ENCRYPTION AND DECRYPTION USING DES ALGORITHM

EENG-5550

Hardware Design Methodologies for ASICs and FPGAs

Submitted to: Dr. Gayatri Mehta

Team members:

Sandesh Hulikal Somashekhar 11153118
Guru Prasanna Srinivasan 11153119

TABLE OF CONTENTS

I INTRODUCTION

1.1 What is Cryptography?

Cryptography is both the study and practice of the techniques used to communicate and/or store information or data privately and securely, without being intercepted by hackers or any third party.

Four main objectives of modern cryptographic techniques:

- Confidentiality
- Integrity
- Non-Repudiation
- Authentication

Types of Cryptography:

1.1.1 Asymmetric Cryptography

In this fashion of cryptography, two different keys are used for encryption and decryption. Every user has to have both the public key and the private key. The private key is kept secret at all times, but the public key may be freely distributed.

1.1.2 Symmetric Cryptography

In this fashion of cryptography, the same key is used for both encryption and decryption. The sender and receiver must already have a shared key that is known to both.

DES algorithm comes under Symmetric Cipher (Symmetric key) cryptography.

1.2 DES Algorithm

It is a block cipher, which encrypts data in 64 bit blocks. At one end of the algorithm, we input a 64 bit block of data known as plain text, on which many operations are done to give a 64 bit cipher text at the other end of the algorithm. Since this is a symmetric key algorithm same key is used for both encryption and decryption.

The length of the key is 64 bit, out of which the 8th bit (LSB) of every block is used as a parity bit. This the effective key length is 56 bits.

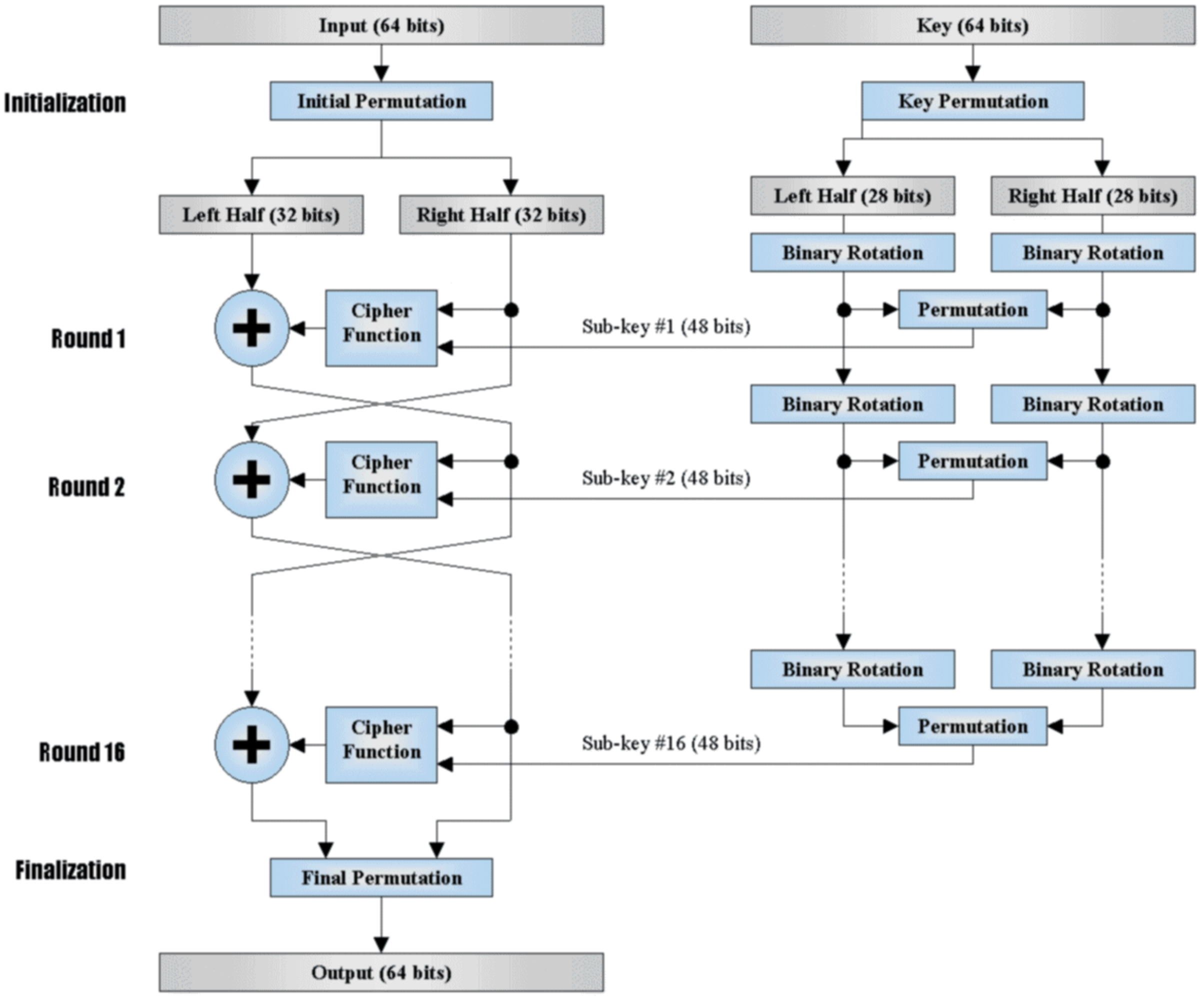
The singe block of DES is a combination of substitution followed by permutation, performed on the plain text based on the key. This is known as a round. DES is iterated 16 times in the same fashion on the block of plain text.

