IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

FPGA Implementation of Secret Communication

K.Thirunavukarasu[*1], Dr. T.V.U. Kiran Kumar[2], Ms. S. Arulselvi[3]

[*1, 2,3] Department of Electronics and Communication Engineering, Bharath University, Selaiyur, Chennai, India

kthiru_sk@rediffmail.com

Abstract

The aim of this project is to communicate the data Secretly using DES Algorithm,(i.e) we first send the data (plain t ext) and the key which is of 64 bit into the encryption process.

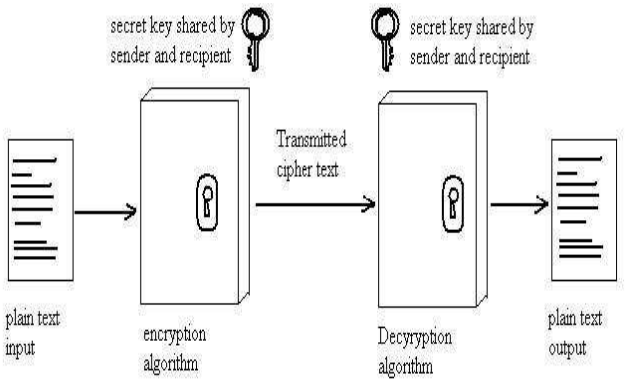
The output of this process will be cipher text. This cipher text is then fed into the decryption process and then the data (plain text) is got as output. The main aim is that since we add the key and shuffle the data it is very hard for the unknown person to find o ut the original data. Since for each key there will be a change in t he cipher text and so the person has to know the key in order to find out the original data.

Introduction

Cryptography

The branch of cryptology dealing with the design of algorithms for encryption and decryption, intended to ensure the secrecy and/or authenticity of messages.

An original message is known as the plain text, while the coded message is called the cipher text. The process of converting the plain text to cipher text is known as enciphering or encryption; restoring the plain text from the cipher text is deciphering or decrypt ion. The many schemes used for enciphering constitute the area of study known as cryptography. Such a scheme is known as a cryptographic system or a cipher. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of cryptanalysis. The crypt analysis is what the lay persons calls“ breaking the code”. Cryptography and cryptanalysis are together called as cryptology.



Encryption

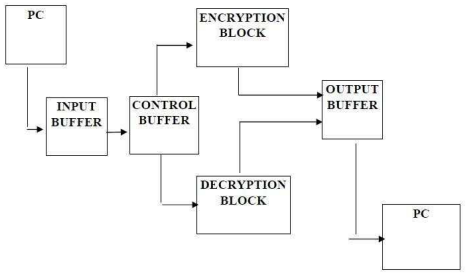
The original intelligible message or data that is fed into the algorithm is the input. Encryption algorithm performs varies substitutions and transformations on the plain text. Secret key is also giv en input to encryption algorithm. The key is a value independent of t he plain text. The algorithm reproduces the different output depending on the specific key being used at the time. The exact substitutions and transformation Performed by the algorithm depends on the key.

The output depends on the plain text and the secret key. For a given message, two different keys will produce two different cipher texts. The cipher text is an apparently random stream of data, as it stands, is unintelligible.

Decryption

This is essentially the encryption algorithm run in reverse. It takes the cipher text and the secret key and produces the original plain text.

Block Diagram



Algorithm Review: (DES)

The DES block cipher algorithm was developed by researchers at IBM and was fine tuned by government agencies, the National Security Agency (NSA) and the National Institute of Standards and Technology (NIST). The American National Standards Institute (ANSI) adopted DES as the federal standard for encryption of commercial and sensitive data. This is defined in Federal information Processing Standards (FIPS 46, 1977) published by NIST [1], [2].

The DES algorithm has a regular structure that lends itself to pipelining and a simple data manipulation to permit fast operations.DES is asymmetric encryption algorithm where the same key is used for both encryption and decryption. DES takes a 64 bit key and a 64bit block of data as inputs, and outputs 64bits of encrypted data. The actual key is only 56bits and the remaining bits, i.e., the least significant bit (LSB) in every byte can be used as parity. The same basic design can be used for both encryption and d encryption.

The Data Encryption Standard (DES)shall consist of the following Data Encryption Algorithm(DES) and Triple Data Encryption Algorithm(TDEA, as described in ANSIX9.52). These devices shal l be designed in such away that they may be used in a computer system or network to provide cryptographic protection to binary coded data. The method of implementation will depend on the application and environment. The devices shall be implemented in such away that they may be tested and validated as accurately performing the transformations specified in the following algorithms.

