DIGITAL DESIGN

ASSIGNMENTREPORT

ASSIGNMENT ID : 02

Student Name: Weibao Fu

Student ID: 11812202

PART 1: DIGITAL DESIGN THEORY

1. (a) Hexadecimal: $(1.A8)_{16}$ Decimal: 1.65625

   (b) Hexadecimal: $(D.4)_{16}$ Decimal: 13.25

   If we move the decimal point of the binary number in (a) three places to the right, the answer is same as the binary number in (b), such that the decimal answer in (b) is 8 times as much as the decimal answer in (a) .

2. (a) 37/2=18……1 18/2=9……0 9/2=4……1

   4/2=2……0 2/2=1……0 1/2=0……1

   0.875*2=1.75 0.75*2=1.5 0.5*2=1

   Such that the decimal 37.875 is $(100101.111)_2$

   (b) 1/7*2=2/7 2/7*2=4/7 4/7*2=8/7

   1/7*2=2/7 2/7*2=4/7 4/7*2=8/7

   1/7*5=2/7 2/7*2=4/7

   $1*2^{-3}+1*2^{-6}=0.140625$

   We can find that 1/7 is equal to $2^{-3} +2^{-6} +2^{-9} +…+2^{-3n} = \sum 2^{-3n}$ (n=1,2…)

   Such that $1/7 - 2^{-3}-2^{-6} = \sum 2^{-3n}$ (n=3,4…) = 2^{-6} /7 = 1/448

   0.140625 / (1/7) = 98.44%

   Such that the binary equivalent of 1/7 out to eight places is $(0.00100100)_2$, corresponding to the decimal number 0.140625. The difference between 1/7 and result is 1/448. And the result is about 98.44% of 1/7.

   (c) $(0.0010\ 0100)_2$ = $(0.24)_{16}$

$(0.24)_{16}$ = $2*16^{-1}$ +$4*16^{-2}$=0.140625

Such that the answer is the same.

3. BCD: 0111 1000 0001 0100

Excess-3 code: 1010 1011 0100 0111

2421 code: 1101 1110 0001 0100

6311 code: 1001 1011 0001 0101

4. (a) AND: 00010100

(b) OR: 10111101

(c) XOR: 10101001

(d) NOT A: 01001010

(e) NOT B: 11100011

(f) NAND: 11101011

(g) NOR: 01000010

5. (a) (a+b+c')(a'b'+c)

= aa'b+ac+a'b'b+bc+a'b'c'+c'c

= ac+bc+a'b'c'

(b) a'b'c+ab'c+abc+a'bc

= b'c(a'+a)+bc(a+a')

= b’c+bc

= c(b’+b)

= c

(c) (a+c)(a’+b+c)(a’+b’+c)

= (aa’+ab+ac+a’c+bc+c)(a’+b’+c)

= (ab+c)(a’+b’+c)

= a’ab+abb’+abc+a’c+b’c+c

= c

(d) A’BD’+ABC’D’+ABCD’

=BD’(A’+AC’+AC)

=BD’(A’+A(C+C’))

=BD’

6. (a) Given the Boolean function E=F1+F2, if F1 and F2 can be expressed by the sum of minterms, then E=F1+F2 is the sum of the minterms of F1 and the sum of the minterims of F2. Such that, the Boolean function E=F1+F2 contains the sum of the minterms of F1 and F2.

(b) Given the Boolean function G=F1*F2, and F1 and F2 can be expressed by the sum of minterms. Then as F1*F2 going on, each minterm of F1 will multiply by each minterm of F2. If the minterm of F1 is different from the minterm of F2, this will result the minterm disappear

since AA’=0. On the other hand, if the minterm of F1 is same as the minterm of F2, the minterm will reserve since AA=A. Such that, the Boolean function G=F1*F2 contains only the minterms that are common to F1 and F2.