The algorithm uses only standard arithmetic and logical operations on numbers of 64 bits at most. The DES algorithm is the basis for encryption standards such as 2DES and 3DES.

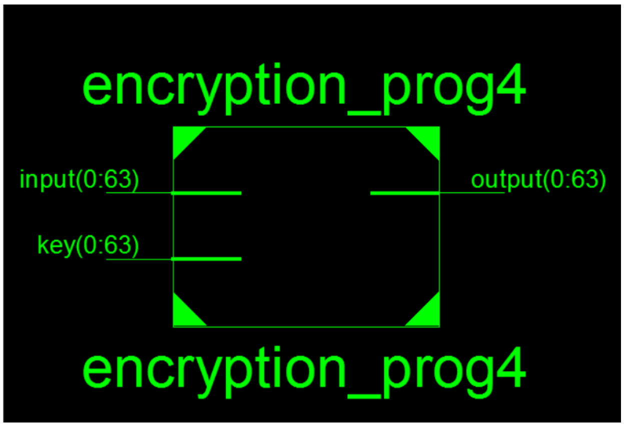
The simplicity of DES made it useful in a variety of embedded systems, smart cards, SIM cards and network devices requiring encryption like modems, set-top boxes and routers.

II BLOCK DIAGRAM

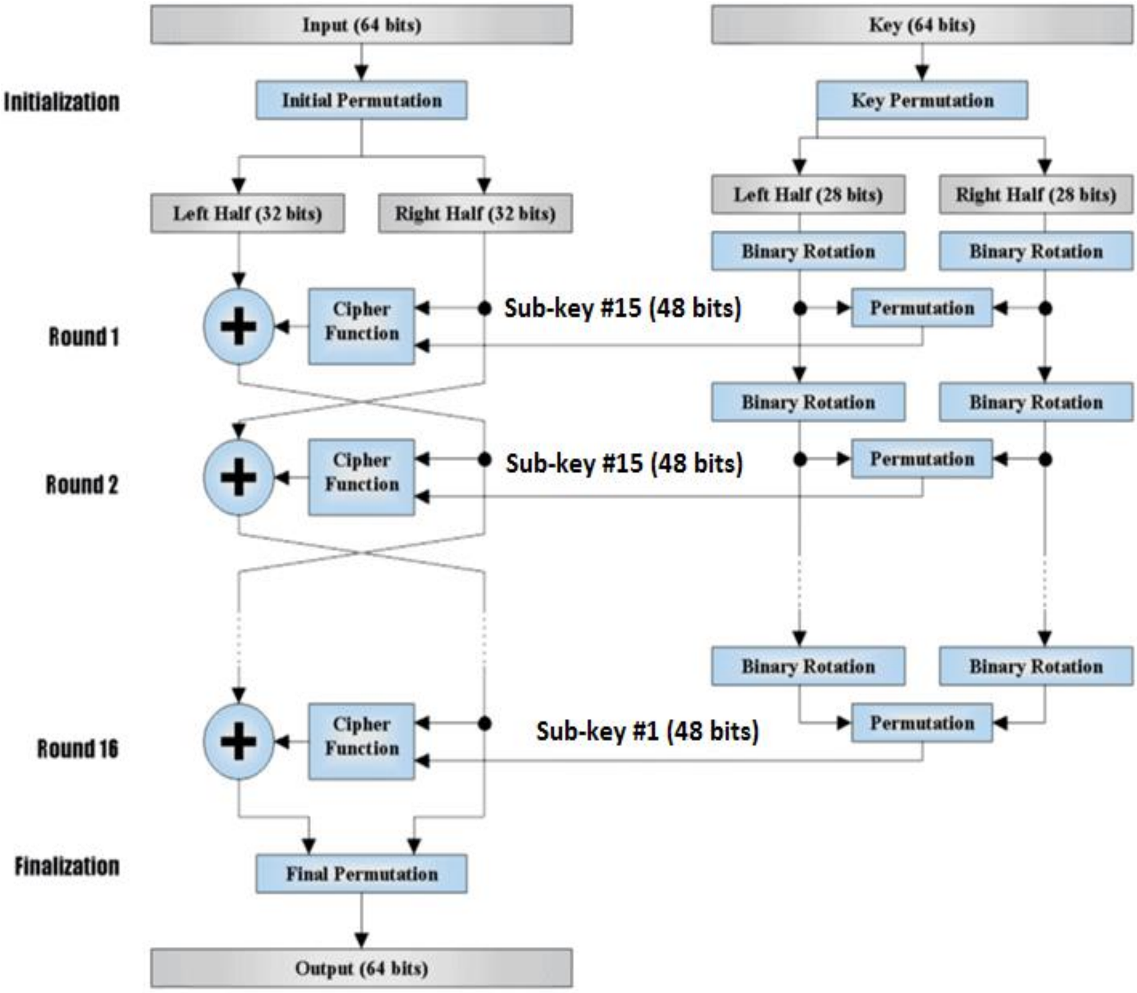
2.1 Encryption



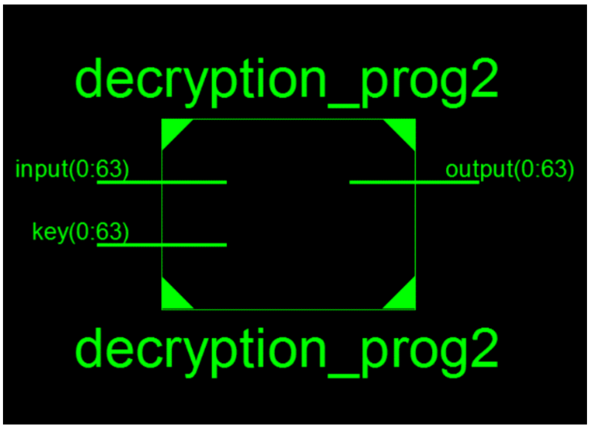
2.1.1 RTL Schematic Encryption



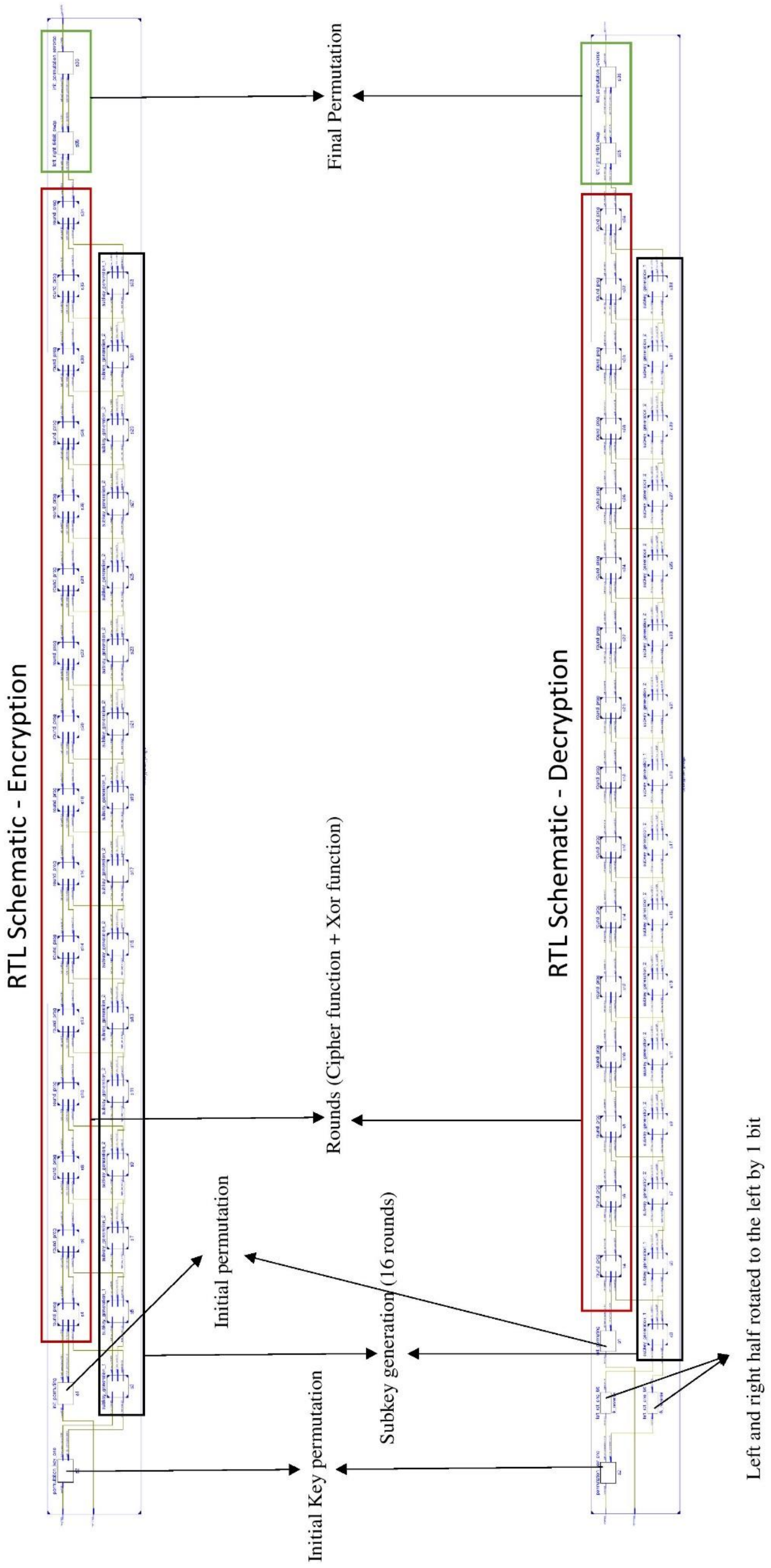
2.2 Decryption



2.2.1 RTL Schematic Decryption



2.3 Full view of the RTL schematic



III DESIGN and WORKING

3.1 Encryption

As described above, the DES operates on a 64 bit block of plain text. The plain text undergoes permutation, by which each bit of the plain text are arranged according to the positions allotted by the permutation table. The values in each matrix identify where each bit of the input message is mapped to in the output message. For example, The matrix for IP shows that the 58th bit from the input gets mapped to the first bit of the output; the 50th of the input maps to the second of the output, and so on. Then it is divided into two 32 bit blocks (i.e. left half and right half). The initial permutation table is as shown below.

IP

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

This is followed by 16 rounds of identical operation called the cipher function (discussed below), in which the data is combined/encrypted using the generated subkey.

3.1.2 Sub-key generation

Initially the key is 64 bits, out of which the every eight bit of the key is used for parity, thus removed before the key is used for encryption. Then the key is subjected to permutation based on the initial key permutation table, which is similar to the Initial permutation of plain text. The initial key permutation table is as given below.

