Design and Low Power VLSI Implementation of Triple -DES Algorithm

Alexandra Camacho, Isaac Sanchez, Eugene B. John and Ram Krishnan
Department of Electrical and Computer Engineering
The University of Texas at San Antonio
One UTSA Circle, San Antonio, TX 78249-0669

(oci482@my.utsa.edu; isanchez87@gmail.com; eugene.john@utsa.edu; ram.krishnan@utsa.edu)

***Abstract*— Triple DES (Data Encryption Standard) is a widely used encryption algorithm known to achieve good performance and high security. In this paper, we describe the design and low power VLSI implementation of the well-known triple DES algorithm. The implementation includes two main parts: key generation and the encryption/decryption process. In the DES module, the key generation part takes the given key and produces 16 distinct keys to be used during the encryption stage. The DES process is then repeated three times for added security. The chip was implemented using TSMC 180nm process. The designed chip has an area of 766,359 μm^2 and the power dissipation is 32.38mW for a Vdd of 1.8V.**

Keywords-Encryption, DES, 3DES, Low Power, Cryptosystem.

I. INTRODUCTION

With today's technology, there is a large amount of data constantly transferred electronically at any given time. It is typical that at some point in the day many people in the U.S. send emails, pay bills, or communicate wirelessly. Needless to say, much of this information is sensitive and the users have a desire for secure transactions. Ideally when data is transferred it is desired that the original information is transformed into apparent nonsense, so that in the event of third parties intercepting the data, it will appear illegible or as complete nonsense. Only the sender and receiver are able to decipher the data using special knowledge of the key used to transform the data. This transformation is called encryption and the algorithm used to map the data to its transformed state is known as a cipher [1]. There are many of such algorithms in existence however the proposed algorithm of interest is referred to as the Data Encryption Standard (DES). It was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 and which has subsequently enjoyed widespread use internationally [2]. Along with the growth of electronic data transmission was the development of the devices used to transmit the data. These include many processors used to drive data through landlines and wireless mediums. Many of these devices use encryption implemented on hardware for speed and use less computational memory of computers to run the algorithm [3]. As such, on this paper we describe the design and low power VLSI implementation of a DES algorithm for sending plain text information. In addition we seek to improve encryption by building on the DES chip and developing a Triple-DES algorithm that is more secure and commonly used by financial institutions for secure transactions.

II. BACKGROUND

A. *Data Encryption Standard*

The Data Encryption Standard (DES) is a block cipher that uses shared secret encryption. DES is the archetypal block cipher which is an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. The block size for DES is 64 bits and the cipher text produced appears as nonsense to third parties not intended to intercept the data. DES uses a customizable key for the transformation, so that decryption can supposedly only be performed by those who know the custom key used to encrypt. The key consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits, and it is never quoted as such. Every 8th bit of the selected key is discarded, that is, positions 8, 16, 24, 32, 40, 48, 56, 64 are removed from the 64 bit key leaving behind only the 56 bit key [2]. The availability of improved computational power eventually made the original DES algorithm less secure as it became more susceptible to brute-force attacks. To counter this, Triple DES was developed as a relatively simple method of increasing the key size of DES to protect against such attacks, without the need to design a completely new block cipher algorithm. Triple-DES has a number of advantages. First, it builds upon the concept of multiple encryption where a single block of plaintext is encrypted 3 times, typically using 2 different keys K1 and K2. Specifically, given a plaintext block, the DES algorithm is run in encryption mode using K1, then run in decryption mode using K2 and finally run in encryption mode using K1. Thus every plaintext block goes through encrypt (K1), decrypt (K2) and encrypt (K1) steps. Second, Triple-DES in encrypt(K1)-decrypt(K2)-encrypt(K1) is backward compatible with single DES since if K1=K2 in this mode, the first two steps cancel each other resulting in single encryption. This allows legacy applications developed to work with single DES to continue to inter-operate with Triple-DES implementations. Finally, Triple-DES when run in this mode with two keys still maintains an effective key size of 112 bits

[4][5]. This is because a meet-in-the-middle known plaintext attack on multiple encryptions still requires an attacker to brute-force through two steps involving two different keys.