Data Encryption Algorithm

The algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64 bit key1. Deciphering must be accomplished by using the same key as for enciphering, b ut with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process and R of bits, LR denotes the block consisting of the bits of L followed by the bits of R. Since concatenation is associative, B1B2...B8,for example, denotes the block consisting of the bits of B1 followed by the bits ofB2...followed by the bits of B8.

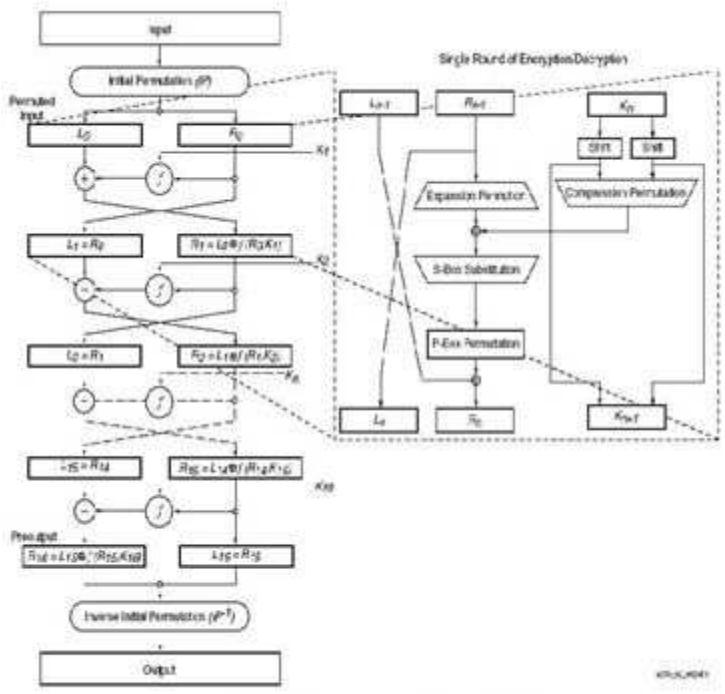


Figure 3.1 – Data Encryption Algorithm

A description of the computation is given first, along with details as to how the algorithm is used for encipherment. Next, the u se of the algorithm for deciphermentis described. Finally, a definition of the cipher function f is given in terms of primitive functions that are called the selection functions Si and the permutation function P .Si, P and KS of the algorithm are contained in Appendix.

The following notation is convenient: Given two blocks L and R of bits, LR denotes the block consisting of the bits of L follow ed by the bits of R. Since concatenation is associative, B1B2...B8,for example, denotes the block consisting of the bits of B1 followed by the bits ofB2...followed by the bits of B8.

Enciphering

A sketch of the enciphering computation is given in Figure 3.2. The 64 bits of the input block to been ciphered are first subjected to the following permutation, called the initial permutation IP:

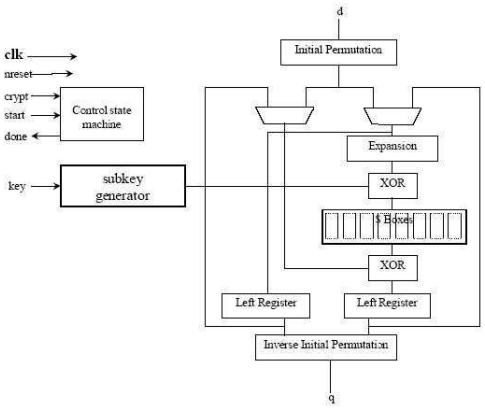


Figure 3.2 Enciphering

58	*50*	*42*	*34*	*26*	*18*	*10*	*2*
60	*52*	*44*	*36*	*28*	*20*	*12*	*4*
62	*54*	*46*	*38*	*30*	*22*	*14*	*6*
64	*56*	*48*	*40*	*32*	*24*	*16*	*8*
57	*49*	*41*	*33*	*25*	*17*	*9*	*1*
59	*51*	*43*	*35*	*27*	*19*	*11*	*3*
61	*53*	*45*	*37*	*29*	*21*	*13*	*5*
63	*55*	*47*	*39*	*31*	*23*	*15*	*7*

Table 3.1 – Initial permutation

That is the permuted input has bit 58 of the input as its firs t bit, bit 50 as its second bit, and so on with bit 7 as its last bit. The permuted input block is then the input to a complex key dependent computation described below. The output of that computation, called the preoutput, is then subjected to the following permutation, which is t he inverse of the initial permutation:

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Table 3.2 – Inverse initialpermutation

That is, the output of the algorithm has bit 40 of the preout put block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preout put block is the last bit of the output.

