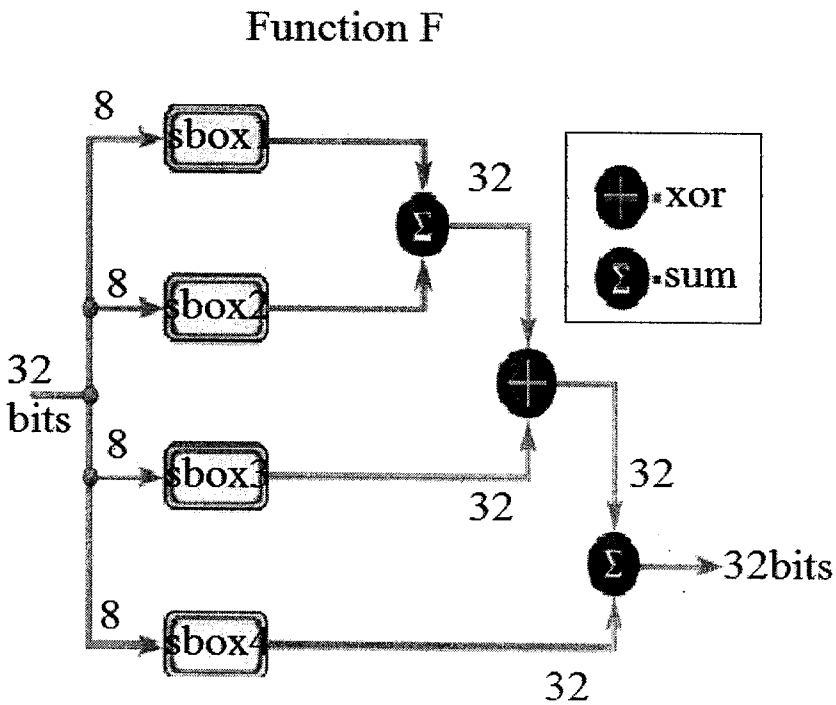


Fig 2.3 Feistel Function F

Divide xL into four eight-bit quarters: a, b, c, and d

$F(XL) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 2^{32}$

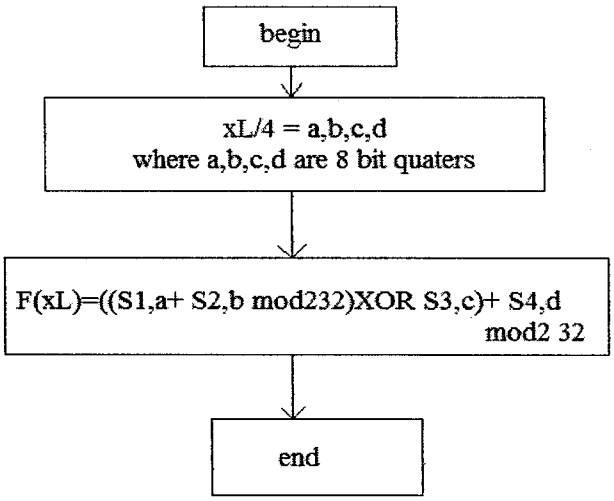


Fig: 2.4: Function F Network-flowchart

2.5 Decryption:

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order. Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all sub-keys are stored in cache.

Divide x into two 32-bit halves: xL, xR

For i = 18 to 3

xL = xL XOR Pi

xR = F(xL) XOR xR

Swap xL and xR

Swap xL and xR (Undo the last swap.)

xR = xR XOR P2

xL = xL XOR P1 ; Recombine xL and xR

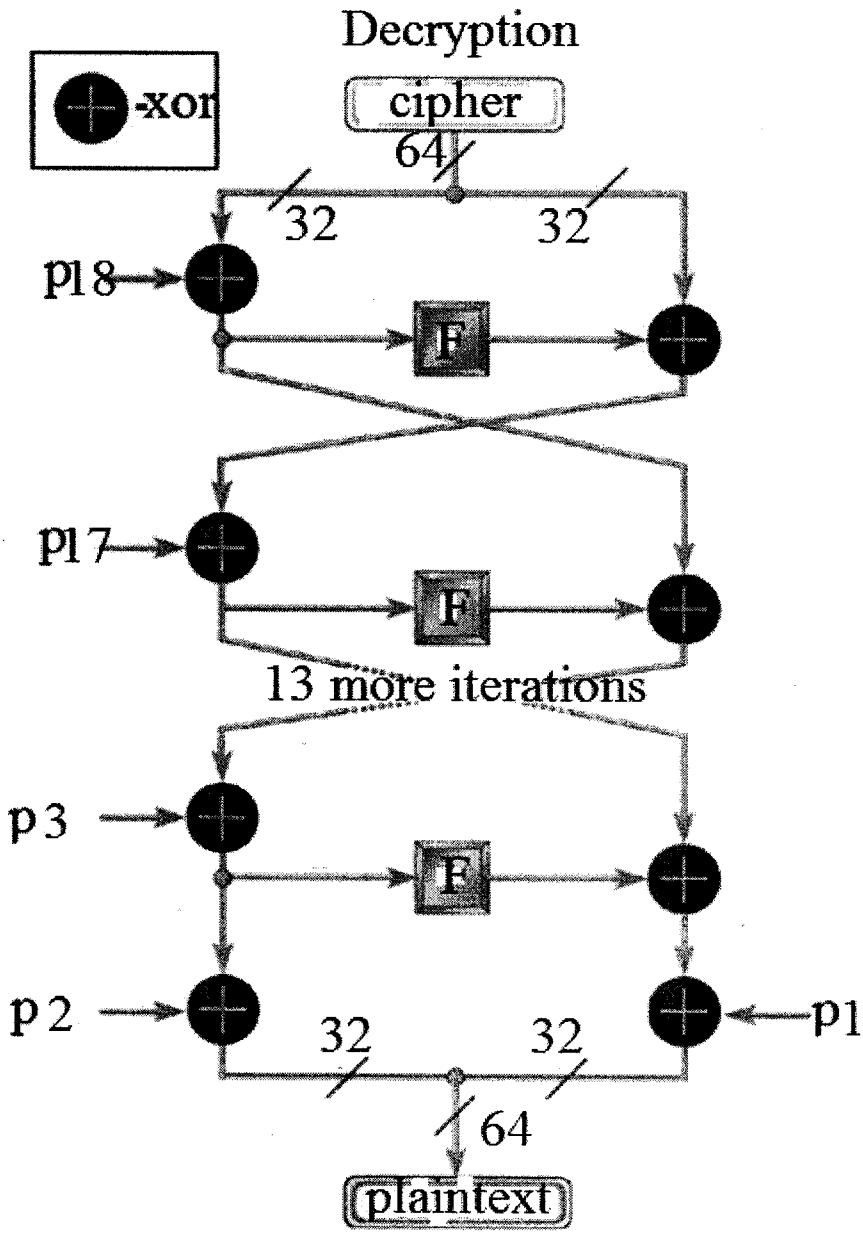


Fig 2.5: Blowfish Decryption

2.6 Sub Key Expansion

The sub-keys are calculated using the Blowfish algorithm. The exact method is as follows:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3).

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits.

3. Encrypt the all-zero string with the Blowfish algorithm, using the sub-keys described in steps (1) and (2).

4. Replace P1 and P2 with the output of step (3).

5. Encrypt the output of step (3) using the Blowfish algorithm with the modified sub-keys.

6. Replace P3 and P4 with the output of step (5).

7. Continue the process, replacing all entries of the P- array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm. In total, 521 iterations are required to generate all required sub-keys. Applications can store the sub-keys rather than execute this derivation process multiple times.

dependently of any other. High-end implementations could still ecompute the subkeys for increased speed, but low-end applications could ly compute the required subkeys when needed.

2 Operation Modes of Blowfish

lowfish is a symmetric block cipher that can be used as a drop-in placement for DES or IDEA so that it can be used in four standard peration modes as DES and IDEA. Four modes are defined as follows

2.1.Electronic Codebook Mode: Electronic codebook (ECB) mode is the ost obvious way to use a block cipher: A block of plaintext encrypts into a ock of ciphertext. Fig. 3.1 shows the ECB mode. Since the same block of aintext always encrypts to the same block of ciphertext, it is theoretically ossible to create a code book of plaintexts and corresponding ciphertexts. he potentially serious problem with this mode is that an adversary could odify encrypted message without knowing the key as to cheat the receiver. his disadvantage can be overcome by introducing a small amount of emory in the encryption process. The three modes below can counter such ind attack called block relay

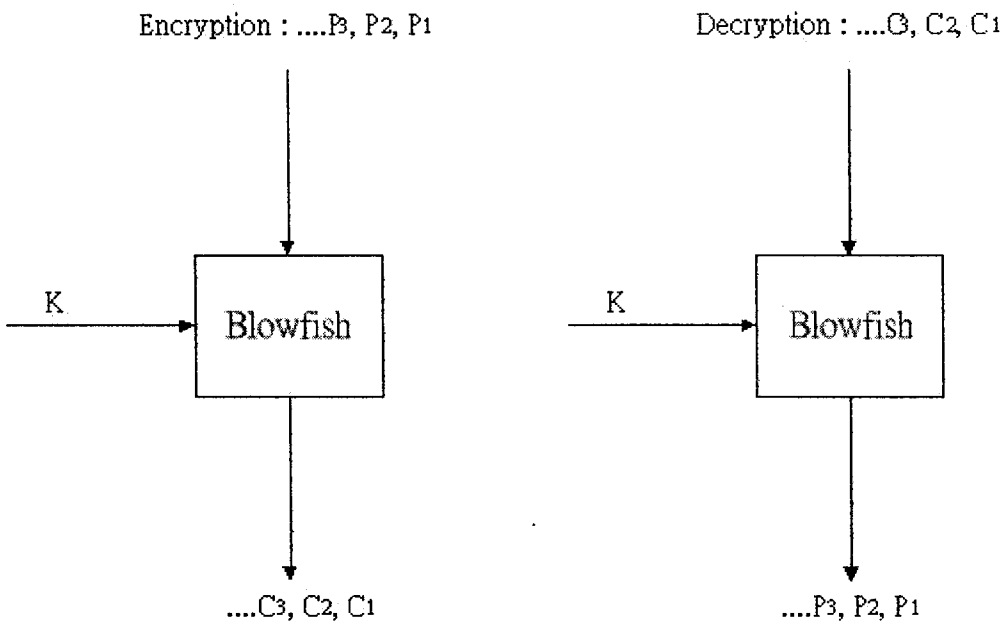


Fig 3.1 : ECB mode

2.2.Cipher Block Chaining Mode: In cipher block chaining (CBC) mode, ı initial value is added modulo 2 (XORed) to the first plaintext block to ɔrm the Blowfish input block. The Blowfish output is the ciphertext. This ıtput is fed back and added modulo 2 to the next plaintext block forming .e new Blowfish input block. This mode produces a ciphertext dependent ı the previous plaintext blocks. Fig. 3.2a shows the CBC encryption mode. ig. 3.2b shows the CBC decryption mode.

$i = Ei\ (Pi \oplus\ C\ i\text{-}1)$

$i = C\ i\text{-}1 \oplus Dk\ (Ci\ )$

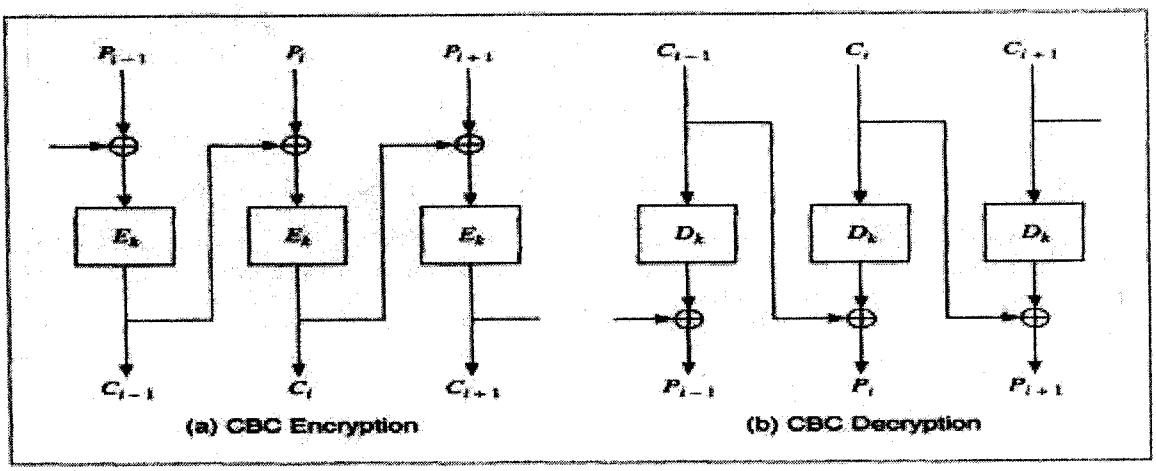


Fig 3.2: CBC Mode

3.2.3.Cipher-feedback Mode: Block ciphers can also be implemented as a self synchronizing stream cipher; this is called cipher-feedback (CFB) mode. In this mode, input is processed by j bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce the next unit of ciphertext. Again, this is useful for encoding long blocks of input. Fig. 3.3 shows the CFB mode.

