A Designed DES Chip with CPLD Technology

Narong Buabthong and Somchart Chokchaitam
Lecturers, Dept.of Electrical Engineering, Faculty of Engineering, Thammasat University

Surachai Hirinitichai and Kanok Musickmas
Students, Dept.of Electrical Engineering, Faculty of Engineering, Thammasat University

Abstract

This paper presents a designed DES chip with CPLD technology. The DES chip is synthesized and simulated by using AHDL technology. Three MAX9000 chips are used for DES algorithm by using 1388 logic cells, 86% utilization, 123-input pins, and 64-output pins.

1. Introduction

Due to the growth of the communication network in the past decade, information technology has become popular. Communication between anywhere in the world can be done within a few seconds through the information superhighway. Although the information superhighway has many advantages, the security of the data passing through is the main problem. Cryptography technique is designed to solve the problem by scrambling data before passing it through the network. Plaintext (data) is changed into cryphertext (scrambled data) by using an encryption technique, and then cryphertext is sent through the network. The cryphertext is converted into plaintext at the end-user by using the decryption technique. Only the right end-user knows how to decrypt the cryphertext, so only the right end-user can convert the plaintext back. On the other hand, if the wrong end-users get the cryphertext, scrambled data is useless. The selected Data Encryption Standard (DES) for designing the hardware to encrypt to DES is based on US National Bureau of Standards. This designed hardware emphasizes high speed operation for the future communication network.

2. DES Algorithm

Data Encryption Standard (DES) [1] is one of the most famous cryptosystems. The DES algorithm is based on the Lucifer block cipher designed by Horst Feistel and was developed at IBM in 1972. This algorithm was approved by the National Bureau of Standard (NBS) and the National Security Agency (NSA), then it was adopted as a Federal Standard in 1977.

The DES, a block cipher, an algorithm with a 64-bit block of plaintext, is enciphered into a 64-bit ciphertext by using a 56-bit secret key. The DES algorithm is shown in fig. 1, it has many steps as following. In the first step, a 64-bit block of plaintext is divided into 2 groups of data; 32 left-data and 32 right-data at the Initial Permutation box (IP box). The 32 left-data and the 32 right-data have to pass 16 rounds of the enciphering process namely, "single loop DES". Only the 32 right-data is scrambled by the single loop DES but the 32 left-data is kept. In the single loop DES, the 32 right-data is expanded from 32 bits to 48 bits by the Expansion Box (E-Box), and then the 48 bits are obtained from the E-box and may be given a 48 loop-key. The 48 bit-data are reduced into 32 bit again by the Substitution Box (S-Box) and the results are combined with the 32 kept left-data. The final result is 32 right-data for the next round and the old 32 right-data is 32 left-data for the next round. After the data pass all 16 rounds, they have to pass an Inverse Initial Permutation (IP^{-1} Box) to convert them to the ciphertext (output).

3. The designed issue

The DES Chip, designed using AHDL language [2,3,4] as shown in fig. 2 [a] and fig. 2 [b], consists of the following:

1 Substitutions
2 Single Loop DES
3 Latch and Data Selector
4 IP Box
5 IP^{-1} Box
6 Key Schedule for Encryption and Decryption