Initial Key Permutation

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

Subkey Permutation

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Once the permutation is done, the 56 bit key is divided into two halves of 28 bit each called the left half and right half key. Next, both the halves are left rotated by one or two bits to the left as defined by the rotation table. Once the rotation is done both the halves are combined together to undergo a subkey permutation, where the 56 bit key is further permuted to generate a 48 bit subkey, to be used in the actual encryption process.

Fig: Subkey permutation table

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Fig: Key shifting factor

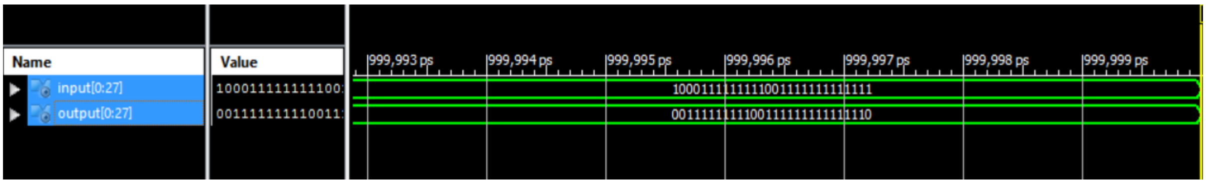


Fig: Simulation result of Left Rotation

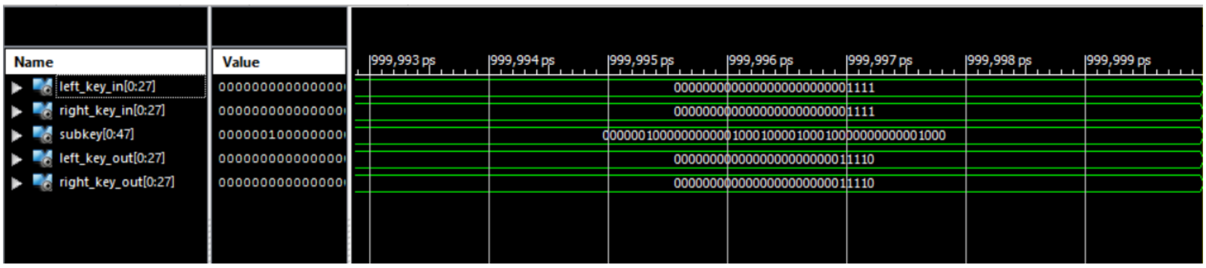


Fig: Simulation result of Subkey generation

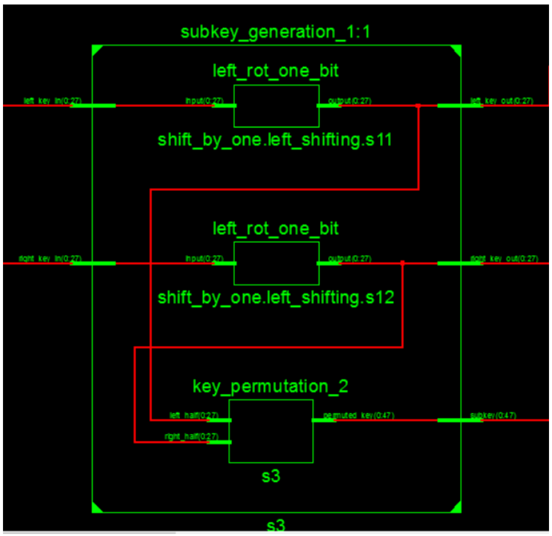


Fig : RTL schematic of Subkey generation

There are 16 such rounds, and hence 16 subkeys are generated.

3.1.3 Cipher function

The cipher function consists of the following four functions:

1. Expansion to 48 bit
2. XOR with subkey
3. Reduction to 32 bit
4. Cipher Permutation

3.1.3.1 The Expansion Permutation:

The right half of the data which is 32 bit long must now be converted into 48 bits, as the subkey that is generated is 48 bits long. This expansion is done by appending the last bit of the previous block (32nd bit) to the beginning of the block and by appending the first bit of the adjacent block (5th bit) to the end of the block.