B. Method and Design

The hardware implementation was derived from the algorithm described in [7] and [8]. DES uses a 64-bit key to encrypt 64-bit blocks of data through 16 rounds of permutations, xors, and table look-ups. The key is also shifted and permuted at each stage to increase security. DES is a symmetric algorithm, which means that ciphertext created using a particular key can be decrypted into plaintext using the same hardware and key. Figure 1 shows a data flow graph of how DES works.

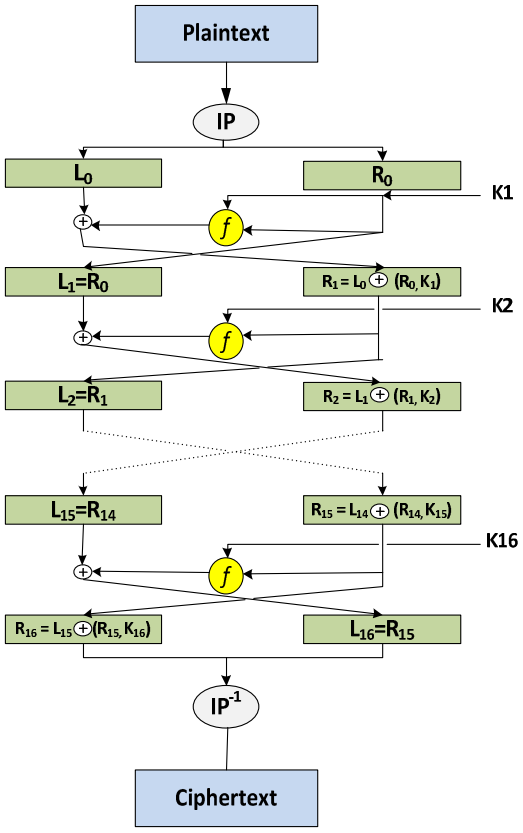
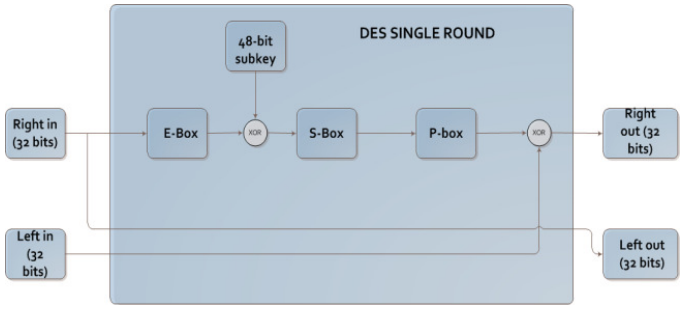


Figure 1: DES Data Flow Graph

During each of the 16 rounds , the data is separated into 32-bit halves. The right half is passed through to become the left half of the next round and is expanded through a permutation to 48-bits before being XORed with the key. The XORed value then becomes the addresses for the 8 sboxes. These sboxes are basically small Look up tables. The output of the sboxes is passed through another permutation before being xored with the left half. This criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes and the only difference is that the subkeys are applied in the reverse order when decrypting. One round of the Feistel Scheme is shown below in Figure 2:

Figure 2: Feistel Scheme



The first step is to pass the 64-bit key through a permutation called Permuted Choice 1, or PC-1 for short. The table for this is given in Figure 4. Note that in all subsequent descriptions of bit numbers, 1 is the left-most bit in the number, and n is the rightmost bit.

Bit	0	1	2	3	4	5	6
1	57	49	41	33	25	17	9
8	1	58	50	42	34	26	18
15	10	2	59	51	43	35	27
22	19	11	3	60	52	44	36
29	63	55	47	39	31	23	15
36	7	62	54	46	38	30	22
43	14	6	61	53	45	37	29
50	21	13	5	28	20	12	4