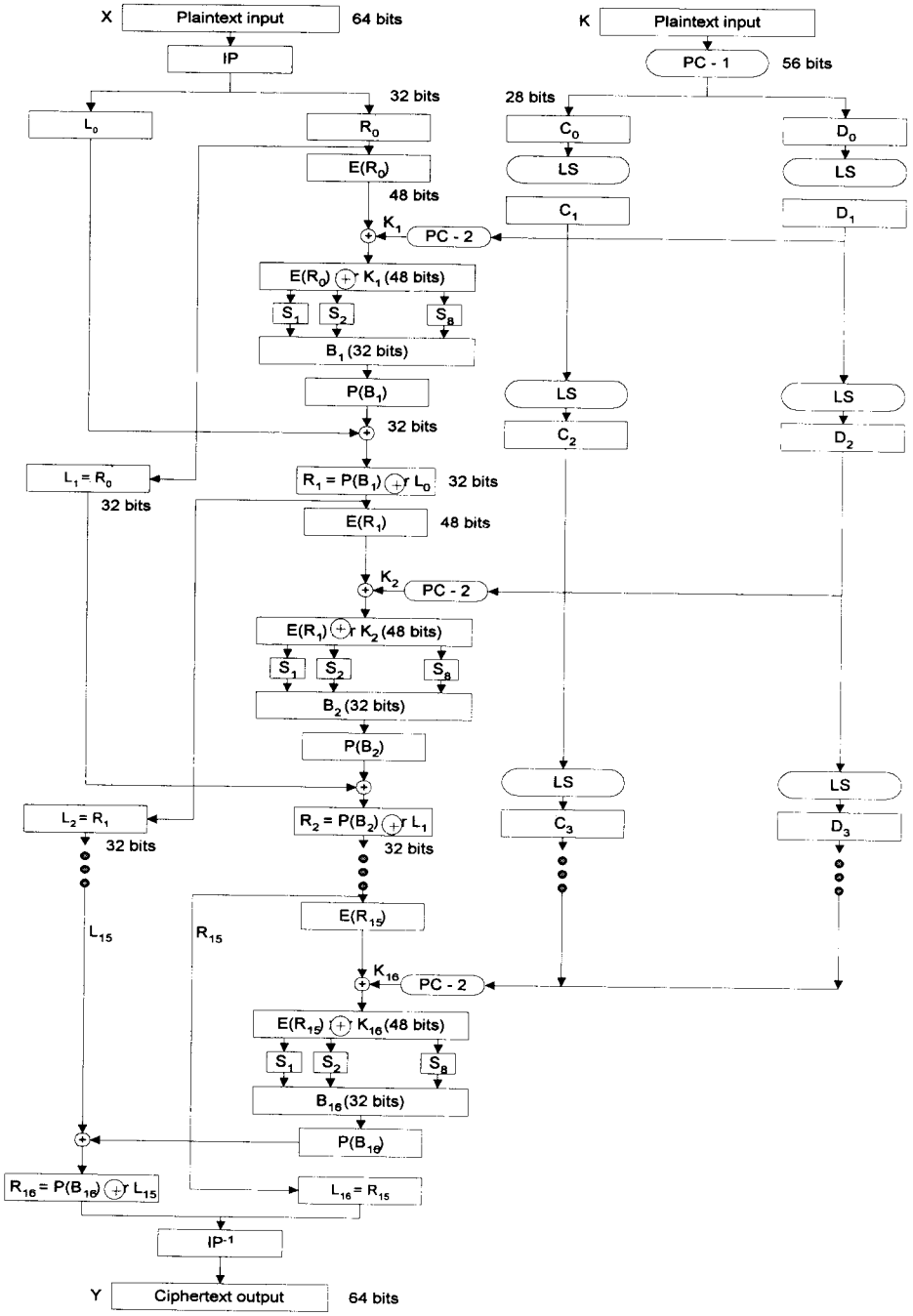

Figure 1: DES Algorithm

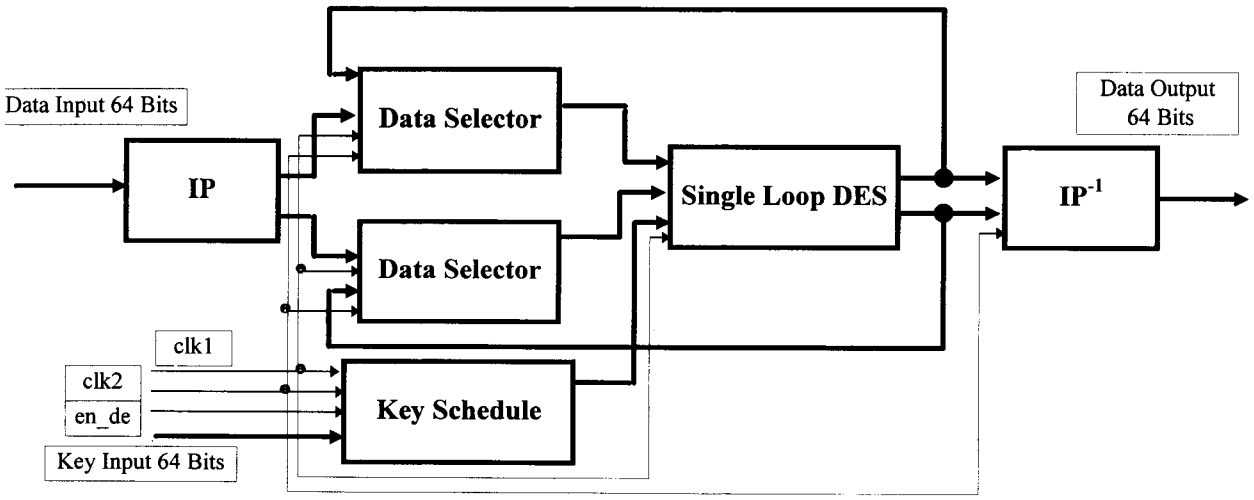

Figure 2 [a]: The DES Chip

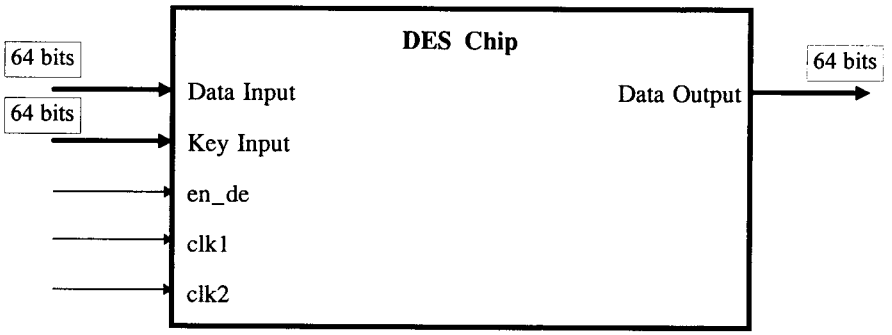

Figure 2 [b]: The design of DES Chip

3.1 The Substitution Box

The substitution box transforms the 48 bit-input data to 32 bit-output data, it is divided into eight S-boxs. Each S-box has a different table to convert 6 bit-data to 4 bit-data. The design and synthesis of S-Box uses AHDL language [2,3,4] as in the CASE statement in example 1. The synthesized result and its designed diagram is shown in fig. 3, they will be used as a function for single loop DES.