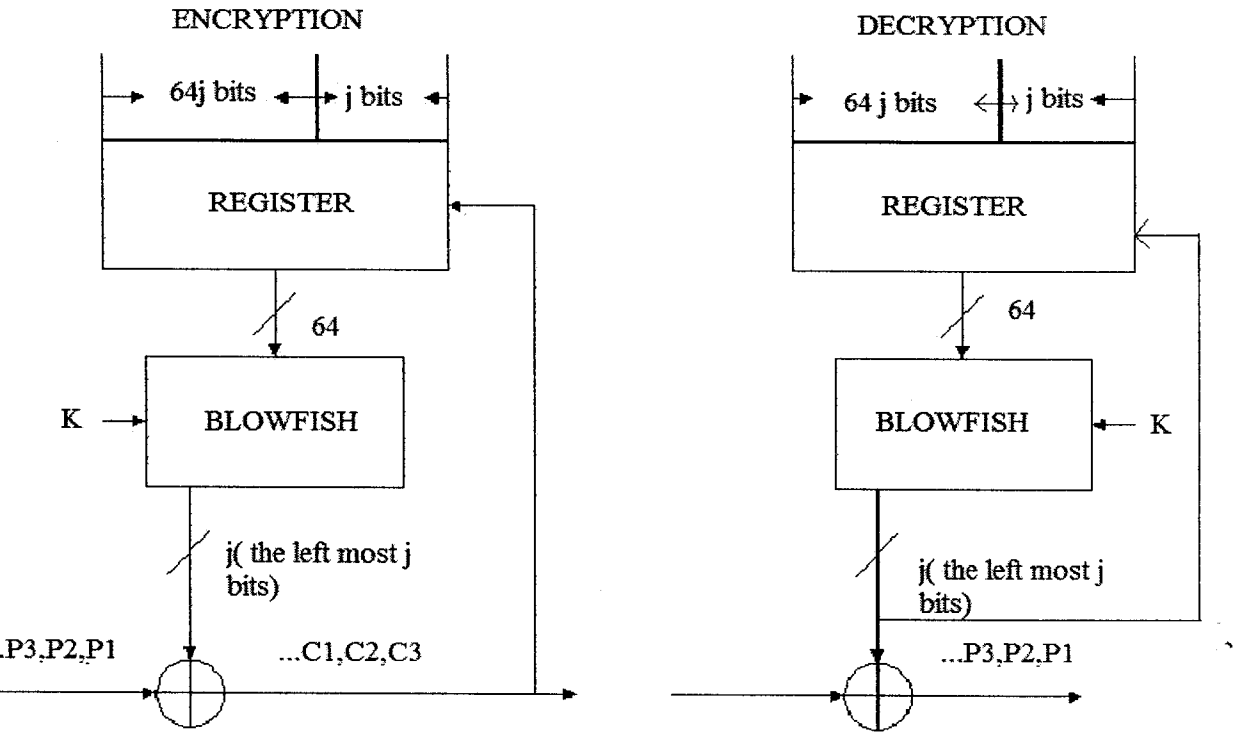


Fig 3.3: CFB Mode

2.4. Output-feedback Mode: The output-feedback (OFB) mode is a ethod of running a block cipher as a synchronous stream cipher. It's milar to CFB mode, except that j bits of the previous output block are oved into the right-most positions of the sequence (see Fig. 3.4). ecryption is the reverse of this process.

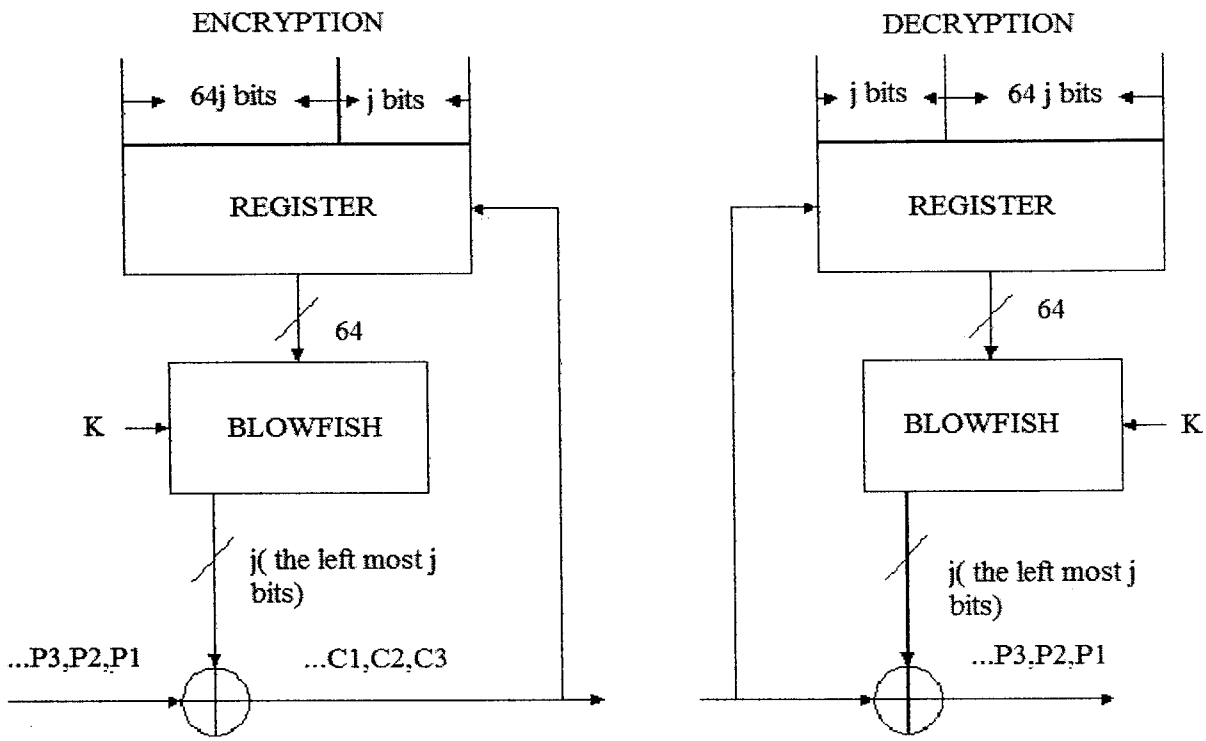


Fig 3.4: OFB Mode

3.3 Cryptanalysis of Blowfish:

Blowfish is very secure, that none came close to actually successfully cracking or providing a cryptanalysis of Blowfish. After the initial proposal of the Blowfish cipher, Dr.Dobb's journal sponsored a cryptanalysis contest in order to ascertain the security of Blowfish. Many interesting results were proposed. However, none came close to actually successfully cracking or providing a cryptanalysis of Blowfish. Some of the most intriguing results of the last century will be presented here for completeness.

nly five results in total were submitted. John Kesley could only break 3-ound Blowfish (nowhere near the 16-round final version), and his yptanalysis cannot be extended beyond 3 rounds. Serge Vaudenay used an tentionally weakened version of Blowfish and found a known-plaintext tack requiring 2^{8r+1} (where r is the number of rounds) known plaintexts to reak; however, this method is impractical in reality and does not work gainst the full 16-round Blowfish algorithm. The most promising attack as proposed in 1996 by Vincent Rijmen in his doctoral dissertation, but is attack can only break 4 rounds of Blowfish and no more. The most ecent work is from Dieter Schmidt, who noted that the third and fourth ubkeys are independent from the user's 64-bit key.

.4 Method of Implementation:

he S-box contains a table of non-linear data which maps inputs to outputs. his design severely disrupts any sort of correlation between the input and utput of the design. The use of custom instructions, when added to a Micro laze processor, allows us to gain the significantly greater processing power f custom hardware logic on an FPGA. At the same time, the ease of the raditional C++ and C software engineering process using Xilinx platform tudio is retained.

5 Advantage over other algorithms

- It is more secured. None came close to actually successfully cracking or providing a cryptanalysis of Blowfish.
- Relatively faster and more efficient.
- It uses strong key for cryptography.
- The encryption or the decryption state lasts for 16 cycles.
- Un-patented and royalty free.
- Simple structure to implement.
- It is significantly faster than DES when implemented on 32-bit microprocessors with large data caches, such as the Pentium and the PowerPC.

.6 Blowfish Limitations

Blowfish is a variable-length key block cipher. It does not meet all the equirements for a new cryptographic standard. It is only suitable for pplications where the key does not change often, like a communications nk or an automatic file encryptor.

'ools:

- The whole algorithm has been developed in Xilinx ISE 8.1i tool
- The blowfish Algorithm has been written in VHDL & C language.
- The simulation and synthesis has been done using Xilinx ISE & Xilinx platform studio tools.

CHAPTER 4

HARDWARE AND SOFTWARE TOOLS

1 Digital Designs

Prompted by the development of new types of sophisticated field-rogrammable devices (FPDs), the process of designing digital hardware has hanged dramatically over the past few years. Unlike previous generations f technology, in which board-level designs included large numbers of SSI hips containing basic gates, virtually every digital design produced today onsists mostly of high-density devices. This applies not only to custom evices like processors and memory, but also for logic circuits such as state hachine controllers, counters, registers, and decoders. When such circuits re destined for high-volume systems they have been integrated into high-ensity gate arrays. However, gate array NRE costs often are too expensive nd gate arrays take too long to manufacture to be viable for prototyping or ther low-volume scenarios. For these reasons, most prototypes, and also hany production designs are now built using FPD s. The most compelling dvantages of FPDs are instant manufacturing turnaround, low start-up osts, low financial risk and (since programming is done by the end user) ase of design changes.

Some of the most important terminologies used are discussed below:

.1.1 **PLA — Programmable Logic Array (PLA)** is a relatively small FPD hat contains two levels of logic, an AND-plane and an OR-plane, where oth levels are programmable (note: although PLA structures are sometimes mbedded into full-custom chips, we refer here only to those PLAs that are rovided as separate integrated circuits and are user-programmable).

.1.2 **PAL — Programmable Array Logic (PAL)** is a relatively small FPD hat has a programmable AND-plane followed by a fixed OR-plane

.1.3 **Field-Programmable Device (FPD)** — a general term that refers to ny type of integrated circuit used for implementing digital hardware, where he chip can be configured by the end user to realize different designs. rogramming of such a device often involves placing the chip into a special rogramming unit, but some chips can also be configured "in-system". Another name for FPDs is *programmable logic devices* (PLDs); although PLDs encompass the same types of chips as FPDs, the term FPD is preferred ecause historically the word PLD has referred to relatively simple types of evices.

.1.3 **CPLD — Complex Programmable Logic Devices** that consists of an rrangement of multiple SPLD (simple PLD either PLA or PAL) like blocks n a single chip. Alternative names are sometimes adopted for this style of hip are Enhanced PLD (EPLD), Super PAL, Mega PAL, and others.

.1.4 **FPGA — a Field-Programmable Gate Array** is an FPD featuring a eneral structure that allows very high logic capacity. Whereas CPLDs

ıture logic resources with a wide number of inputs (AND planes), FPGAs ſer more narrow logic resources. FPGAs also offer a higher ratio of flip-ps to logic resources than do CPLDs.