The computation which uses the permuted input block as its input to produce the preoutput block consists, but for a final inter change of blocks, of 16 iterations of a calculation that is described below in terms of the cipher function *f* which operates on two blocks, on e of 32 bits and one of 48 bits, and produces a block of 32 bits.

Let the 64 bits of the input block to an iteration consist of a 32 bit block *L* followed by a 32 bit block *R*. Using the notation de fined in the introduction, the input block is then *LR*. Let *K* be a bloc k of 48 bits chosen from the 64 bit key. Then the output *L'R'* of an ite ration with input *LR* is defined by:

$$L' = R$$
$$R' = L \oplus F(R, K)$$

Where $\oplus$ denotes bit by bit addition modulo 2.

As remarked before, the input of the first iteration of the calculation is the permuted input block. If *L'R'* is the output of the16th iteration then *R'L'* is the preoutput block. At each iteration a different block *K* of key bits is chosen from the 64 bit key designated by *KE Y* With more notations we can describe the iterations of the computation in more detail. Let *KS* be a function, which takes an integer *n* in t he range from1 to16 and a 64 bit block *KEY* as input and yields as o utput a 48 bit block *Kn* that is a permuted selection of bits from *KEY*.

That isKn=KS (n, KEY) 3.2

With *Kn* determined by the bits in 48 distinct bit positions of *KEY*. *KS* is called the key schedule because the block *K* used in the *n*'th iteration of (1)is the block *Kn* determined by (2).

As before, let the permuted input block be *LR*. Finally, let L ()and R () be respectively L and Rand let Ln and Rn be respectively L'and R'of(1) when *L* and *R* are respectively Ln-
1 and Rn 1and K is Kn; that is, when n is in the range from 1 to 16,

$$L_N = R_{N-1} \quad 3.3$$
$$R_N = L_{N-1} \oplus F(R_{N-1}, K_N)$$

The preoutput block is then *R16L16*. The key schedule K S of the algorithm is described in detail in the Appendix. The key schedule produces the16 *Kn*, which are required for the algorithm.

Deciphering

The permutation *IP -1* applied to the preoutput block is the inverse of the initial permutation *IP* applied to the input. Further, from (1) it follows that:

$$R = L' \; 3.4$$
$$L = R' \; F(L', K)$$

Consequently, to *decipher* it is only necessary to apply the *very same algorithm to anenciphered* message block, taking care that teach iteration of the computation *the same block of key bits* K *is used* during decipherment as was used during the encipherment of th e block. Using the notation of the previous section, this can be expressed by the equations:

$$R_{N-1} = L_N$$
$$L_{N-1} = R_N \; F(L_N, K_N)$$

Where now *R16L16* is the permuted input block for the deciphering calculation and *L0R0* is the preoutput block. That is, for the decipherment calculation with *R16L16* as the permuted input, *K16* is used in the first iteration, *K15* in the second, and so on,with*K1* used in the 16 titration.

The cipher function

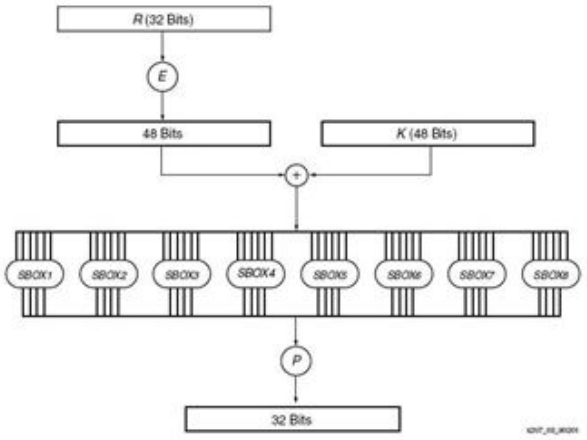


Figure 3.3 A sketch of the calculation of*f(R,K)*

The primitive function *P* is:

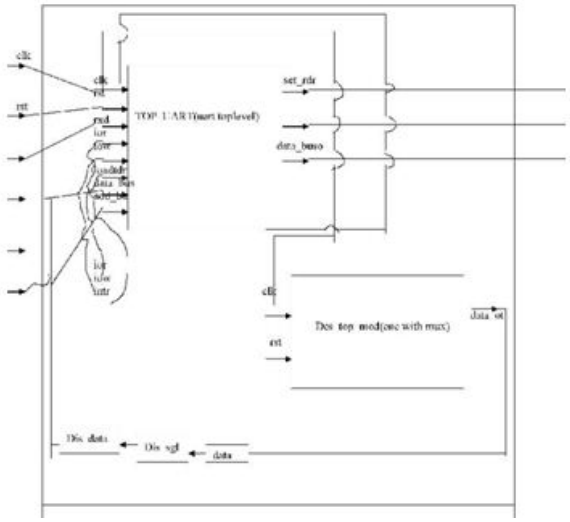
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Table 3.7 Primitive function