Example 1

```
SUBDESIGN table5
(       data_in[1..48] : INPUT;
        data_out[1..32] : OUTPUT;       )
VARIABLE
        i1m[1..6],i2m[1..6],i3m[1..6],i4m[1..6],i5m[1..6],i6m[1..6],i7m[1..6],i8m[1..6] : NODE;
BEGIN
        i1m[] = data_in[1..6];
        i2m[] = data_in[7..12];
        i3m[] = data_in[13..18];
        i4m[] = data_in[19..24];
        i5m[] = data_in[25..30];
        i6m[] = data_in[31..36];
        i7m[] = data_in[37..42];
        i8m[] = data_in[43..48];
        CASE i1m[] IS
                WHEN B"000000" =>
                        data_out[1..4] = B"1110";
                WHEN B"000010" =>
                        data_out[1..4] = B"0100";
                WHEN B"000100" =>
                        data_out[1..4] = B"1101";
                WHEN B"000110" =>
                        data_out[1..4] = B"0001";
```

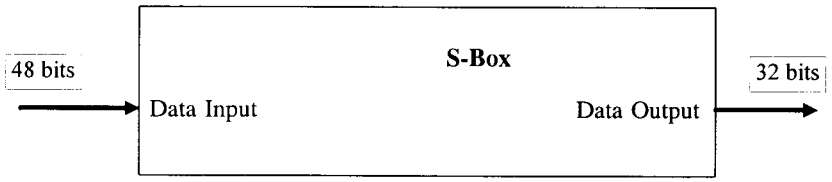

Figure 3: The disign of S-Box

3.2 Single Loop DES

The main function of a single loop DES is to scramble data with a 56 key-input. Its designed diagram as shown in fig. 4, consists of 4-main parts; E-box, P-box, S-box and PC-2. The input of a single loop DES is 32 left-data, 32 right-data, 56 key-input and CLK. The CLK is a communication clock for contrlling the operation period of the chip. The chip is operated when CLK is "0". The output of the single loop is 32 left-data and 32 right-data; the input for the next loop. The single loop DES is synthesized by using AHDL language [2,3,4] as an example 2.

Example 2

```
FUNCTION table5 (data_in[1..48])
  RETURNS (data_out[1..32]);
SUBDESIGN macro5
(       clk : INPUT;
        data_in_r[1..32] : INPUT;
        data_in_l[1..32] : INPUT;
        key_in[1..56] : INPUT;
        data_out_r[1..32] : OUTPUT;
        data_out_l[1..32] : OUTPUT;    )
VARIABLE
        imm[1..32] : node;
        imr[1..32],iml[1..32] : node;
        key1_im[1..56] : node;
        e_im[1..48],pc2_im[1..48],l_im[1..48] : node;
        s1im[1..32],p_im[1..32] : node;
BEGIN
        imr[] = data_in_r[]; iml[] = data_in_l[]; key1_im[] = key_in[];
        IF clk == GND THEN
        e_im[] = (imr32,imr1,imr2,imr3,imr4,imr5,imr4,imr5,imr6,imr7,imr8,imr9,imr8,imr9
                ,imr10,imr11,imr12,imr13,imr12,imr13,imr14,imr15,imr16,imr17,imr16,imr17
                ,imr18,imr19,imr20,imr21,imr20,imr21,imr22,imr23,imr24,imr25,imr24,imr25
                ,imr26,imr27,imr28,imr29,imr28,imr29,imr30,imr31,imr32,imr1);
        pc2_im[] = (key1_im14,key1_im17,key1_im11,key1_im24,key1_im1,key1_im5,
                    key1_im3,key1_im28,key1_im15,key1_im6,key1_im21,key1_im10,
                    key1_im23,key1_im19,key1_im12,key1_im4,key1_im26,key1_im8,
                    key1_im16,key1_im7,key1_im27,key1_im20,key1_im13,key1_im2,
          key1_im41,key1_im52,key1_im31,key1_im37,key1_im47,key1_im55,
                    key1_im30,key1_im40,key1_im51,key1_im45,key1_im33,key1_im48,
                    key1_im44,key1_im49,key1_im39,key1_im56,key1_im34,key1_im53,
                    key1_im46,key1_im42,key1_im50,key1_im36,key1_im29,key1_im32);
        l_im[] = e_im[] $ pc2_im[];
        s1im[] = table5(l_im[]);
        p_im[]=(s1im16,s1im7,s1im20,s1im21,s1im29,s1im12,s1im28,s1im17,s1im1,s1im15,
                s1im23,s1im26,s1im5,  s1im18,s1im31,s1im10,s1im2,s1im8,s1im24,s1im14,
                s1im32,s1im27,s1im3,s1im9,s1im19,s1im13,s1im30,s1im6,s1im22,s1im11,
                s1im4,s1im25);
        imm[] = p_im[] $ iml[];
        data_out_r[] = imm[];
        data_out_l[] = imr[];
        END IF;
```

END;