Figure 4: PC-1 Subkey Mapping

For example, we can use the PC-1 table to figure out how bit 30 of the original 64-bit key transforms to a bit in the new 56-bit key. Find the number 30 in the table, and notice that it belongs to the column labeled 5 and the row labeled 36. Add up the value of the row and column to find the new position of the bit within the key. For bit 30, 36 + 5 = 41, so bit 30 becomes bit 41 of the new 56-bit key. Note that bits 8, 16, 24, 32, 40, 48, 56 and 64 of the original key are not in the table. These are the unused parity bits that are discarded when the final 56-bit key is created. Now that we have the 56-bit key, the next step is to use this key to generate 16 48-bit subkeys, called K[1]-K[16], which are used in the 16 rounds of DES for encryption and decryption. These are generated by splitting the current 56-bit key, K, into two 28-bit blocks, L and R. Using the subkey rotation table in Figure 5 below, L and R are shifted by the number of bits indicated by the subkey index. L and R are then rejoined and permuted choice 2 (PC-2) is applied as PC-1 was before to get the final resulting subkey. Sixteen 48-bit subkeys are gereated for each round of the DES algorithm.

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

Bit	0	1	2	3	4	5
1	14	17	11	24	1	5
7	3	28	15	6	21	10
13	23	19	12	4	26	8
19	16	7	27	20	13	2
25	41	52	31	37	47	55
31	30	40	51	45	33	48
37	44	49	39	56	34	53
43	46	42	50	36	29	32

Figure 5: Rotation table and PC-2

As stated earlier the original 64-bit block of data is split into two 32-bit halves. Before that, the data undergoes an initial permutation similar to PC-1 and PC-2. The inverse of this permuation is performed at the end of the 16 rounds of DES algorithm. The initial permutation and its inverse are shown in Figure 6.

IP: Initial Permutation

Bit	0	1	2	3	4	5	6	7
1	58	50	42	34	26	18	10	2
9	60	52	44	36	28	20	12	4
17	62	54	46	38	30	22	14	6
25	64	56	48	40	32	24	16	8
33	57	49	41	33	25	17	9	1
41	59	51	43	35	27	19	11	3
49	61	53	45	37	29	21	13	5
57	63	55	47	39	31	23	15	7

IP ^(-1): Inverse Initial Permutation

Bit	0	1	2	3	4	5	6	7
1	40	8	48	16	56	24	64	32
9	39	7	47	15	55	23	63	31
17	38	6	46	14	54	22	62	30
25	37	5	45	13	53	21	61	29
33	36	4	44	12	52	20	60	28
41	35	3	43	11	51	19	59	27
49	34	2	42	10	50	18	58	26
57	33	1	41	9	49	17	57	25

Figure 6: Initial Permutation and its inverse.

Whenever the permuted data is split, the right half is expanded from 32 bits to 48 bits using the expansion table in Figure 7. The original right half is stored to become the left half for the next round in the DES algorithm later on. Notice that the expansion table includes repeat of bits.

E-Bit Selection Table

Bit	0	1	2	3	4	5
1	32	1	2	3	4	5
7	4	5	6	7	8	9
13	8	9	10	11	12	13
19	12	13	14	15	16	17
25	16	17	18	19	20	21
31	20	21	22	23	24	25
37	24	25	26	27	28	29
43	28	29	30	31	32	1

Figure 7: Expansion Table

The output of the expansion block is xored with the 48-bit subkey previously generated for that round. This result is split into eight 6-bit blocks. Each block is used as an address for its corresponding s-box look up table, therefore there are eight s-box tables. Below s-box 1 is shown. The first and last bit of the 6-bit blocks are used to index the row of the table. The middle 4 bits index the columns. The value at those indices are concatenated with remaining s-box results. The eight addresses produces 4 bit values each, so the resulting data is 32-bits. At this point another permuation is performed using the p-box table in Figure 8. This result is xored with the original left half during the round and the result becomes the right half for the next round and as stated earlier the next rounds left half is the current rounds initial right half.

S-BOX 1: Substitution Box 1

Row/Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

P-PERMUTATION TABLE

Bit	0	1	2	3
1	16	7	20	21
5	29	12	28	17
9	1	15	23	26
13	5	18	31	10
17	2	8	24	14
21	32	27	3	9
25	19	13	30	6
29	22	11	4	25

Figure 8: S-Box and P-Box

The previous steps are repeated for the duration of the 16 rounds of the DES algorithm. The ciphertext is produced when the inverse initial permutation block is applied. We are easily able to implement a triple DES by applying DES 3 times to 64 bit plaintext data with 3 different keys and sets of permutations each time the DES is applied.