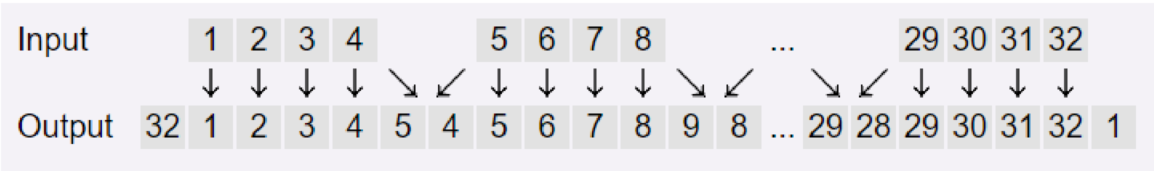


Fig: Cipher expansion

Consider an example:

11110000101010101111000010101010

1111 0000 1010 1010 1111 0000 1010 1010

011110 100001 010101 010101 011110 100001 010100 010101.

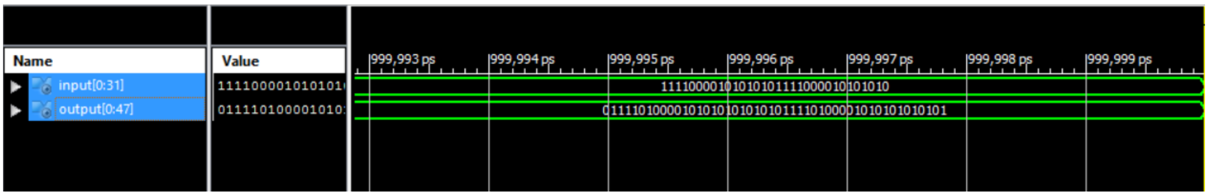
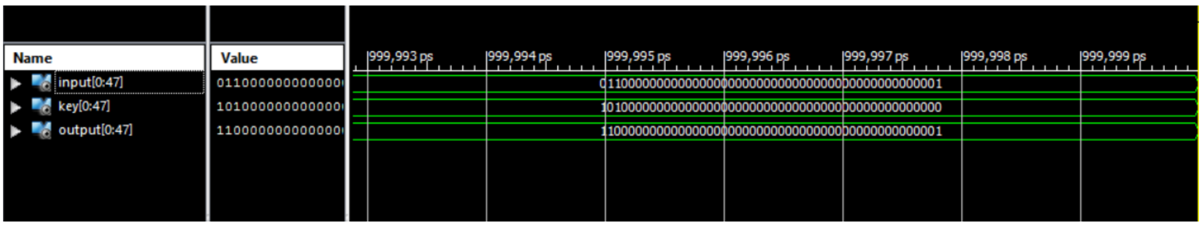


Fig: Expansion

3.1.3.2 48 bit XOR

The expanded 48 bit data from the plain text is now XORed with the 48 bit subkey generated. This data is then passed to the Sbox.



3.1.3.3 The Substitution Box

The obtained 48 bit sequence is broken down into eight groups of six bits each, each of which are passed through eight unique substitution boxes (abbreviated as S-box). This procedure removes the two bits padded to each block during expansion to produce a four bit data. This results in a 32 bit sequence which is passed for permutation.

Eg: Let us consider the above S – box 5.

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Suppose the 5th block is having data 011011

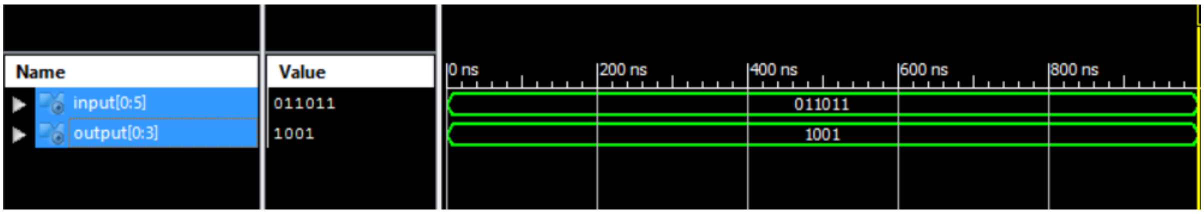
The outer bits are used to determine the row number and the four inner bits determine the column. The output of the S- box is then the value of that cell in the table.