2 Commercially Available FPGAs

There are two basic categories of FPGAs on the market today: 1. RAM-based FPGAs and 2. Anti fuse- based FPGAs. In the first category, linx and Altera are the leading manufacturers in terms of number of users, th the major competitor being AT&T. For antifuse-based products, Actel, uick logic and Cypress, and Xilinx offer competing products.

2.1 Needs For FPGA

Because they offer high speeds and a range of capacities, FPGAs are eful for a very wide assortment of applications, from implementing ndom glue logic to prototyping small gate arrays. One of the most mmon uses in industry at this time, and a strong reason for the large owth of the FPGA market, is the conversion of designs that consist of ultiple SPLDs into a smaller number of FPGAs. FPGAs can realize asonably complex designs, such as graphics controller, LAN controllers, ARTs, cache control, and many others. As a general rule-of-thumb, rcuits that can exploit wide AND/OR gates, and do not need a very large umber of flip-flops are good candidates for implementation in FPGAs.

A significant advantage of FPGAs is that they provide simple design hanges through re-programming Within system programmable FPGAs it is ven possible to re-configure hardware (an example might be to change a

rotocol for a communications circuit) without power-down. Predictability f circuit implementation is one of the strongest advantages of FPGA rchitectures.

.3 Spartan-3E FPGA Features:

'he Spartan-3E Starter Kit board highlights the unique features of the partan-3E FPGA family and provides a convenient development board for mbedded processing applications. The board highlights these features:

- Parallel NOR Flash configuration
- MultiBoot FPGA configuration from Parallel NOR Flash PROM
- SPI serial Flash configuration

:mbedded development

- MicroBlaze™ 32-bit embedded RISC processor
- PicoBlaze™ 8-bit embedded controller
- DDR memory interfaces

Key Components and Features:

'he key features of the Spartan-3E Starter Kit board are:

- Up to 232 user-I/O pins
- 320-pin FBGA package
- Over 10,000 logic cells
- Xilinx 4 Mbit Platform Flash configuration PROM
- Xilinx 64-macrocell XC2C64A CoolRunner CPLD
- 16 Byte (512 Mbit) of DDR SDRAM, x16 data interface, 100+ MHz
- 16 Byte (128 Mbit) of parallel NOR Flash (Intel StrataFlash)
- 16Mbits of SPI serial Flash (STMicro)

- FPGA configuration storage
- MicroBlaze code shadowing
- 2-line, 16-character LCD screen
- PS/2 mouse or keyboard port
- VGA display port
- 10/100 Ethernet PHY (requires Ethernet MAC in FPGA)
- Two 9-pin RS-232 ports (DTE- and DCE-style)
- On-board USB-based FPGA/CPLD download/debug interface
- 50 Hz clock oscillator
- 50 SHA-1 1-wire serial EEPROM for bitstream copy protection
- Hirose FX2 expansion connector
- Three Digilent 6-pin expansion connectors
- Four-output, SPI-based Digital-to-Analog Converter (DAC)
- Two-input, SPI-based Analog-to-Digital Converter (ADC) with programmable-gain Pre-amplifier.
- ChipScope™ SoftTouch debugging port
- Rotary-encoder with push-button shaft
- Eight discrete LEDs
- Four slide switches
- Four Push Button Switches
- SMA clock input
- 8 pin DIP socket for auxiliary clock oscillator

4.4 FPGA Configuration Options

The Spartan-3E Starter Kit board supports a variety of configuration options:

• Download FPGA designs directly to the Spartan-3E FPGA via JTAG, using the onboard USB interface. The on-board USB-JTAG logic also provides in-system programming for the on-board Platform Flash PROM and the Xilinx XC2C64A CPLD. SPI serial Flash and StrataFlash programming are performed separately.

• Program the on-board 4 Mbit Xilinx XCF04S serial Platform Flash PROM, then configure the FPGA from the image stored in the Platform Flash PROM using Master Serial mode.

• Program the on-board 16 Mbit ST Microelectronics SPI serial Flash PROM, then configure the FPGA from the image stored in the SPI serial Flash PROM using SPI mode.

• Program the on-board 128 Mbit Intel StrataFlash parallel NOR Flash PROM, then configure the FPGA from the image stored in the Flash PROM using BPI Up or BPI Down configuration modes. Further, an FPGA application can dynamically load two different FPGA configurations using the Spartan-3E FPGA's MultiBoot mode.

4.4.1 Configuration Methods:

A typical FPGA application uses a single non-volatile memory to store configuration images. To demonstrate new Spartan-3E capabilities, the starter kit board has three different configuration memory sources that all need to function well together. The extra configuration functions make the starter kit board more complex than typicalSpartan-3E applications.

The starter kit board also includes an on-board USB-based JTAG programming interface. The on-chip circuitry simplifies the device

programming experience. In typical applications, the JTAG programming hardware resides off-board or in a separate programming module, such as the Xilinx Platform USB cable.

4.4.2 Voltages for all applications:

The Spartan-3E Starter Kit board showcases a triple-output regulator developed by Texas Instruments, the **TPS75003** specifically to power Spartan-3 and Spartan-3E FPGAs. This regulator is sufficient for most stand-alone FPGA applications. However, the starter kit board includes DDR SDRAM, which requires its own high-current supply. Similarly, the USB-based JTAG download solution requires a separate 1.8V supply.

4.4.3 JTAG:

JTAG primary purpose is to allow a computer to take control of the state of all the IO pins on a board. Standard JTAG commands are used for devices testing purpose.

FPGAs are JTAG-aware and so all the FPGA IO pins can be controlled from the JTAG interface. FPGAs add the ability to be configured through JTAG (using proprietary JTAG commands).

JTAG consists of 4 signals: TDI, TDO, TMS and TCK. A fifth pin, TRST, is optional. A single JTAG port can connect to one or multiple devices (as long as they are all JTAG-aware parts). With multiple devices, "JTAG chain" can be created. The TMS and TCK are tied to all the devices directly, but the TDI and TDO form a chain: TDO from one device goes to TDI of the next one in the chain. The master controlling the chain (a computer usually) closes the chain.

.4.4 RS 232:

The Spartan-3E Starter Kit board has two RS-232 serial ports: a female)B9 DCE connector and a male DTE connector. The DCE-style port onnects directly to the serial port connector available on most personal omputers and workstations via a standard straight-through serial cable. Null nodem, gender changers, or crossover cables are not required.

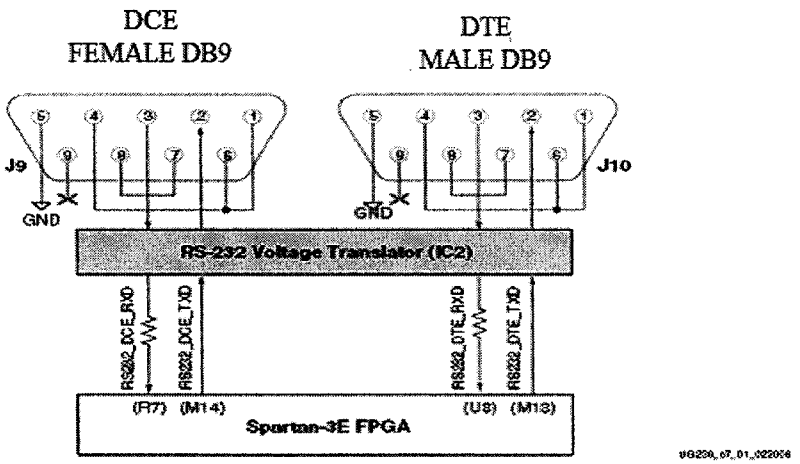


Figure 7-1: **RS-232 Serial Ports**

Fig 4.1 Connection between the FPGA and the two DB9 connectors

The FPGA supplies serial output data using LVTTL or LVCMOS evels to the Maxim device, which in turn, converts the logic value to the appropriate RS-232 voltage level. Likewise, the Maxim device converts the RS-232 serial input data to LVTTL levels for the FPGA. A series resistor

etween the Maxim output pin and the FPGA's RXD pin protects against ccidental logic conflicts.Hardware flow control is not supported on the onnector.

.5 Development Tools:

The FPGA / FPGA chip is supported with a complete set of software and ardware development tools which includes Xilinx Embedded Development it (EDK). EDK itself contains two parts

- Xilinx Platform Studio (XPS)
- Software Development Kit (SDK)

DK is used to create a simple processor system and the process of adding a ustom OPB peripheral (an 32-bit adder circuit) to that processor system by sing the Import Peripheral Wizard. The microprocessors available for use n Xilinx Field Programmable Gate Arrays (FPGAs) with Xilinx EDK oftware tools can be broken down into two broad categories. There are soft-ore microprocessors (MicroBlaze) and the hard-core embedded nicroprocessor (PowerPC).

.5.1 Microblaze & EDK:

The MicroBlaze is a virtual microprocessor that is built by combining locks of code called cores inside a Xilinx Field Programmable Gate Array FPGA). The MicroBlaze processor is a 32-bit Harvard Reduced Instruction et Computer (RISC)architecture optimized for implementation in Xilinx PGAs with separate 32-bit instruction and data buses running at full speed o execute programs and access data from both on-chip and external memory t the same time.

The backbone of the architecture is a single-issue, 3-stage pipeline with 32 general-purpose registers, an Arithmetic Logic Unit (ALU), a shift unit, and two levels of interrupt. This basic design can then be configured with more advanced features to tailor to the exact needs of the target embedded application such as: barrel shifter, divider, multiplier, single precision floating-point unit (FPU), instruction and data caches, exception handling, debug logic, Fast Simplex Link (FSL) interfaces and others. This flexibility allows the user to balance the required performance of the target application against the logic area cost of the soft processor.

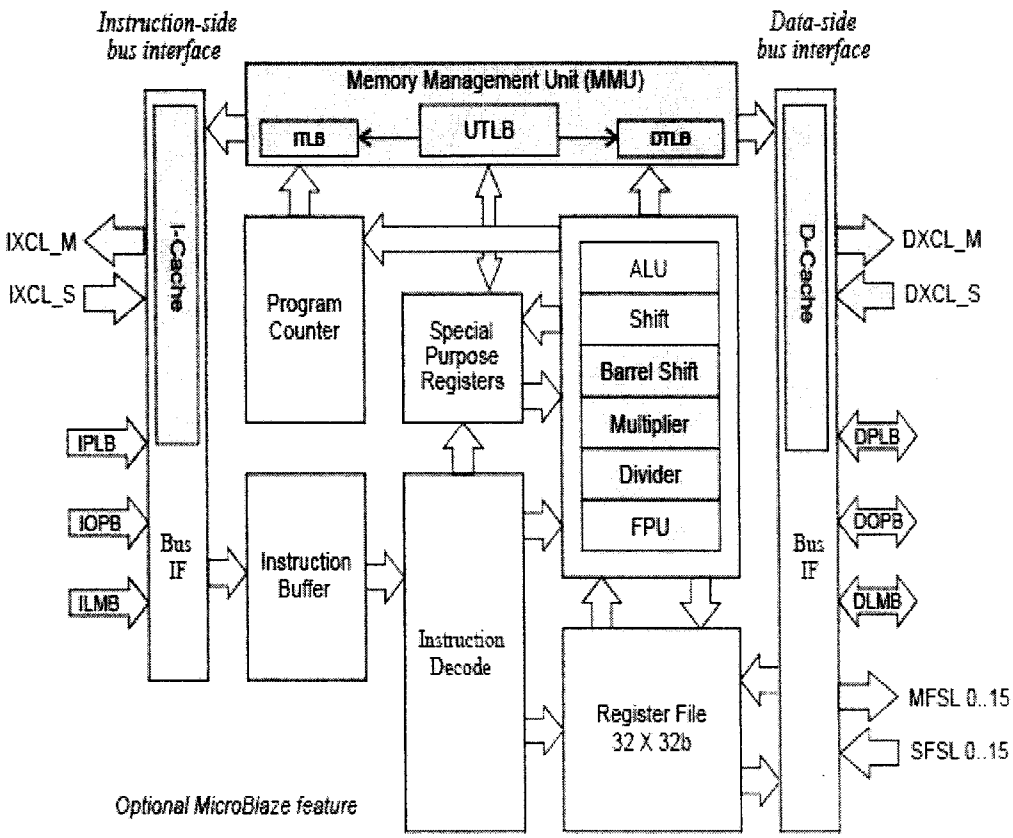


Figure 1-1: **MicroBlaze Core Block Diagram**

Fig 4.2: Microblaze Core Block Diagram

MicroBlaze supports reset, interrupt, user exception, break and hardware exceptions. For interrupts, MicroBlaze supports only one external interrupt source (connecting to the Interrupt input port). If multiple interrupts are needed, an interrupt controller must be used to handle multiple interrupt requests to MicroBlaze. An interrupt controller is available for use with the Xilinx Embedded Development Kit (EDK) software tools.