Recall that *Kn*, for $1 \le n \le 16$, is the block of 48 bits in (2) of the algorithm. Hence, to describe *KS*, it is sufficient to describe th e calculation of *Kn* from *KEY* for *n* =1, 2,..., 16. That calculation is illustrated in Figure. To complete the definition of *KS* it is therefore sufficient to describe the two permuted choices, as well as the schedule of left shifts. One bit in each 8 bit byte of the *KEY* may be utilized for error detection in key generation, distribution and storage. Bits

8, 16,..., 64 are for use in assuring that each byte is of odd parity.

Blocks With Respect to the Coding: Encryption with Uart

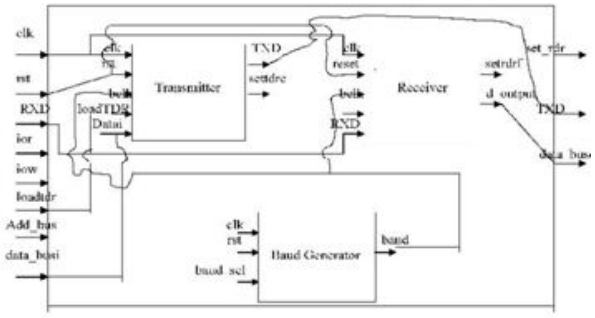


In this block we are combining uart with encryption in order to set the output in real time.

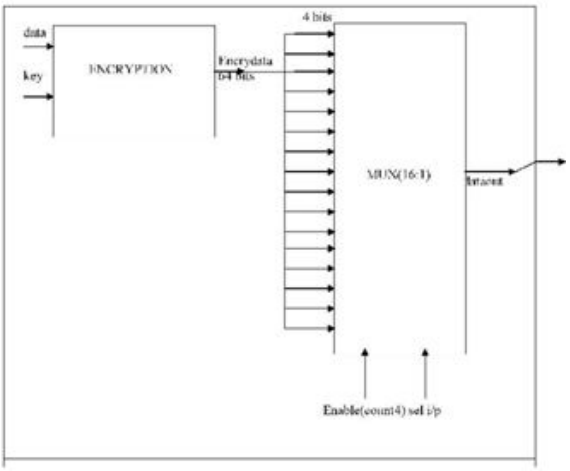
THE UART Block

This block consists of the transmitter, receiver, Baudrate generator and Synchronizer block. With the help of this we can see the output in the

pc.

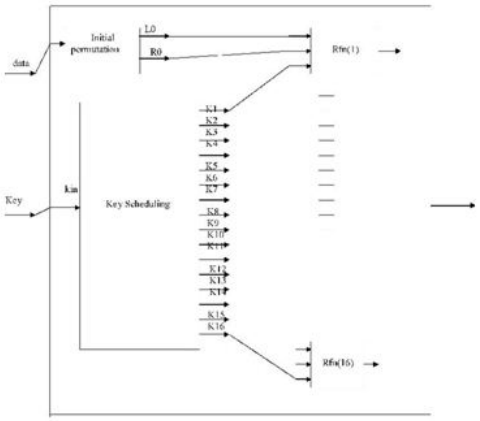


DES_TOP_MOD (encryption with mux)



This block consists of both encryption and mux. The output of the encryption block which is the cipher text is fed to a 16:1 mux with each input 4 bits. The selection line is enabled using a counter the and an enable signal is also used which is activated by using the4 bit of the counter. The output of this is converted to ascii value and t hen given to UART.

Encryption Block



This is the block where the algorithm is being implemented. That i s the DES algorithm steps are being implemented here.

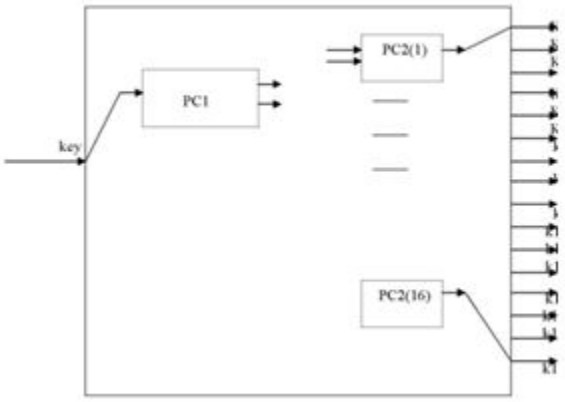
Initial Permutation

First the 64 bit data is being given to the initial permutation block and the data is being split into L0 and R0 with 32 bits. It is done with the help of a table and at the end of this process we will getL0 and R0 which is 32 bits each.