Outer bits – Row - 01

Inner bits – Column – 1101

From the s box, matching the row and column, we get the data 1001

Simulation of the same is as shown below.



S-boxes

S_1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1yyyy1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0yyyy1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1yyyy0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1yyyy1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Fig: S-Box Substitution

3.1.3.4 Cipher Permutation

The 32 bit data received undergoes permutation according to the cipher permutation table as shown below.

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

Fig: Cipher permutation table

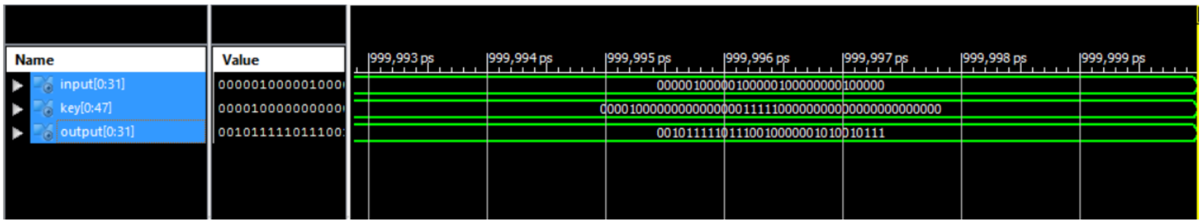


Fig: Simulation of Cipher function – combination of all 4 operations.

The output of this cipher function is then combined with the left half using another XOR. The overall result of the above operation gives us the new right half. The new left half is nothing but the old right half. This is repeated 16 times, making the 16 rounds of DES.

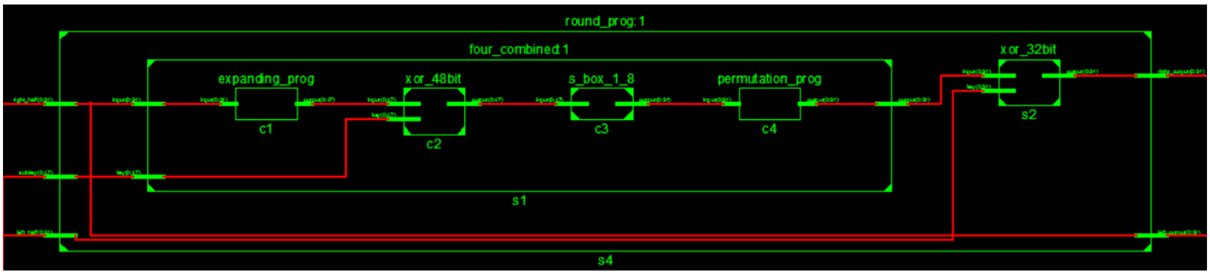


Fig : RTL Schematic of Cipher function and XOR forming one round

3.1.4 Final Permutation

This is the final step of the encryption process. It involves two main functions, Swapping of the left and right halves of encrypted plain text and applying inverse permutation table.

IP^{-1}

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

Fig: Inverse permutation table

3.2 Decryption

The same algorithm is used for decryption also. Functions like XOR and Rotation are selected such that DES has the same algorithm for both encryption and decryption with minor differences like for instance, all the binary rotations performed on decryption are right rotations and that the keys are used in the reverse order of that used for encryption.

IV EXPERIMENTAL SETUP

The design is implemented on Xilinx ISE using VHDL.