Writing software to control the MicroBlaze processor must be done in C/C++ language. Using C/C++ is the preferred method by most people and is the format that the Xilinx Embedded Development Kit (EDK) software tools expect. The EDK tools have built in C/C++ compilers to generate the necessary machine code for the MicroBlaze processor.

The processor system by EDK is connected by On-chip Peripheral Bus (OPB) and/or Processor Local Bus (PLB), for which the custom peripheral must be OPB or PLB compliant. Meaning the top-level module of custom peripheral must contain a set of bus ports that is compliant to OPB or PLB protocol, so that it can be attached to the system OPB or PLB bus.

EDK uses Intellectual-Property Interface (IPIF) library to implement common functionality among various processor peripherals. It gives a set of simplified bus protocol called IP Interconnect (IPIC), which is much easier to use rather than operate on OPB or PLB bus protocol directly. Using the IPIF module with parameterization will greatly reduce your design and test effort. This is done in EDK with a wizard that walks through the entire process.

.5.2 Microblaze Features:

The MicroBlaze soft core processor is highly configurable, allowing ou to select a specific set of features required by your design.

'he fixed feature set of the processor includes:

Thirty-two 32-bit general purpose registers

32-bit instruction word with three operands and two addressing modes

32-bit address bus

Single issue pipeline

In addition to these fixed features, the MicroBlaze processor is arameterized to allow selective enabling of additional functionality.

'rocessor Local Bus (PLB) Interface:

'he MicroBlaze PLB interfaces are implemented as byte-enable capable 32-it masters.

On-Chip Peripheral Bus (OPB) Interface:

'he MicroBlaze OPB interfaces are implemented as byte-enable capable nasters.

Local Memory Bus (LMB) Interface Description:

'he LMB is a synchronous bus used primarily to access on-chip block RAM. It uses a minimum number of control signals and a simple protocol to ensure that local block RAM are accessed in a single clock cycle.

.5.3 Xilinx Platform Studio (XPS):

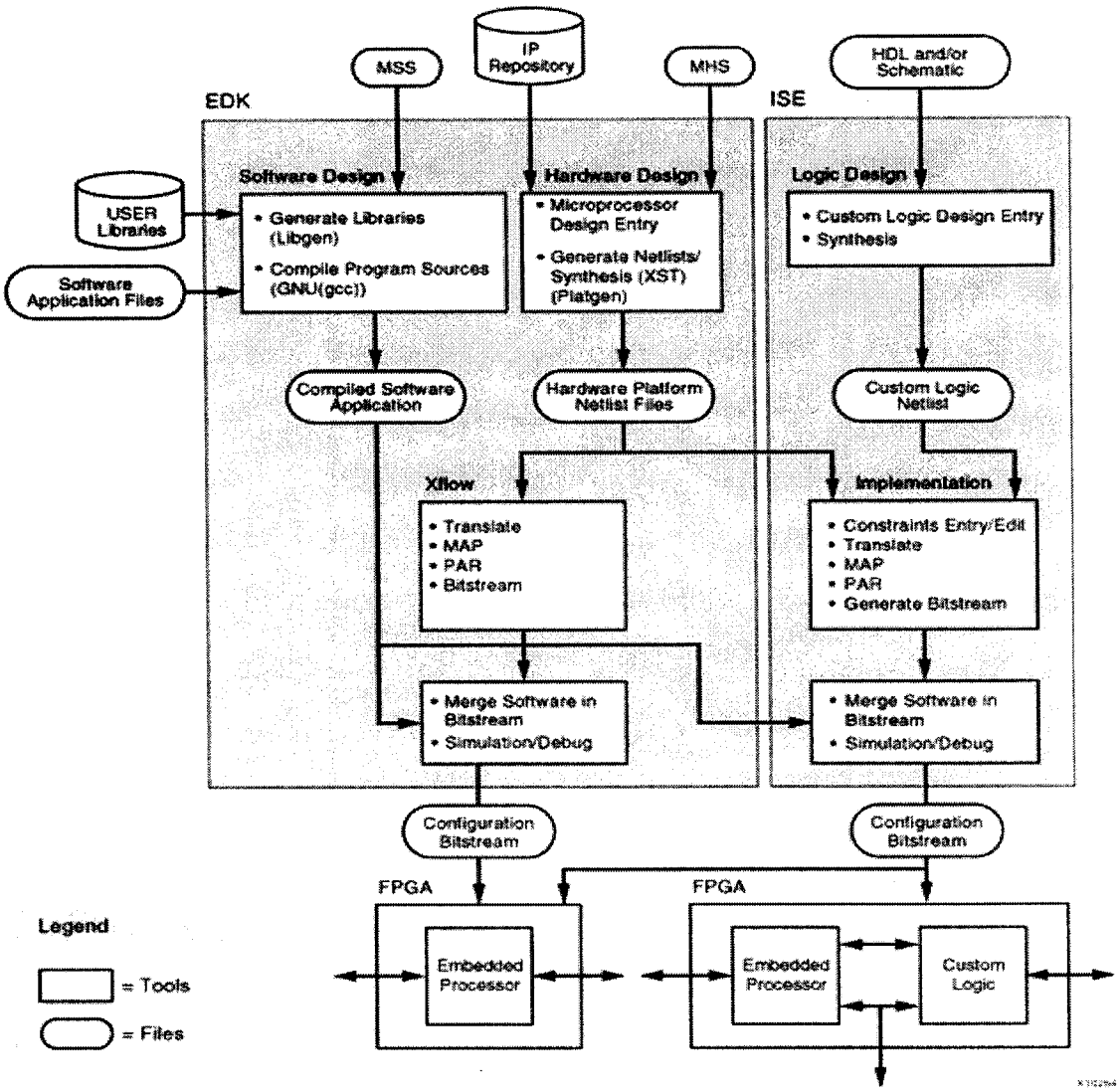


Fig 4.3: Hardware And Software Architecture Development

XPS includes a graphical user interface (GUI), along with a set of tools that aid in project design. From the XPS GUI, a complete embedded processor system can be designed for implementation within a Xilinx FPGA device.

.5.3.1 Microprocessor Hardware Specification (MHS):

XPS provides an interactive development environment that allows to pecify all aspects of a hardware platform. XPS maintains the hardware latform description in a high-level form, known as the Microprocessor Iardware Specification (MHS) file. The MHS, an editable text file, is the rincipal source file representing the hardware component of embedded ystem. XPS synthesizes the MHS source file into Hardware Description Language (HDL) netlists ready for FPGA place and route.

The MHS file is integral for design process. It contains all peripherals along with their parameters. The MHS file defines the configuration of the embedded processor system and includes information on the bus architecture, peripherals, processor, connectivity, and address space.

4.5.3.2 Microprocessor Software Specification (MSS):

XPS maintains an analogous software system description in the Microprocessor Software Specification (MSS) file. The MSS file, together with your software applications, are the principal source files representing the software elements of the embedded system. This collection of files, used in conjunction with EDK installed libraries and drivers and any custom libraries and drivers for custom peripherals you provide, allows XPS to compile your applications. The compiled software routines are available as an Executable and Linkable Format (ELF) file. The ELF file is the binary ones and zeros that are run on the processor hardware. The figure below shows the files and flow stages that generate the ELF file.

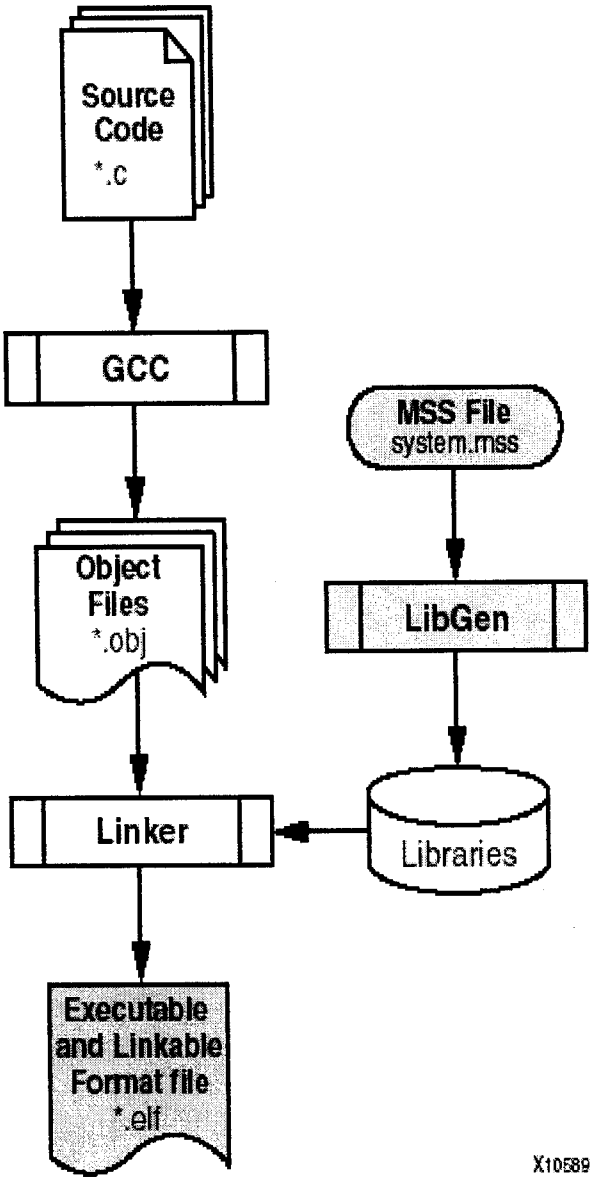


Fig 4.4: Elements and Stages of ELF File Generation

After the hardware and software designs are completed, the FPGA device is configured and the design is downloaded. Then the design is executed and the result is obtained as a hyperterminal output.

6 EXECUTION RESULTS:

ne Simulation results are obtained from Xilinx ISE 8.1I tool

YNTHESIS REPORT:

arted : "Synthesize".

```
=========================================================================
                        HDL Compilation                  *
=========================================================================
ompiling vhdl file "C:/Documents and
ttings/Singam/Desktop/blowfish/blow_decryp.vhd" in Library work.
rchitecture behavioral of Entity toplevel is up to date.
rchitecture behavioral of Entity round is up to date.
rchitecture behavioral1 of Entity keyarray is up to date.
rchitecture behavioral of Entity ffun is up to date.
rchitecture behavioral of Entity sbox is up to date.
rchitecture behavioral of Entity bxr is up to date.

=========================================================================
                        HDL Analysis                     *
=========================================================================
nalyzing Entity <toplevel> (Architecture <behavioral>).
ntity <toplevel> analyzed. Unit <toplevel> generated.

nalyzing Entity <keyarray> (Architecture <behavioral1>).
ntity <keyarray> analyzed. Unit <keyarray> generated.

nalyzing Entity <bxr> (Architecture <behavioral>).
ntity <bxr> analyzed. Unit <bxr> generated.

nalyzing Entity <round> (Architecture <behavioral>).
ntity <round> analyzed. Unit <round> generated.
```

```
nalyzing Entity <ffun> (Architecture <behavioral>).
ntity <ffun> analyzed. Unit <ffun> generated.

nalyzing Entity <sbox> (Architecture <behavioral>).
ntity <sbox> analyzed. Unit <sbox> generated.

=========================================================================
*                    HDL Synthesis                    *
=========================================================================

ynthesizing Unit <sbox>.
  Related source file is "C:/Documents and
ettings/Singam/Desktop/blowfish/blow_decryp.vhd".
  Found 256x32-bit ROM for signal <s>.
  Summary:
        inferred   1 ROM(s).
nit <sbox> synthesized.


Synthesizing Unit <ffun>.
  Related source file is "C:/Documents and
Settings/Singam/Desktop/blowfish/blow_decryp.vhd".
  Found 32-bit adder for signal <e1>.
  Found 32-bit adder for signal <j>.
  Found 32-bit xor2 for signal <jj>.
  Summary:
        inferred   2 Adder/Subtractor(s).
Unit <ffun> synthesized.
```

```
ıthesizing Unit <round>.
Related source file is "C:/Documents and
tings/Singam/Desktop/blowfish/blow_decryp.vhd".
ARNING:Xst:1780 - Signal <q> is never used or assigned.
it <round> synthesized.

nthesizing Unit <bxr>.
Related source file is "C:/Documents and
ttings/Singam/Desktop/blowfish/blow_decryp.vhd".
Found 32-bit xor2 for signal <r1>.
ıit <bxr> synthesized.

nthesizing Unit <keyarray>.
Related source file is "C:/Documents and
ttings/Singam/Desktop/blowfish/blow_decryp.vhd".
ıit <keyarray> synthesized.
nthesizing Unit <toplevel>.
Related source file is "C:/Documents and
ttings/Singam/Desktop/blowfish/blow_decryp.vhd".
ıit <toplevel> synthesized.

=========================================================================
DL Synthesis Report
acro Statistics
ROMs                                    : 64
56x32-bit ROM                           : 64
Adders/Subtractors                      : 32
2-bit adder                             : 32
Xors                                    : 50
2-bit xor2                              : 50
=========================================================================
```