III. Experimental Methods and Results

In this section we describe the experimental methods and the design results. Figure 9 depicts the synthesized top level DES module. The inputs include the 64-bit key and plain text data, a clock, enable, encrypt/decrypt and reset. By toggling the encrypt/decrypt input from 0 to 1, we can set the DES to encrypt the plain text or decrypt the cipher text. Decrypting works exactly in the same manner; however, the subkeys are applied in reverse order. The image on the right shows the next level schematic which is a more detailed view of all necessary connections and modules instantiated for the full 16 rounds. The rounds and constituting sub modules are explained in more detail in the following paragraphs.

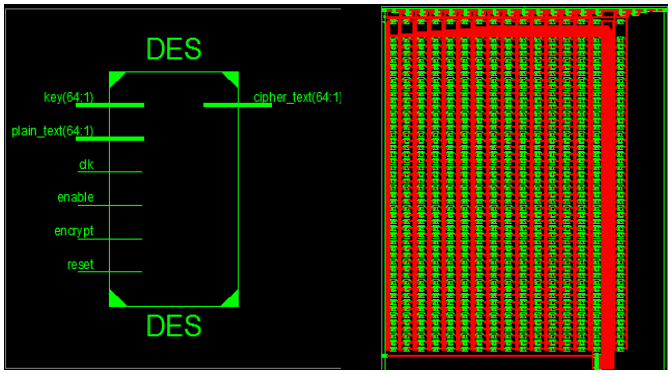


Figure 9: DES Module (Left): Top Module (Right): Next Level

Figure 10 shows the schematic of 1 of the 16 rounds in the DES algorithm. Notice the 8 S-boxes. The original algorithm flow chart is also shown to show the translation.

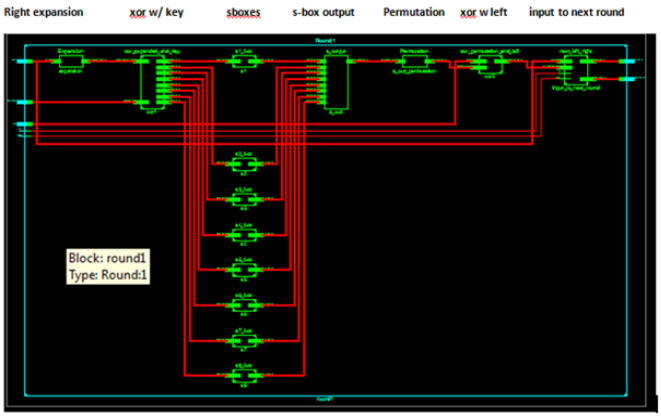


Figure 10: Single Round of DES Algorithm

Figure 11 shows the top level schematic of the Triple DES. It works exactly the same, but requires 3 keys to be input into the top module.

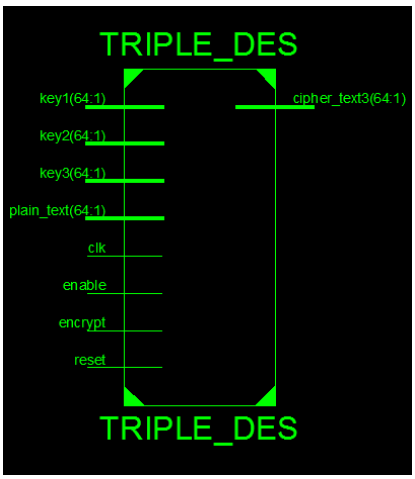


Figure 11: Triple DES Top Module

Behavioral Test:

In order to test the functionality of our design we used the data shown in Figure 12. The top portion of the figure shows 2 blocks of plain text data to test the DES algorithm. Together the blocks form the string "Now is the time". As shown, the string is broken up into 8 byte blocks, since the DES algorithm is designed for 64bit data. Each character is represented in Hex as well. The bottom portion of Figure 12 shows an example of credit information that is generally encrypted using the triple DES algorithm. We sought to successfully encrypt and decrypt this data.