Design Properties:

- Source Type : HDL
- Family : Artix 7
- Device: XC7A100T
- Package : CSG324
- Synthesis tool : XST (VHDL/Verilog)
- Simulator : ISim (VHDL/Verilog)

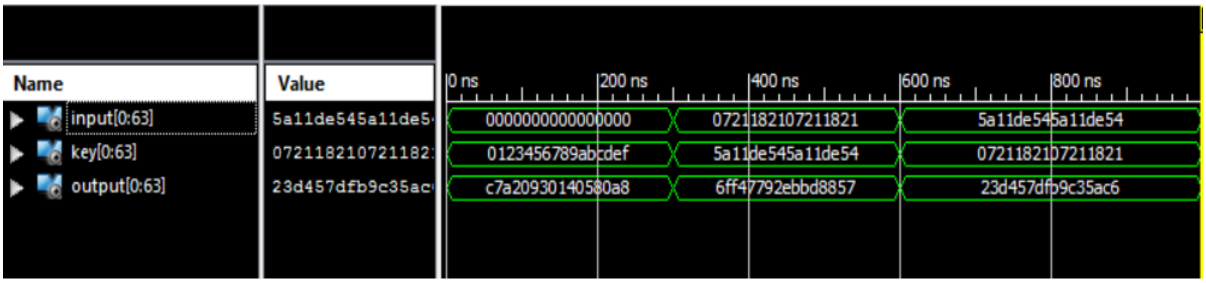
The functionality of the design is tested by writing test benches for each case with multiple instances.

V CONCLUSION

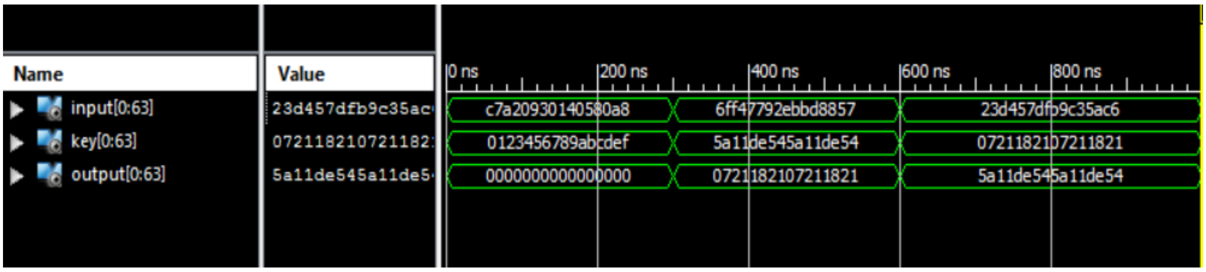
- DES has desirable properties such as avalanche effect and completeness.
- Trying 2^{56} combinations is not that easy.
- The 64 bit key does not provide enough security for most cases.
- Increasing key size would lead to a complex S-box mapping and the increase in the number of rounds would hamper differential analysis.
- DES is the building block for encryption standards such as 2-DES and 3-DES.
- AES is one of the successors to DES.

VI RESULTS

4.1 Encryption simulation



4.2 Decryption simulation



VII CONTRIBUTIONS

Sandesh

- *Cipher Function*
    - *Expansion program*
    - *S box*
    - *Cipher permutation*
- *Encryption*
    - *Initial permutation*
    - *Final permutation*
- *Left-right swap*

Guru

- *Key generation*
    - *Key permutation 1*
    - *Key permutation 2*
    - *Left, right rotate*
- *Decryption*
- *XOR function*

REFERENCES:

- Understanding Cryptography by Christof Paar and Jan Pelzl
- http://www.nku.edu/~christensen/DESschneier.pdf
- https://en.wikipedia.org/wiki/S
- https://en.wikipedia.org/wiki/DES_supplementary_material
- http://ccm.net/contents/134-introduction-to-encryption-with-des
- https://billstclair.com/grabbe/des.htm
- http://homepage.usask.ca/~dtr467/400/
- https://www.tutorialspoint.com/cryptography/data_encryption_standard.htm
- http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm
- https://en.wikipedia.org/wiki/Data_Encryption_Standard

QUESTIONS ASKED IN CLASS

What is Avalanche effect?

It refers to an attractive property of block ciphers and cryptographic hash functions algorithms. The avalanche effect is satisfied if: The output changes significantly (e.g., half the output bits flip) as a result of a slight change in input (e.g., flipping a single bit). A slight (a char or bit) change in the plaintext will drastically change the cipher text. each bit of ciphertext depends upon multiple bits of plaintext.

What is Completeness?

In cryptography, a boolean function is said to be complete if the value of each output bit depends on all input bits. Each bit of ciphertext will be changed if a single bit of plaintext is altered.