Advanced HDL Synthesis *

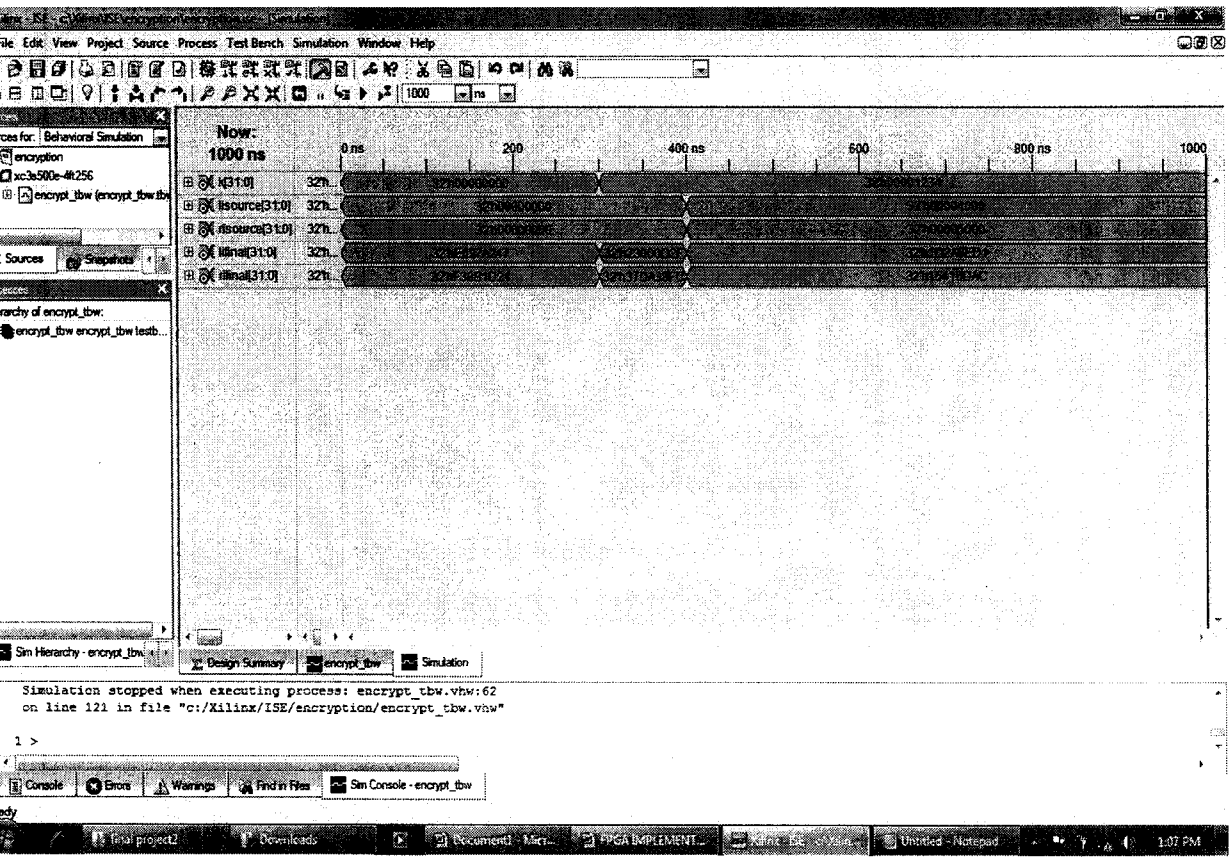
dvanced HDL Synthesis Report

Iacro Statistics

ROMs	: 64
256x32-bit ROM	: 64
Adders/Subtractors	: 32
32-bit adder	: 32
Xors	: 50
32-bit xor2	: 50

IMULATION WAVEFORMS USING XILINX ISE 8.1I

NCRYPTION



all in hexadecimal values)

Key: 1234

nput

Li = 4000

Ri = 5000

Cipher (encryption o/p)