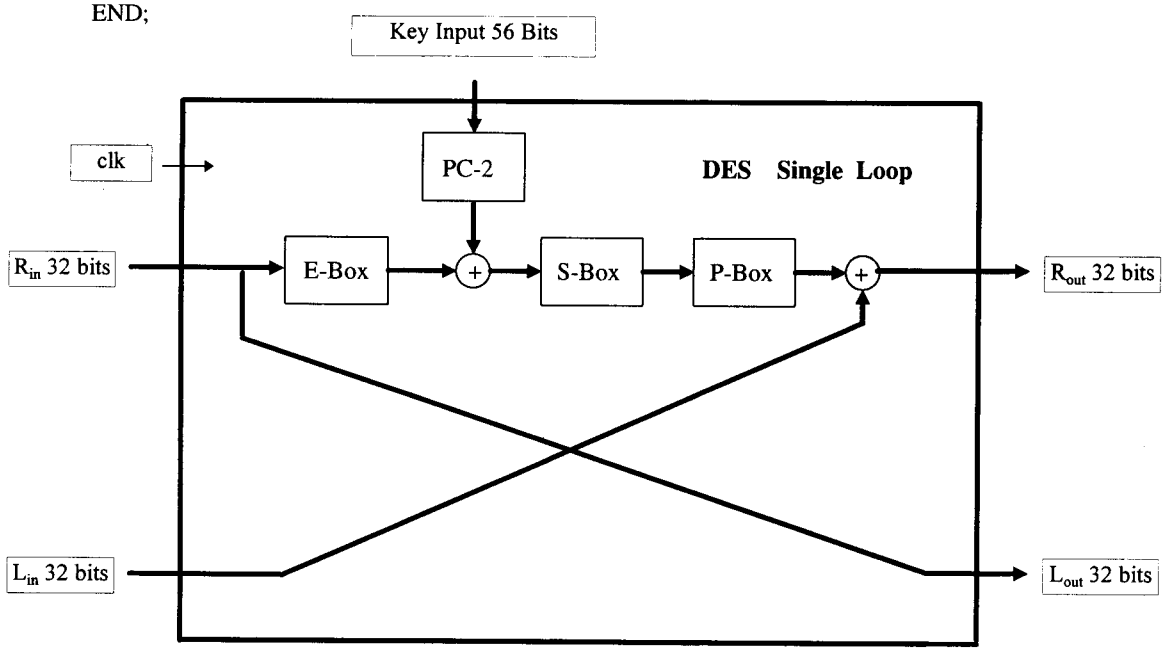

Figure 4: The Single Loop DES

3.3 Latch Data Selector

The main function of the latch data selector is to select data between user data and single loop DES output data for looping them 16 rounds as DES algorithm. There are 4 inputs; data input from user, data input from output of the single loop, CLK 1 and CLK 2. The CLK 1 is a clock to control the operation period of the chip and the CLK 2 is a clock to control data input and latch data by using D-flip-flop. The latch data selector is synthesized, by using AHDL [2,3,4] as in example 3. Its design diagram is shown in fig. 5

Example 3

```
SUBDESIGN tri32
(       clk1 : INPUT;
        clk2 : INPUT;
        data_1in[1..32] : INPUT;
        data_2in[1..32] : INPUT;
        data_out[1..32] : OUTPUT;       )
VARIABLE
        im[1..32] : DFF;
BEGIN
        IF clk1 == VCC THEN im[].d = data_1in[];
        ELSE im[].d = data_2in[];
        END IF;
        im[].clk = clk2;
        data_out[] = im[].q;
```

END;

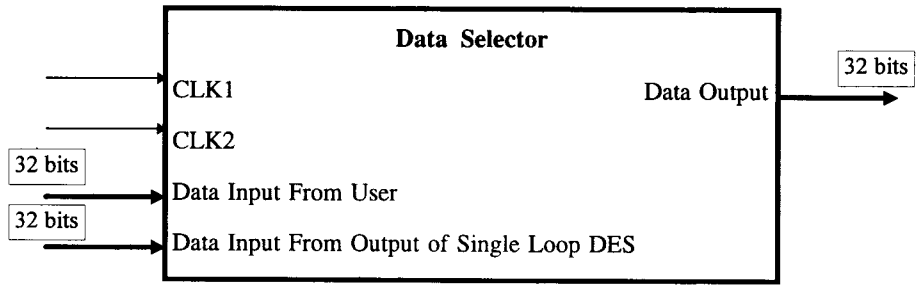

Figure 5: The Latch and Data Selector

3.4 IP Box

The main function of IP box is to divide 64 bit-data input to 32 left-data and 32 right-data. The IP box is synthesized by using AHDL langugue [2,3,4] as in example 4. Its designed diagram is shown in fig. 6.

Example 4

```
SUBDESIGN ip_box
(       data_in[1..64] : INPUT;
        data_rout[1..32] : OUTPUT;
        data_lout[1..32] : OUTPUT;      )
VARIABLE
        in[1..64] : node;
BEGIN
        in[] = data_in[];
        data_lout[] = (in58,in50,in42,in34,in26,in18,in10,in2,in60,
                       in52,in44,in36,in28,in20,in12,in4,in62,in54,
                       in46,in38,in30,in22,in14,in6,in64,in56,in48,
                      in40,in32,in24,in16,in8);
        data_rout[] = (in57,in49,in41,in33,
                       in25,in17,in9,in1,in59,in51,in43,in35,in27,
                       in19,in11,in3,in61,in53,in45,in37,in29,in21,
                       in13,in5,in63,in55,in47,in39,in31,in23,in15,
                       in7);
END;
```