First block:

```
DES INPUT BLOCK  = N  o  w  _  i  s  _  t
(IN HEX)         = 4E 6F 77 20 69 73 20 74
KEY              = 01 23 45 67 89 AB CD EF
DES OUTPUT BLOCK = 3F A4 0E 8A 98 4D 43 15
```

Second block:

```
DES INPUT BLOCK  = h  e  _  t  i  m  e  _
(IN HEX)         = 68 65 20 74 69 6D 65 20
KEY              = 01 23 45 67 89 AB CD EF
DES OUTPUT BLOCK = 6A 27 17 87 AB 88 83 F9
```

Encrypted Message

Field	Length	Example
Customer Reference Number	5 Digits	12345
Credit Card Number	15 Digits	376066666655555
Expiration Date	4 Digits	1205
Total	24 Digits	

Figure 12: Behavioral Data

The correct operation of this circuit was verified by running simulations for both the single DES and Triple-DES designs. The encryption input was switched between 1 and 0 to encrypt/decrypt the information seen in Figure 12.

Final Layout

Figure 13 shows the final layout of the Triple DES chip that was synthesized using Cadence Encounter and RTL compiler for TSMC 180nm process. Through the power and timing analysis we were able to distinguish the specs for the Triple DES and Single DES by analyzing the single DES modules.

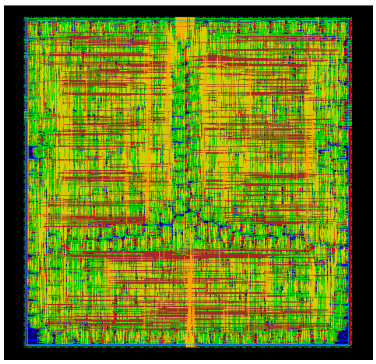


Figure 13: GDS2 Layout of Triple DES Chip

The specs for the designed Triple-DES chip are as follows: (1) Designed using TSMC 180 nm process (2) Required VDD: 1.8V (3) Area: 766,359 μm^2 (4)Total number of pins on chip: 326 (Three 64bit Keys, One 64bit plaintext input, One 64bit ciphertext output and 6 additional pins for clk, enable, encrypt, vdd, and gnd) (5) Leakage power dissipation of the chip is 20.72 µW and the switching power dissipation is 32.36 mW. The total power consumption of the Triple-DES is 32.38 mW. A summary of the specifications is given in Table 1.

Chip Specifications	
Technology	TSMC 180 nm
VDD	1.8 V
Area	766,359 µm2
Number of pins	326
Leakage Power	20.72 µW
Switching Power	32.36 mW

Table 1: Triple-DES Chip Specifications

IV. CONCLUSION

In this paper we present the design and low power VLSI implementation of Single DES and Triple-DES algorithm. Since the original DES algorithm was less secure as it became more susceptible to brute-force attacks, triple DES was developed as a relatively simple alternative without the need to design a completely new block cipher algorithm. In our design we used 3 instances of our Single DES algorithm to improve the encryption strength. It should be noted that the Triple DES implementation is considerably slower, but more secure. The designed chip can be used for financial data transfer such as credit card information where a high level of security is needed. The chip was designed using TSMC 180nm process for 1.8 V Vdd. The area of the designed chip is 766,359 μm^2, the number of pins on the chip is 326, and the power dissipation is 32.38mW.

REFERENCES

[1] Bellare, Mihir; Rogaway, Phillip (21 September 2005). "Introduction". Introduction to Modern Cryptography. p. 10

[2] NBS FIPS PUB, ªData Encryption Standard,ºNat'l Bureau of Standards, US Dept. of Commerce, Jan. 1977.

[3] http://www.via.com.tw/en/initiatives/padlock/whyhardwareisbetter.jsp

[4] Ralph Merkle, Martin Hellman: On the Security of Multiple Encryption , Communications of the ACM, Vol 24, No 7, pp 465–467, July 1981.

[5] Paul van Oorschot, Michael J. Wiener, A known-plaintext attack on two-key triple encryption , EUROCRYPT'90, LNCS 473, 1990, pp 318–325.

[6] NIST Special Publication 800-57 Recommendation for Key Management Part 1: General (Revised), March, 2007

[7] www.jhdl.org/documentation/latestdocs/code/JHDL_examples1.html

[8] http://www.tropsoft.com/strongenc/des.html