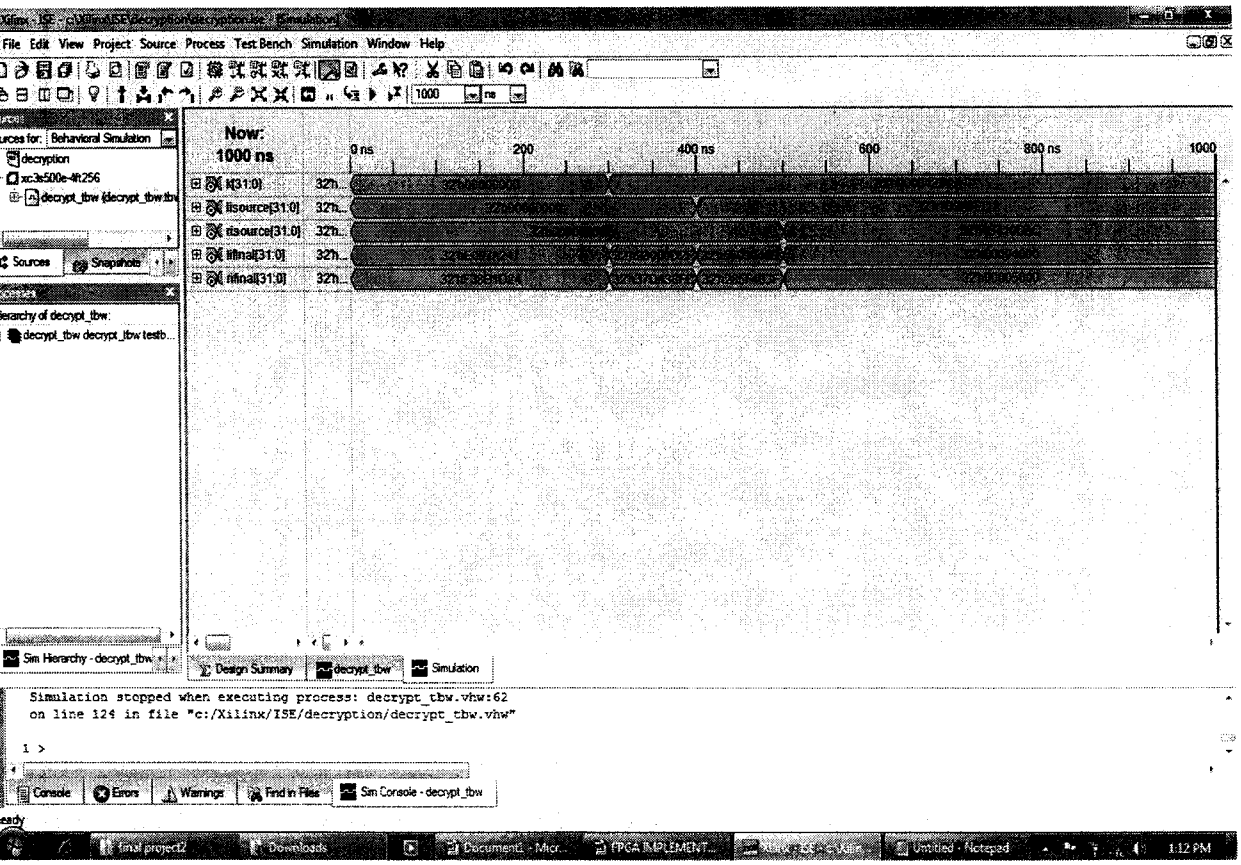
Lf = 1D2A6ED1

Rf = 15410DAC

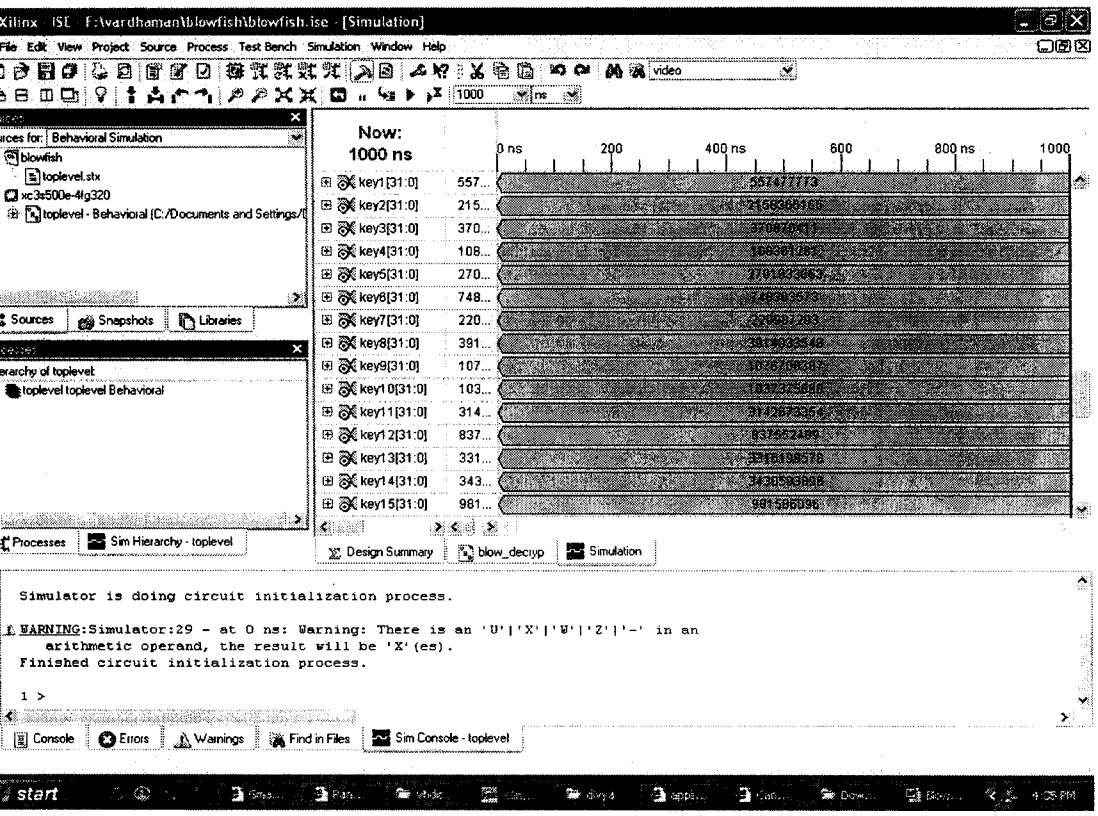
Decryption output

Lf =4000 ; Rf =5000

ECRYPTION

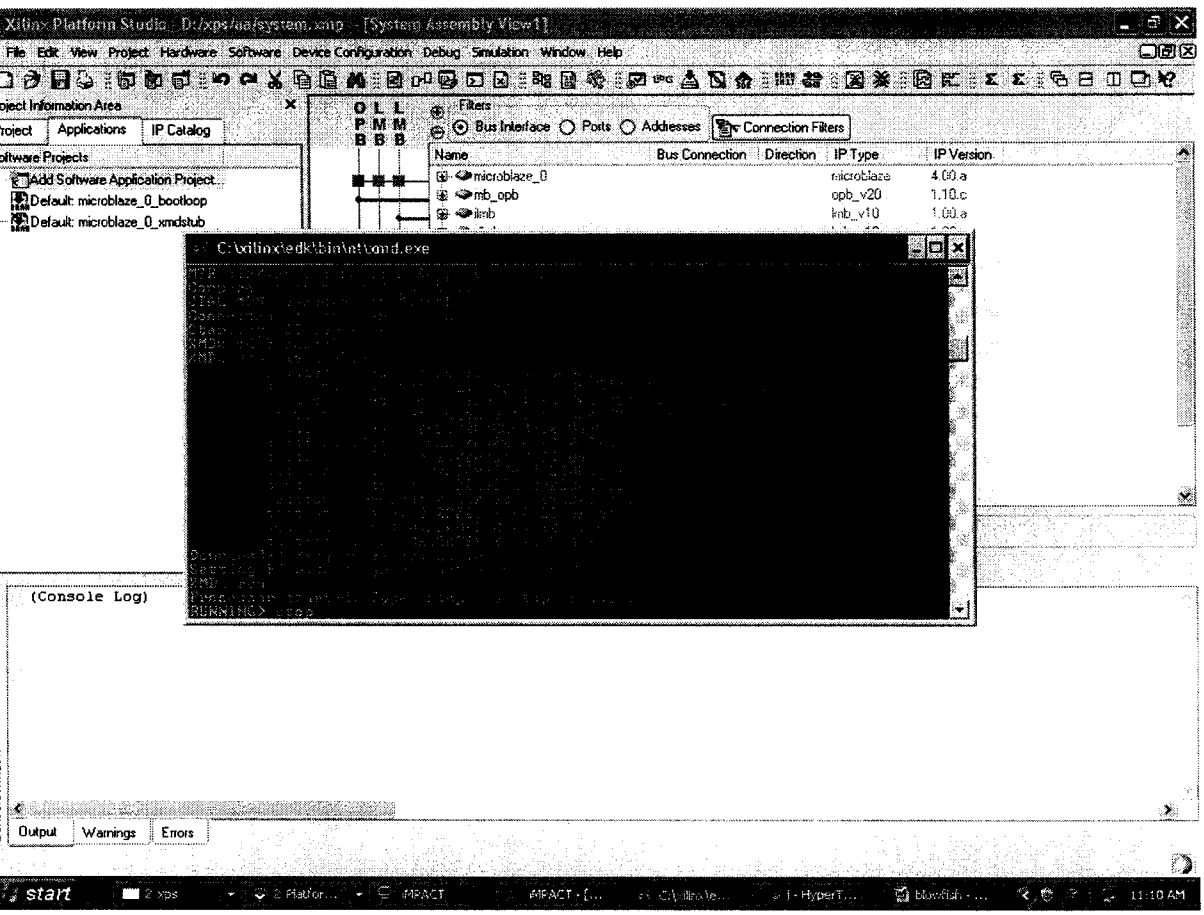

File Edit View Project Source Process Test Bench Simulation Window Help
Now:
1000 ns
0 ns
200
400 ns
600
800 ns
1000
i[31:0]
iisource[31:0]
risource[31:0]
iifinal[31:0]
rifinal[31:0]
Design Summary
decrypt_tbw
Simulation
Simulation stopped when executing process: decrypt_tbw.vhw:62
on line 124 in file "c:/Xilinx/ISE/decryption/decrypt_tbw.vhw"
Console
Errors
Warnings
Find in Files
Sim Console - decrypt_tbw

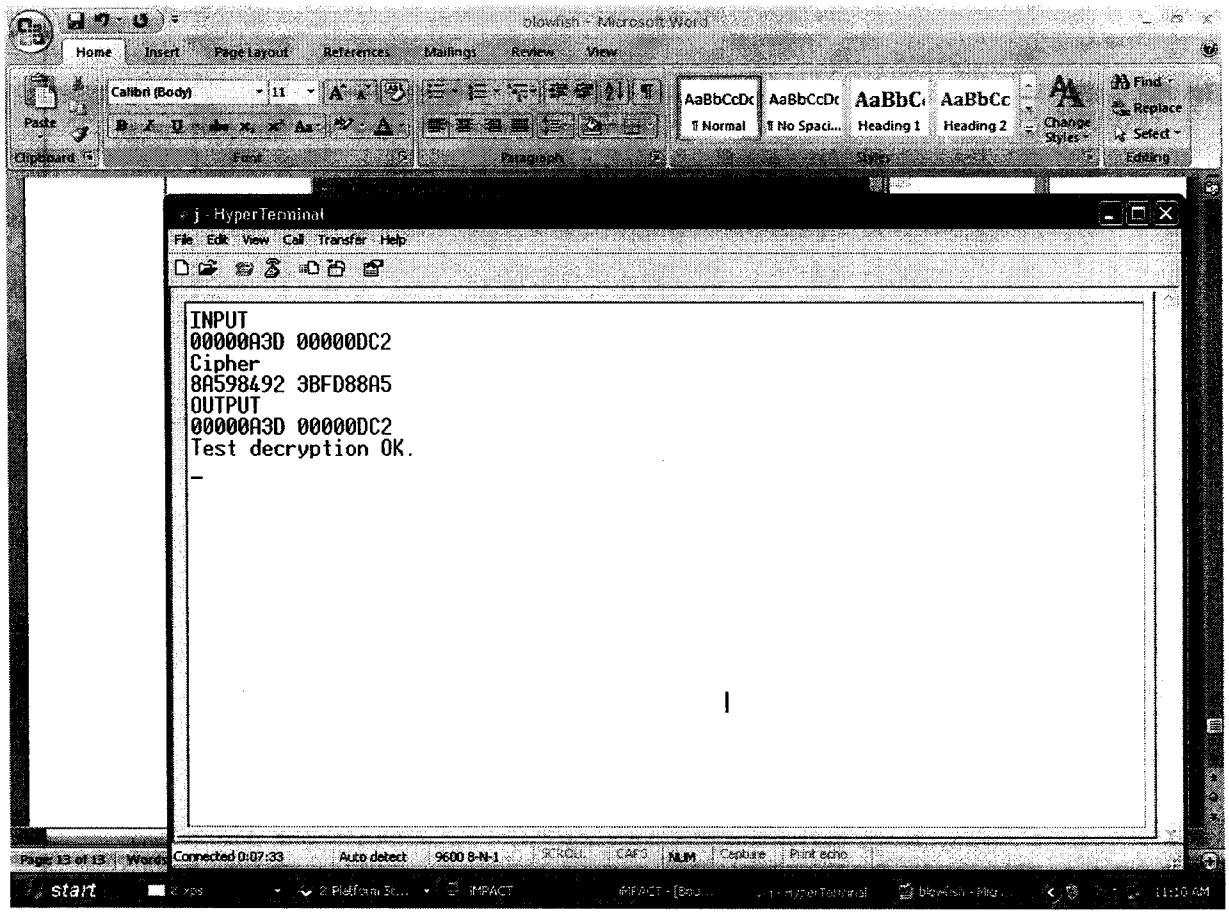
EY MANAGEMENT UNIT OUTPUT:



ARDWARE IMPLEMENTATION OUTPUT:

he hardware implementation of Blowfish Cryptographic algorithm in Field rogrammable Gate Array Chip (Spartan-3E XC3S500E FT256) using ilinx Platform Studio Tool is obtained as a Hyperterminal.




blowfish - Microsoft Word
j - HyperTerminal
File Edit View Call Transfer Help
INPUT
00000A3D 00000DC2
Cipher
8A598492 3BFD88A5
OUTPUT
00000A3D 00000DC2
Test decryption OK.
Connected 0:07:33
Auto detect
9600 8-N-1
start

.7 Applications of the algorithm:

ome Software Products that use blowfish algorithm:

) Access Manager by Citi Software Ltd.

.) Aedit- Windows word processor

) Foopchat:Encrypted chat and advanced file sharing using a client/server rchitechture.

-) JFile by Land-J Technologies:Database program for the palmOS latform.

) Freedom by Zero-Knowledge: Privacy for web browsing, e-mail, chat nd newsgroups.

Applications of the project

he project relies on the advantage of implementing the algorithm in ardware and hence it can be used for secure wireless communications.

.8 CONCLUSION:

A high-speed design of Blowfish cryptographic algorithm has been mplemented in hardware. Blowfish is among the fastest block ciphers vailable. The design takes advantage of the conceptual simplicity of Blowfish for encryption and decryption which can be used for secure vireless communications. Future works might include further increasing the peed of encryption and decryption through pipelining concepts, reducing he number of substitution boxes etc.

APPENDIX I

DEFAULT SBOX AND P ARRAY

```
tatic const unsigned long ORIG_P[16 + 2] = {
    0x243F6A88L, 0x85A308D3L, 0x13198A2EL, 0x03707344L,
    0xA4093822L, 0x299F31D0L, 0x082EFA98L, 0xEC4E6C89L,
    0x452821E6L, 0x38D01377L, 0xBE5466CFL, 0x34E90C6CL,
    0xC0AC29B7L, 0xC97C50DDL, 0x3F84D5B5L, 0xB5470917L,
    0x9216D5D9L, 0x8979FB1BL
};

static const unsigned long ORIG_S[4][256] = {
  { 0xD1310BA6L, 0x98DFB5ACL, 0x2FFD72DBL, 0xD01ADFB7L,
    0xB8E1AFEDL, 0x6A267E96L, 0xBA7C9045L, 0xF12C7F99L,
    0x24A19947L, 0xB3916CF7L, 0x0801F2E2L, 0x858EFC16L,
    0x636920D8L, 0x71574E69L, 0xA458FEA3L, 0xF4933D7EL,
    0x0D95748FL, 0x728EB658L, 0x718BCD58L, 0x82154AEEL,
    0x7B54A41DL, 0xC25A59B5L, 0x9C30D539L, 0x2AF26013L,
    0xC5D1B023L, 0x286085F0L, 0xCA417918L, 0xB8DB38EFL,
    0x8E79DCB0L, 0x603A180EL, 0x6C9E0E8BL, 0xB01E8A3EL,
    0xD71577C1L, 0xBD314B27L, 0x78AF2FDAL, 0x55605C60L,
    0xE65525F3L, 0xAA55AB94L, 0x57489862L, 0x63E81440L,
    0x55CA396AL, 0x2AAB10B6L, 0xB4CC5C34L, 0x1141E8CEL,
    0xA15486AFL, 0x7C72E993L, 0xB3EE1411L, 0x636FBC2AL,
    0x2BA9C55DL, 0x741831F6L, 0xCE5C3E16L, 0x9B87931EL,
    0xAFD6BA33L, 0x6C24CF5CL, 0x7A325381L, 0x28958677L,
    0x3B8F4898L, 0x6B4BB9AFL, 0xC4BFE81BL, 0x66282193L,
    0x61D809CCL, 0xFB21A991L, 0x487CAC60L, 0x5DEC8032L,
    0xEF845D5DL, 0xE98575B1L, 0xDC262302L, 0xEB651B88L,
    0x23893E81L, 0xD396ACC5L, 0x0F6D6FF3L, 0x83F44239L,
```