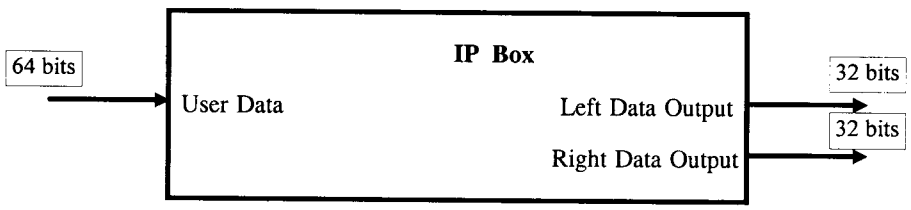

Figure 6: The IP Box

3.5 IP^{-1} Box

The main function of the IP^{-1} box or Inverse Initial Permutation is to convert 32 left-data and 32 right-data to the output of DES chip by using CLK to control the operation period. The IP^{-1} box is synthesized by using AHDL [2,3,4] as in example 5. Its design diagram is shown in fig. 7

Example 5

```
SUBDESIGN inv_ip
(       clk : INPUT;
        data_rin[1..32] : INPUT;
        data_lin[1..32] : INPUT;
        data_out[1..64] : OUTPUT;       )
VARIABLE
        imo[1..64] : node;
        im[1..64] : DFF;
BEGIN
        imo[] = (data_rin[],data_lin[]);
        im[].d=(imo40,imo8,imo48,imo16,imo56,imo24,imo64,imo32,imo39,imo7,imo47,imo15,
            imo55,imo23,imo63,imo31,imo38,imo6,imo46,imo14,imo54,imo22,imo62,imo30,
            imo37,imo5,imo45,imo13,imo53,imo21,imo61,imo29,imo36,imo4,imo44,imo12,
            imo52,imo20,imo60,imo28,imo35,imo3,imo43,imo11,imo51,imo19,imo59,imo27,
            imo34,imo2,imo42,imo10,imo50,imo18,imo58,imo26,imo33,imo1,imo41,imo9,
            imo49,imo17,imo57,imo25);
        im[].clk = clk;
        data_out[] = im[].q;
END;
```

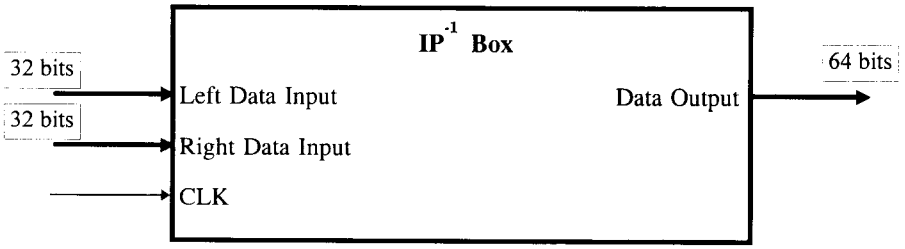

Figure 7: The IP^{-1} Box

3.6 Key Schedule

The main function of the key schedule is to create key in each loop. We select encryption operation or decryption operation by using en_de pin. The CLK 1 is a clock to control input data and the CLK 2 is a clock to control DES loop. The key schedule is synthesized as in the diagram shown in fig. 8.

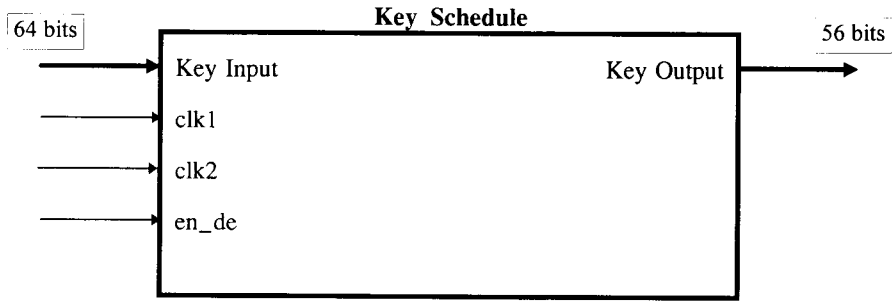

Figure 8: The Key Schedule

4. Simulation Results

The simulation of the DES chip has been done by using MAX+plus II program [2, 3], the synthesized results are shown in the report #1.

From the synthesized results three MAX9000 chips are used to design DES chip by using 1388 logic cells and 86% utilization. There are 123-input pins and 64-output pins for DES chip as shown in fig. 10 and fig. 11. The test results of the DES chip as shown in figs. 12-15 by forcing "0x785AC3A4BD0FE12D" as the input-data, "0x38A84FF898B90B8F" as the input-key, and setting "1" to en_de pin. After the data passes all the processes, the "0xFD9CBA5D26331F38" operates as the output-data (ciphertext). To test DES chip again by forcing "0xFD9CBA5D26331F38" as the input-data, "0x38A84FF898B90B8F" as the input-key, and setting "0" to en_de. The "0x785AC3A4BD0FE12D" operates as the output-data back.

From the simulations, the DES chip is able to operate at 4,480 kbps by using 1.25 MHz clock speed. To conclude, the DES chip operation speed is fast enough for applying to speech application. For the other higher speed application, the chip speed can be increased by increasing the clock speed, according to the DES chip delay time from our simulation

Project Information c:\winappl\max2work\present\macro15.rpt

MAX+plus II Compiler Report File
Version 5.4 06/19/95
Compiled: 03/21/96 15:02:46

***** Project compilation was successful

** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	LCs	Shareable Expanders	% Utilized
macro15	EPM9560GC280	130	66	0	537	306	95 %
macro151	EPM9480RC208	72	56	0	471	209	98 %
macro152	EPM9560RC240	98	72	0	380	317	67 %
TOTAL:		300	194	0	1388	832	86 %
User Pins:		123	64	0			