Key Scheduling

In this block the 64 bit key is given as input and this is again given to PC1 Which gives output as l0 and r0 with each 28 bi ts. This is again given to another block called PC2 which gives the output as 48 bits. So when we give 64 bits as input to the key scheduling block we get the output as 48 bits.

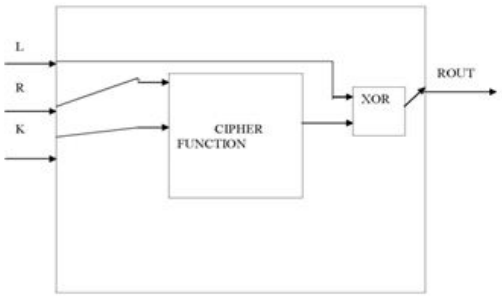
Key Scheduling Block



Round Function

This function takes k,l,r as input. Then r and k is given as input to the cipher function block. The output of the cipher function is again xored with the l0 and that is given as output to the rfn block.

Round Function Block



Way to Run the Code

First generate the hex file with the given code. Then download the file into the kit with the help of the sands prompt. After t he hex file is being loaded then change the cable from programmable port to serial port. Go to output and select port settings and then change the baud rate from 19200 to 9600 and then enable the clock. You can see the encrypted output on the pc. Same way when load the hex file for decryption the same procedure should be followed and this time will get the decrypted output. There should a separate hex file for both encryption and encryption.

Troubleshooting

In simulation we did not face any problem. But in synthesis we faced problem while interfacing it with uart. We had many
procedures to do. That is dividing the 64 bit output from the encryption block using a 16:1 mux. Then that is converted into its respective ascii value and then fed into the uart. Then output is taken from the uart.

Conclusion

In this DES algorithms the data and the key is of 64 bit This can be increased by using the AES Algorithm or ADES or APES Algorithm. But in AES Algorithm the data is of 128 bits. So as a result of which DES is implemented in real time and AES is implemented in simulation level. It is because of the gate counts that we were no t able to implement the AES algorithm in synthesis level. So in future if we use a higher version of IC so that we can also implement AES algorithm in synthesis level.

Datasheet to See the Output in the Next PC

When u want to see the output in the next pc first download the hex file into the kit. After that remove the cable from the programmable port and the connect it to the serial port. The other end of th e cable is connected to the next pc. Then open the sands prompt in that pc and then enable the clock

This Part of the datasheet is on Hardware and Software requirement specifications to get started with the FPGA trainer Kit.

The kit has the following Deliverables:
1. FPGA Kit in a box containing inbuilt units
XILINXXC2S150 FPGA Development Platform
4 x 3 Keypad Power Supply

2. Accessories
Programming Cable
Power Cord
40 Pin FRC Cable
FPGA Trainer Kit User Manual
FPGA/CPLD Development platform Installation CD consisting of the following:
-
FPGA/CPLD Development platform for programming the FPGA device. Sample Programs Kit Testing Programs-
PDF files for Data Sheet of Spartan IIFPGA, ADC0804, DAC 080 8 XILINX Web pack

Hardware and Software Requirements

Hardware Component: IBM -compatible Pentium Class Machine Monitor : Colour Operating
System : Windows 2000 or Windows XP
CD Drive : CD drive on system for Installation Ports : Serial Port to program the device System Memory : Minimum 256 MBS emulation Tool : Model SIM or any other simulator Synthesis Tool : XILINX Alliance, Foundation series or XILINX
Web Pack Programming Tool : FPGA/CPLD Development Platform

Software Installation Procedure

Insert the Installation CD in the CD drive. The installation automatically starts when the CD is inserted After the software is installed, message “Installation Completed” is displayed.

FPGA/CPLD Development Platform

Our windows based software is used to configure the FPGA trainer kit. This is also used to program any digital circuits in the FPGA, which is built using the HDLs (Hardware Description Language) such as VHDL(Very High Speed Inte grated Circuits Hardware Description Language) and/or Verilog. Our software configures the FPGA device and checks only the functionality of the same. Support microcontroller interface and an FPGA serial communication. This avoids costly downloading cables and software. The GUI is user friendly and aimed to avoid overheads.

PIN Assignment

HEADER / JUMPER SETTINGS HEADER Details of BERG JP7