0x2E0B4482L, 0xA4842004L, 0x69C8F04AL, 0x9E1F9B5EL,
0x21C66842L, 0xF6E96C9AL, 0x670C9C61L, 0xABD388F0L,
0x6A51A0D2L, 0xD8542F68L, 0x960FA728L, 0xAB5133A3L,
0x6EEF0B6CL, 0x137A3BE4L, 0xBA3BF050L, 0x7EFB2A98L,
0xA1F1651DL, 0x39AF0176L, 0x66CA593EL, 0x82430E88L,
0x8CEE8619L, 0x456F9FB4L, 0x7D84A5C3L, 0x3B8B5EBEL,
0xE06F75D8L, 0x85C12073L, 0x401A449FL, 0x56C16AA6L,
0x4ED3AA62L, 0x363F7706L, 0x1BFEDF72L, 0x429B023DL,
0x37D0D724L, 0xD00A1248L, 0xDB0FEAD3L, 0x49F1C09BL,
0x075372C9L, 0x80991B7BL, 0x25D479D8L, 0xF6E8DEF7L,
0xE3FE501AL, 0xB6794C3BL, 0x976CE0BDL, 0x04C006BAL,
0xC1A94FB6L, 0x409F60C4L, 0x5E5C9EC2L, 0x196A2463L,
0x68FB6FAFL, 0x3E6C53B5L, 0x1339B2EBL, 0x3B52EC6FL,
0x6DFC511FL, 0x9B30952CL, 0xCC814544L, 0xAF5EBD09L,
0xBEE3D004L, 0xDE334AFDL, 0x660F2807L, 0x192E4BB3L,
0xC0CBA857L, 0x45C8740FL, 0xD20B5F39L, 0xB9D3FBDBL,
0x5579C0BDL, 0x1A60320AL, 0xD6A100C6L, 0x402C7279L,
0x679F25FEL, 0xFB1FA3CCL, 0x8EA5E9F8L, 0xDB3222F8L,
0x3C7516DFL, 0xFD616B15L, 0x2F501EC8L, 0xAD0552ABL,
0x323DB5FAL, 0xFD238760L, 0x53317B48L, 0x3E00DF82L,
0x9E5C57BBL, 0xCA6F8CA0L, 0x1A87562EL, 0xDF1769DBL,
0xD542A8F6L, 0x287EFFC3L, 0xAC6732C6L, 0x8C4F5573L,
0x695B27B0L, 0xBBCA58C8L, 0xE1FFA35DL, 0xB8F011A0L,
0x10FA3D98L, 0xFD2183B8L, 0x4AFCB56CL, 0x2DD1D35BL,
0x9A53E479L, 0xB6F84565L, 0xD28E49BCL, 0x4BFB9790L,
0xE1DDF2DAL, 0xA4CB7E33L, 0x62FB1341L, 0xCEE4C6E8L,
0xEF20CADAL, 0x36774C01L, 0xD07E9EFEL, 0x2BF11FB4L,
0x95DBDA4DL, 0xAE909198L, 0xEAAD8E71L, 0x6B93D5A0L,
0xD08ED1D0L, 0xAFC725E0L, 0x8E3C5B2FL, 0x8E7594B7L,
0x8FF6E2FBL, 0xF2122B64L, 0x8888B812L, 0x900DF01CL,

0x4FAD5EA0L, 0x688FC31CL, 0xD1CFF191L, 0xB3A8C1ADL,
0x2F2F2218L, 0xBE0E1777L, 0xEA752DFEL, 0x8B021FA1L,
0xE5A0CC0FL, 0xB56F74E8L, 0x18ACF3D6L, 0xCE89E299L,
0xB4A84FE0L, 0xFD13E0B7L, 0x7CC43B81L, 0xD2ADA8D9L,
0x165FA266L, 0x80957705L, 0x93CC7314L, 0x211A1477L,
0xE6AD2065L, 0x77B5FA86L, 0xC75442F5L, 0xFB9D35CFL,
0xEBCDAF0CL, 0x7B3E89A0L, 0xD6411BD3L, 0xAE1E7E49L,
0x00250E2DL, 0x2071B35EL, 0x226800BBL, 0x57B8E0AFL,
0x2464369BL, 0xF009B91EL, 0x5563911DL, 0x59DFA6AAL,
0x78C14389L, 0xD95A537FL, 0x207D5BA2L, 0x02E5B9C5L,
0x83260376L, 0x6295CFA9L, 0x11C81968L, 0x4E734A41L,
0xB3472DCAL, 0x7B14A94AL, 0x1B510052L, 0x9A532915L,
0xD60F573FL, 0xBC9BC6E4L, 0x2B60A476L, 0x81E67400L,
0x08BA6FB5L, 0x571BE91FL, 0xF296EC6BL, 0x2A0DD915L,
0xB6636521L, 0xE7B9F9B6L, 0xFF34052EL, 0xC5855664L,
0x53B02D5DL, 0xA99F8FA1L, 0x08BA4799L, 0x6E85076AL },
{ 0x4B7A70E9L, 0xB5B32944L, 0xDB75092EL, 0xC4192623L,
0xAD6EA6B0L, 0x49A7DF7DL, 0x9CEE60B8L, 0x8FEDB266L,
0xECAA8C71L, 0x699A17FFL, 0x5664526CL, 0xC2B19EE1L,
0x193602A5L, 0x75094C29L, 0xA0591340L, 0xE4183A3EL,
0x3F54989AL, 0x5B429D65L, 0x6B8FE4D6L, 0x99F73FD6L,
0xA1D29C07L, 0xEFE830F5L, 0x4D2D38E6L, 0xF0255DC1L,
0x4CDD2086L, 0x8470EB26L, 0x6382E9C6L, 0x021ECC5EL,
0x09686B3FL, 0x3EBAEFC9L, 0x3C971814L, 0x6B6A70A1L,
0x687F3584L, 0x52A0E286L, 0xB79C5305L, 0xAA500737L,
0x3E07841CL, 0x7FDEAE5CL, 0x8E7D44ECL, 0x5716F2B8L,
0xB03ADA37L, 0xF0500C0DL, 0xF01C1F04L, 0x0200B3FFL,
0xAE0CF51AL, 0x3CB574B2L, 0x25837A58L, 0xDC0921BDL,
0xD19113F9L, 0x7CA92FF6L, 0x94324773L, 0x22F54701L,
0x3AE5E581L, 0x37C2DADCL, 0xC8B57634L, 0x9AF3DDA7L,

0xA9446146L, 0x0FD0030EL, 0xECC8C73EL, 0xA4751E41L,
0xE238CD99L, 0x3BEA0E2FL, 0x3280BBA1L, 0x183EB331L,
0x4E548B38L, 0x4F6DB908L, 0x6F420D03L, 0xF60A04BFL,
0x2CB81290L, 0x24977C79L, 0x5679B072L, 0xBCAF89AFL,
0xDE9A771FL, 0xD9930810L, 0xB38BAE12L, 0xDCCF3F2EL,
0x5512721FL, 0x2E6B7124L, 0x501ADDE6L, 0x9F84CD87L,
0x7A584718L, 0x7408DA17L, 0xBC9F9ABCL, 0xE94B7D8CL,
0xEC7AEC3AL, 0xDB851DFAL, 0x63094366L, 0xC464C3D2L,
0xEF1C1847L, 0x3215D908L, 0xDD433B37L, 0x24C2BA16L,
0x12A14D43L, 0x2A65C451L, 0x50940002L, 0x133AE4DDL,
0x71DFF89EL, 0x10314E55L, 0x81AC77D6L, 0x5F11199BL,
0x043556F1L, 0xD7A3C76BL, 0x3C11183BL, 0x5924A509L,
0xF28FE6EDL, 0x97F1FBFAL, 0x9EBABF2CL, 0x1E153C6EL,
0x86E34570L, 0xEAE96FB1L, 0x860E5E0AL, 0x5A3E2AB3L,
0x771FE71CL, 0x4E3D06FAL, 0x2965DCB9L, 0x99E71D0FL,
0x803E89D6L, 0x5266C825L, 0x2E4CC978L, 0x9C10B36AL,
0xC6150EBAL, 0x94E2EA78L, 0xA5FC3C53L, 0x1E0A2DF4L,
0xF2F74EA7L, 0x361D2B3DL, 0x1939260FL, 0x19C27960L,
0x5223A708L, 0xF71312B6L, 0xEBADFE6EL, 0xEAC31F66L,
0xE3BC4595L, 0xA67BC883L, 0xB17F37D1L, 0x018CFF28L,
0xC332DDEFL, 0xBE6C5AA5L, 0x65582185L, 0x68AB9802L,
0xEECEA50FL, 0xDB2F953BL, 0x2AEF7DADL, 0x5B6E2F84L,
0x1521B628L, 0x29076170L, 0xECDD4775L, 0x619F1510L,
0x13CCA830L, 0xEB61BD96L, 0x0334FE1EL, 0xAA0363CFL,
0xB5735C90L, 0x4C70A239L, 0xD59E9E0BL, 0xCBAADE14L,
0xEECC86BCL, 0x60622CA7L, 0x9CAB5CABL, 0xB2F3846EL,
0x648B1EAFL, 0x19BDF0CAL, 0xA02369B9L, 0x655ABB50L,
0x40685A32L, 0x3C2AB4B3L, 0x319EE9D5L, 0xC021B8F7L,
0x9B540B19L, 0x875FA099L, 0x95F7997EL, 0x623D7DA8L,
0xF837889AL, 0x97E32D77L, 0x11ED935FL, 0x16681281L,

0x0E358829L, 0xC7E61FD6L, 0x96DEDFA1L, 0x7858BA99L,
0x57F584A5L, 0x1B227263L, 0x9B83C3FFL, 0x1AC24696L,
0xCDB30AEBL, 0x532E3054L, 0x8FD948E4L, 0x6DBC3128L,
0x58EBF2EFL, 0x34C6FFEAL, 0xFE28ED61L, 0xEE7C3C73L,
0x5D4A14D9L, 0xE864B7E3L, 0x42105D14L, 0x203E13E0L,
0x45EEE2B6L, 0xA3AAABEAL, 0xDB6C4F15L, 0xFACB4FD0L,
0xC742F442L, 0xEF6ABBB5L, 0x654F3B1DL, 0x41CD2105L,
0xD81E799EL, 0x86854DC7L, 0xE44B476AL, 0x3D816250L,
0xCF62A1F2L, 0x5B8D2646L, 0xFC8883A0L, 0xC1C7B6A3L,
0x7F1524C3L, 0x69CB7492L, 0x47848A0BL, 0x5692B285L,
0x095BBF00L, 0xAD19489DL, 0x1462B174L, 0x23820E00L,
0x58428D2AL, 0x0C55F5EAL, 0x1DADF43EL, 0x233F7061L,
0x3372F092L, 0x8D937E41L, 0xD65FECF1L, 0x6C223BDBL,
0x7CDE3759L, 0xCBEE7460L, 0x4085F2A7L, 0xCE77326EL,
0xA6078084L, 0x19F8509EL, 0xE8EFD855L, 0x61D99735L,
0xA969A7AAL, 0xC50C06C2L, 0x5A04ABFCL, 0x800BCADCL,
0x9E447A2EL, 0xC3453484L, 0xFDD56705L, 0x0E1E9EC9L,
0xDB73DBD3L, 0x105588CDL, 0x675FDA79L, 0xE3674340L,
0xC5C43465L, 0x713E38D8L, 0x3D28F89EL, 0xF16DFF20L,
0x153E21E7L, 0x8FB03D4AL, 0xE6E39F2BL, 0xDB83ADF7L },
{ 0xE93D5A68L, 0x948140F7L, 0xF64C261CL, 0x94692934L,
0x411520F7L, 0x7602D4F7L, 0xBCF46B2EL, 0xD4A20068L,
0xD4082471L, 0x3320F46AL, 0x43B7D4B7L, 0x500061AFL,
0x1E39F62EL, 0x97244546L, 0x14214F74L, 0xBF8B8840L,
0x4D95FC1DL, 0x96B591AFL, 0x70F4DDD3L, 0x66A02F45L,
0xBFBC09ECL, 0x03BD9785L, 0x7FAC6DD0L, 0x31CB8504L,
0x96EB27B3L, 0x55FD3941L, 0xDA2547E6L, 0xABCA0A9AL,
0x28507825L, 0x530429F4L, 0x0A2C86DAL, 0xE9B66DFBL,
0x68DC1462L, 0xD7486900L, 0x680EC0A4L, 0x27A18DEEL,
0x4F3FFEA2L, 0xE887AD8CL, 0xB58CE006L, 0x7AF4D6B6L,