Report #1 the synthesized result

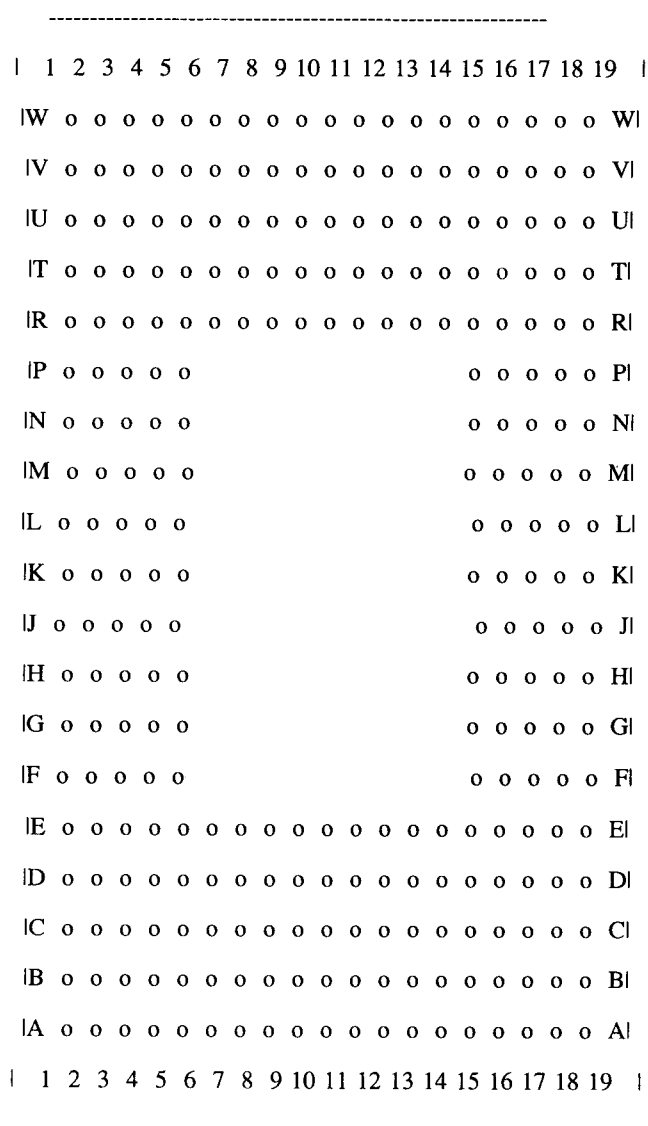

Figure 10 The Bottom View of EPM9560GC280 chip

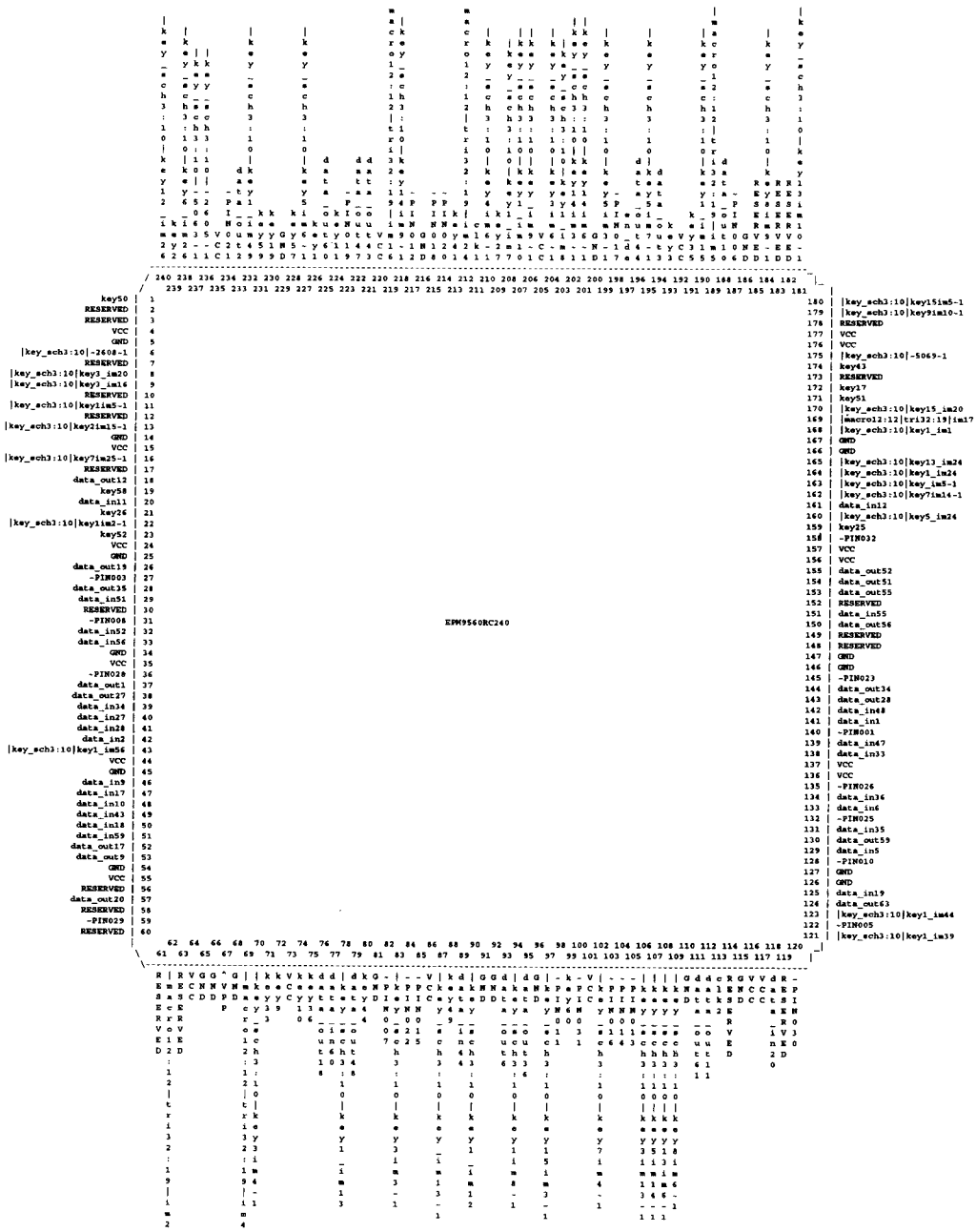

Figure 11 Pin assignment of EPM9560RC240 chip

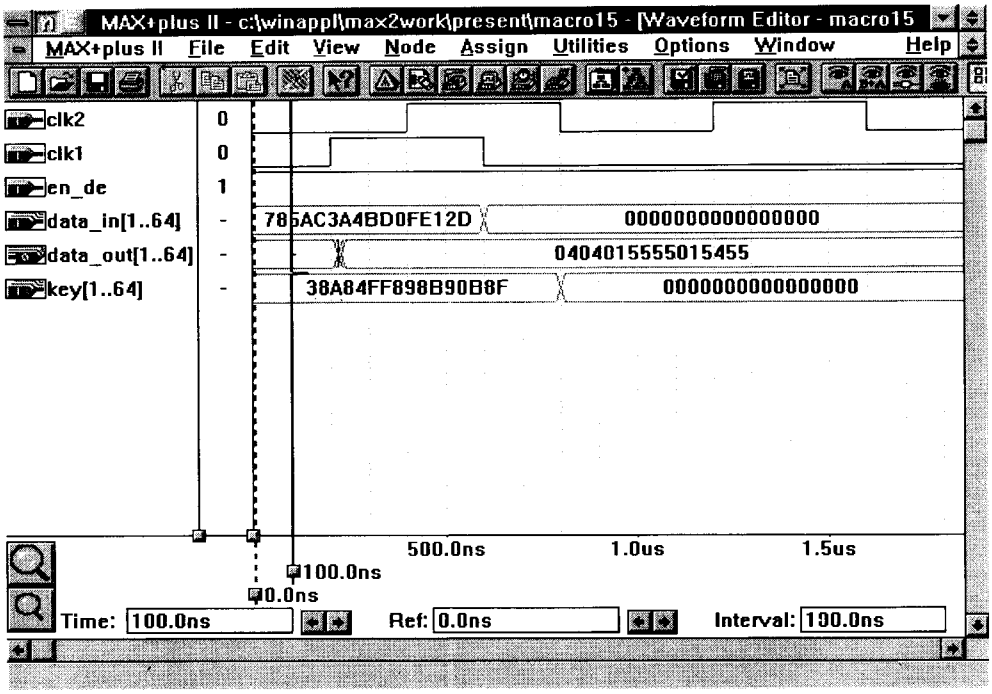

Figure 12 Simulated Waveforms of Encryption

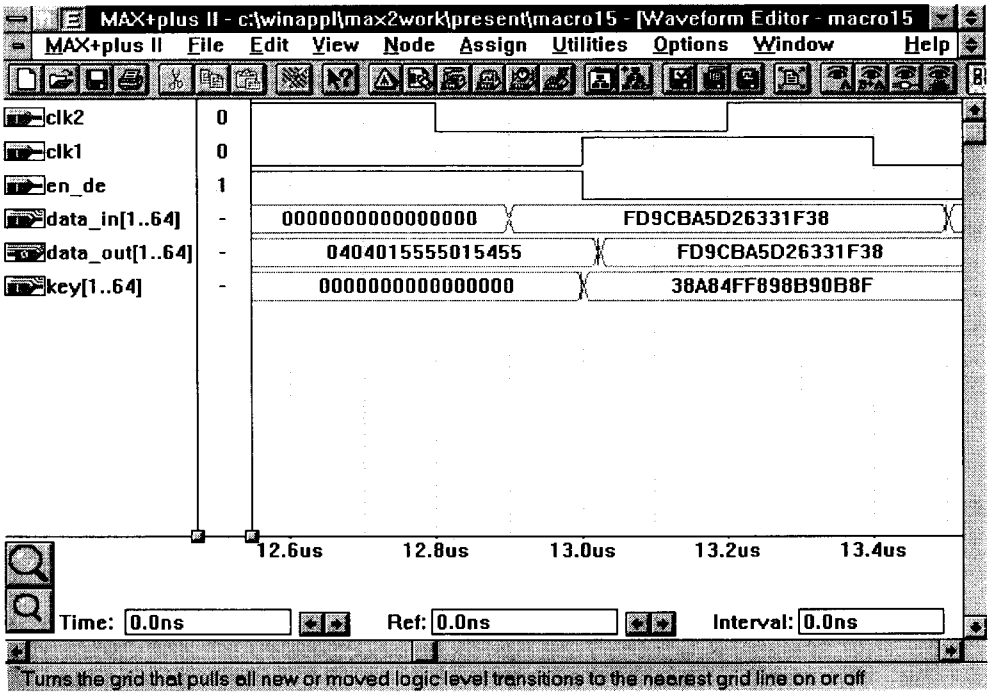

Figure 13 The results after encryption

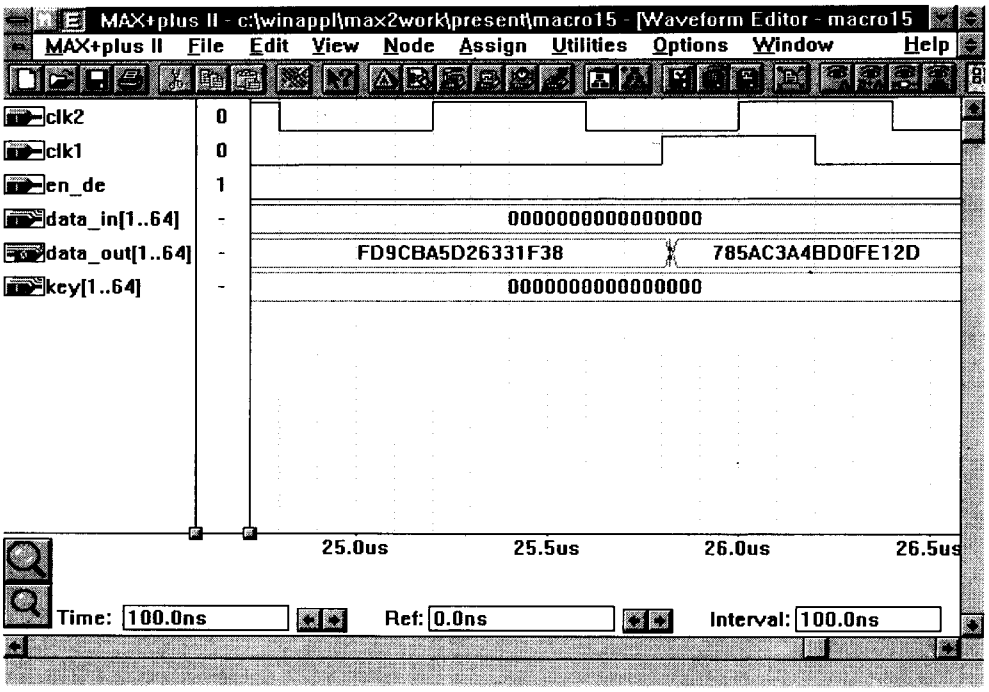

Figure 14 The results after Decryption

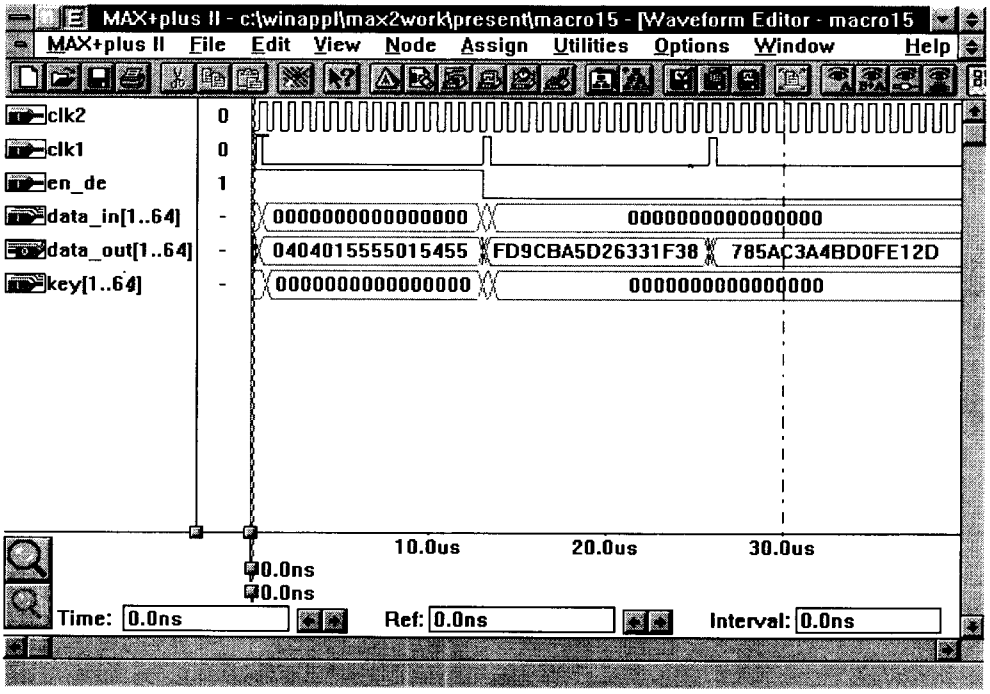

Figure 15 The Simulation Waveforms of DES Chip

5. Reference

[1] Rhee, M.Y. (1994), Cryptography and Secure Communications, McGraw Hill Book, New York.

[2] Altera Corporation. (1994), MAX+PLUS ® II, Getting Started , Altera Corporation, New York.

[3] Altera Corporation. (1994), MAX+PLUS ® II AHDL , Altera Corporation, New York.

[4] Atlera Corporation. (1995), Altera Data Book, Altera Corporation, New York.