0xAACE1E7CL, 0xD3375FECL, 0xCE78A399L, 0x406B2A42L,
0x20FE9E35L, 0xD9F385B9L, 0xEE39D7ABL, 0x3B124E8BL,
0x1DC9FAF7L, 0x4B6D1856L, 0x26A36631L, 0xEAE397B2L,
0x3A6EFA74L, 0xDD5B4332L, 0x6841E7F7L, 0xCA7820FBL,
0xFB0AF54EL, 0xD8FEB397L, 0x454056ACL, 0xBA489527L,
0x55533A3AL, 0x20838D87L, 0xFE6BA9B7L, 0xD096954BL,
0x55A867BCL, 0xA1159A58L, 0xCCA92963L, 0x99E1DB33L,
0xA62A4A56L, 0x3F3125F9L, 0x5EF47E1CL, 0x9029317CL,
0xFDF8E802L, 0x04272F70L, 0x80BB155CL, 0x05282CE3L,
0x95C11548L, 0xE4C66D22L, 0x48C1133FL, 0xC70F86DCL,
0x07F9C9EEL, 0x41041F0FL, 0x404779A4L, 0x5D886E17L,
0x325F51EBL, 0xD59BC0D1L, 0xF2BCC18FL, 0x41113564L,
0x257B7834L, 0x602A9C60L, 0xDFF8E8A3L, 0x1F636C1BL,
0x0E12B4C2L, 0x02E1329EL, 0xAF664FD1L, 0xCAD18115L,
0x6B2395E0L, 0x333E92E1L, 0x3B240B62L, 0xEEBEB922L,
0x85B2A20EL, 0xE6BA0D99L, 0xDE720C8CL, 0x2DA2F728L,
0xD0127845L, 0x95B794FDL, 0x647D0862L, 0xE7CCF5F0L,
0x5449A36FL, 0x877D48FAL, 0xC39DFD27L, 0xF33E8D1EL,
0x0A476341L, 0x992EFF74L, 0x3A6F6EABL, 0xF4F8FD37L,
0xA812DC60L, 0xA1EBDDF8L, 0x991BE14CL, 0xDB6E6B0DL,
0xC67B5510L, 0x6D672C37L, 0x2765D43BL, 0xDCD0E804L,
0xF1290DC7L, 0xCC00FFA3L, 0xB5390F92L, 0x690FED0BL,
0x667B9FFBL, 0xCEDB7D9CL, 0xA091CF0BL, 0xD9155EA3L,
0xBB132F88L, 0x515BAD24L, 0x7B9479BFL, 0x763BD6EBL,
0x37392EB3L, 0xCC115979L, 0x8026E297L, 0xF42E312DL,
0x6842ADA7L, 0xC66A2B3BL, 0x12754CCCL, 0x782EF11CL,
0x6A124237L, 0xB79251E7L, 0x06A1BBE6L, 0x4BFB6350L,
0x1A6B1018L, 0x11CAEDFAL, 0x3D25BDD8L, 0xE2E1C3C9L,
0x44421659L, 0x0A121386L, 0xD90CEC6EL, 0xD5ABEA2AL,
0x64AF674EL, 0xDA86A85FL, 0xBEBFE988L, 0x64E4C3FEL,

0x9DBC8057L, 0xF0F7C086L, 0x60787BF8L, 0x6003604DL,
0xD1FD8346L, 0xF6381FB0L, 0x7745AE04L, 0xD736FCCCL,
0x83426B33L, 0xF01EAB71L, 0xB0804187L, 0x3C005E5FL,
0x77A057BEL, 0xBDE8AE24L, 0x55464299L, 0xBF582E61L,
0x4E58F48FL, 0xF2DDFDA2L, 0xF474EF38L, 0x8789BDC2L,
0x5366F9C3L, 0xC8B38E74L, 0xB475F255L, 0x46FCD9B9L,
0x7AEB2661L, 0x8B1DDF84L, 0x846A0E79L, 0x915F95E2L,
0x466E598EL, 0x20B45770L, 0x8CD55591L, 0xC902DE4CL,
0xB90BACE1L, 0xBB8205D0L, 0x11A86248L, 0x7574A99EL,
0xB77F19B6L, 0xE0A9DC09L, 0x662D09A1L, 0xC4324633L,
0xE85A1F02L, 0x09F0BE8CL, 0x4A99A025L, 0x1D6EFE10L,
0x1AB93D1DL, 0x0BA5A4DFL, 0xA186F20FL, 0x2868F169L,
0xDCB7DA83L, 0x573906FEL, 0xA1E2CE9BL, 0x4FCD7F52L,
0x50115E01L, 0xA70683FAL, 0xA002B5C4L, 0x0DE6D027L,
0x9AF88C27L, 0x773F8641L, 0xC3604C06L, 0x61A806B5L,
0xF0177A28L, 0xC0F586E0L, 0x006058AAL, 0x30DC7D62L,
0x11E69ED7L, 0x2338EA63L, 0x53C2DD94L, 0xC2C21634L,
0xBBCBEE56L, 0x90BCB6DEL, 0xEBFC7DA1L, 0xCE591D76L,
0x6F05E409L, 0x4B7C0188L, 0x39720A3DL, 0x7C927C24L,
0x86E3725FL, 0x724D9DB9L, 0x1AC15BB4L, 0xD39EB8FCL,
0xED545578L, 0x08FCA5B5L, 0xD83D7CD3L, 0x4DAD0FC4L,
0x1E50EF5EL, 0xB161E6F8L, 0xA28514D9L, 0x6C51133CL,
0x6FD5C7E7L, 0x56E14EC4L, 0x362ABFCEL, 0xDDC6C837L,
0xD79A3234L, 0x92638212L, 0x670EFA8EL, 0x406000E0L },
{ 0x3A39CE37L, 0xD3FAF5CFL, 0xABC27737L, 0x5AC52D1BL,
0x5CB0679EL, 0x4FA33742L, 0xD3822740L, 0x99BC9BBEL,
0xD5118E9DL, 0xBF0F7315L, 0xD62D1C7EL, 0xC700C47BL,
0xB78C1B6BL, 0x21A19045L, 0xB26EB1BEL, 0x6A366EB4L,
0x5748AB2FL, 0xBC946E79L, 0xC6A376D2L, 0x6549C2C8L,
0x530FF8EEL, 0x468DDE7DL, 0xD5730A1DL, 0x4CD04DC6L,

0x2939BBDBL, 0xA9BA4650L, 0xAC9526E8L, 0xBE5EE304L,
0xA1FAD5F0L, 0x6A2D519AL, 0x63EF8CE2L, 0x9A86EE22L,
0xC089C2B8L, 0x43242EF6L, 0xA51E03AAL, 0x9CF2D0A4L,
0x83C061BAL, 0x9BE96A4DL, 0x8FE51550L, 0xBA645BD6L,
0x2826A2F9L, 0xA73A3AE1L, 0x4BA99586L, 0xEF5562E9L,
0xC72FEFD3L, 0xF752F7DAL, 0x3F046F69L, 0x77FA0A59L,
0x80E4A915L, 0x87B08601L, 0x9B09E6ADL, 0x3B3EE593L,
0xE990FD5AL, 0x9E34D797L, 0x2CF0B7D9L, 0x022B8B51L,
0x96D5AC3AL, 0x017DA67DL, 0xD1CF3ED6L, 0x7C7D2D28L,
0x1F9F25CFL, 0xADF2B89BL, 0x5AD6B472L, 0x5A88F54CL,
0xE029AC71L, 0xE019A5E6L, 0x47B0ACFDL, 0xED93FA9BL,
0xE8D3C48DL, 0x283B57CCL, 0xF8D56629L, 0x79132E28L,
0x785F0191L, 0xED756055L, 0xF7960E44L, 0xE3D35E8CL,
0x15056DD4L, 0x88F46DBAL, 0x03A16125L, 0x0564F0BDL,
0xC3EB9E15L, 0x3C9057A2L, 0x97271AECL, 0xA93A072AL,
0x1B3F6D9BL, 0x1E6321F5L, 0xF59C66FBL, 0x26DCF319L,
0x7533D928L, 0xB155FDF5L, 0x03563482L, 0x8ABA3CBBL,
0x28517711L, 0xC20AD9F8L, 0xABCC5167L, 0xCCAD925FL,
0x4DE81751L, 0x3830DC8EL, 0x379D5862L, 0x9320F991L,
0xEA7A90C2L, 0xFB3E7BCEL, 0x5121CE64L, 0x774FBE32L,
0xA8B6E37EL, 0xC3293D46L, 0x48DE5369L, 0x6413E680L,
0xA2AE0810L, 0xDD6DB224L, 0x69852DFDL, 0x09072166L,
0xB39A460AL, 0x6445C0DDL, 0x586CDECFL, 0x1C20C8AEL,
0x5BBEF7DDL, 0x1B588D40L, 0xCCD2017FL, 0x6BB4E3BBL,
0xDDA26A7EL, 0x3A59FF45L, 0x3E350A44L, 0xBCB4CDD5L,
0x72EACEA8L, 0xFA6484BBL, 0x8D6612AEL, 0xBF3C6F47L,
0xD29BE463L, 0x542F5D9EL, 0xAEC2771BL, 0xF64E6370L,
0x740E0D8DL, 0xE75B1357L, 0xF8721671L, 0xAF537D5DL,
0x4040CB08L, 0x4EB4E2CCL, 0x34D2466AL, 0x0115AF84L,
0xE1B00428L, 0x95983A1DL, 0x06B89FB4L, 0xCE6EA048L,

```
0x6F3F3B82L, 0x3520AB82L, 0x011A1D4BL, 0x277227F8L,
0x611560B1L, 0xE7933FDCL, 0xBB3A792BL, 0x344525BDL,
0xA08839E1L, 0x51CE794BL, 0x2F32C9B7L, 0xA01FBAC9L,
0xE01CC87EL, 0xBCC7D1F6L, 0xCF0111C3L, 0xA1E8AAC7L,
0x1A908749L, 0xD44FBD9AL, 0xD0DADECBL, 0xD50ADA38L,
0x0339C32AL, 0xC6913667L, 0x8DF9317CL, 0xE0B12B4FL,
0xF79E59B7L, 0x43F5BB3AL, 0xF2D519FFL, 0x27D9459CL,
0xBF97222CL, 0x15E6FC2AL, 0x0F91FC71L, 0x9B941525L,
0xFAE59361L, 0xCEB69CEBL, 0xC2A86459L, 0x12BAA8D1L,
0xB6C1075EL, 0xE3056A0CL, 0x10D25065L, 0xCB03A442L,
0xE0EC6E0EL, 0x1698DB3BL, 0x4C98A0BEL, 0x3278E964L,
0x9F1F9532L, 0xE0D392DFL, 0xD3A0342BL, 0x8971F21EL,
0x1B0A7441L, 0x4BA3348CL, 0xC5BE7120L, 0xC37632D8L,
0xDF359F8DL, 0x9B992F2EL, 0xE60B6F47L, 0x0FE3F11DL,
0xE54CDA54L, 0x1EDAD891L, 0xCE6279CFL, 0xCD3E7E6FL,
0x1618B166L, 0xFD2C1D05L, 0x848FD2C5L, 0xF6FB2299L,
0xF523F357L, 0xA6327623L, 0x93A83531L, 0x56CCCD02L,
0xACF08162L, 0x5A75EBB5L, 0x6E163697L, 0x88D273CCL,
0xDE966292L, 0x81B949D0L, 0x4C50901BL, 0x71C65614L,
0xE6C6C7BDL, 0x327A140AL, 0x45E1D006L, 0xC3F27B9AL,
0xC9AA53FDL, 0x62A80F00L, 0xBB25BFE2L, 0x35BDD2F6L,
0x71126905L, 0xB2040222L, 0xB6CBCF7CL, 0xCD769C2BL,
0x53113EC0L, 0x1640E3D3L, 0x38ABBD60L, 0x2547ADF0L,
0xBA38209CL, 0xF746CE76L, 0x77AFA1C5L, 0x20756060L,
0x85CBFE4EL, 0x8AE88DD8L, 0x7AAAF9B0L, 0x4CF9AA7EL,
0x1948C25CL, 0x02FB8A8CL, 0x01C36AE4L, 0xD6EBE1F9L,
0x90D4F869L, 0xA65CDEA0L, 0x3F09252DL, 0xC208E69FL,
0xB74E6132L, 0xCE77E25BL, 0x578FDFE3L, 0x3AC372E6L }
};
```

REFERENCES:

1) An Implementation of the Blowfish Cryptosystem by Russell K. Meyers and Ahmed H. Desoky, IEEE 2008.

2) B. Schneier. The Blowfish Encryption Algorithm.

3) Speed Comparisons of Block Ciphers on a Pentium.

4) B. Schneier, "Blowfish: One Year Later."

5) "Blowfish (Cipher)." Wikipedia http://en.wikipedia.org/wiki/Blowfish_(cipher)

6) Y.-K. Lai and Y.-C. Shu, "VLSI architecture design and implementation for BLOWFISH block cipher with secure modes of operation," *IEEE International Symposium on Circuits and Systems*, 2001.