7. (a) $F(x, y, z) = \prod(0, 2, 4, 6)$

(b) $F(A, B, C, D) = \sum(0, 1, 2, 4, 6, 7, 9, 10, 12, 14)$

8. (a) $(b + d)(a' + b' + c)(a + c)$

$= (a'b + bc + a'd + b'd + cd)(a + c)$

$= a'ab + a'bc + abc + bc + a'ad + a'cd + ab'd + b'cd + acd + cd$

$= bc + cd + ab'd$

$= (a + a')(d + d')bc + cd(a + a')(b + b') + ab'(c + c')d$

$= a'b'cd + a'bcd' + a'bcd + ab'c'd + ab'cd + abcd' + abcd$

$= \sum(3, 6, 7, 9, 11, 14, 15)$

(b) $a'b + a'c' + bc$

$= a'(b + c') + c(b + c')$

$= (a' + c)(b + c')$

$= (a' + c + b)(a' + c + b')(a + b + c')(a' + b + c')$

$= (a' + b + c)(a' + b' + c)(a + b + c')(a' + b + c')$

$= \prod(1, 4, 5, 6)$

9. (a) Left = y'z' + yz' + x'z = z'(y + y') + zx' = z' + zx' = z'+x'

Right = x' + xz = x' + z ≠ Left

Such that the Boolean equation is false.

(b) Left = x'y' + xz' + yz = x'y'(z + z') + xz'(y + y') + yz(x + x')

= x'y'z + x'y'z + x'yz + xy'z' + xyz' + xyz = ∑(0, 1, 3, 4, 6, 7)

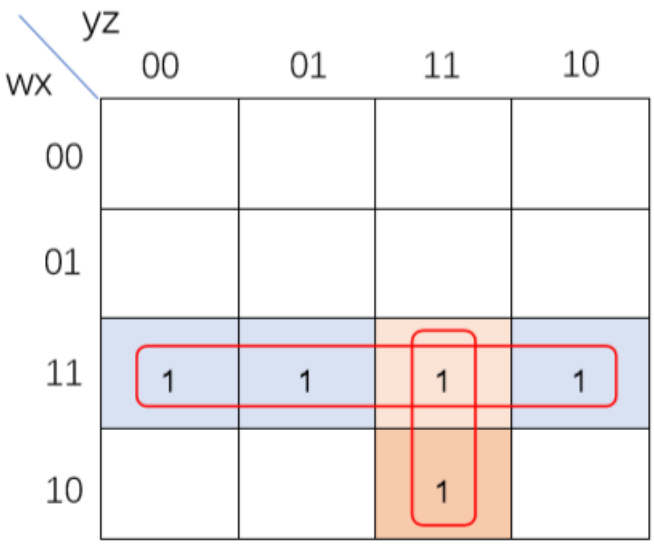
Right = y'z' + xy + x'z = y'z'(x + x') + xy(z + z') + x'z(y + y')

= x'y'z + x'y'z + x'yz + xy'z' + xyz' + xyz = ∑(0, 1, 3, 4, 6, 7) = Left

Such that the Boolean equation is true.

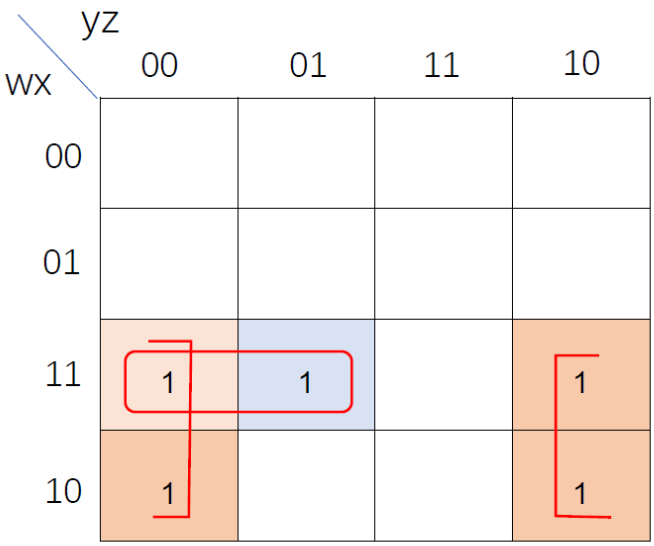
10.

(a)



From the Karnaugh maps, it gives F(w, x, y, z) = wx + wyz

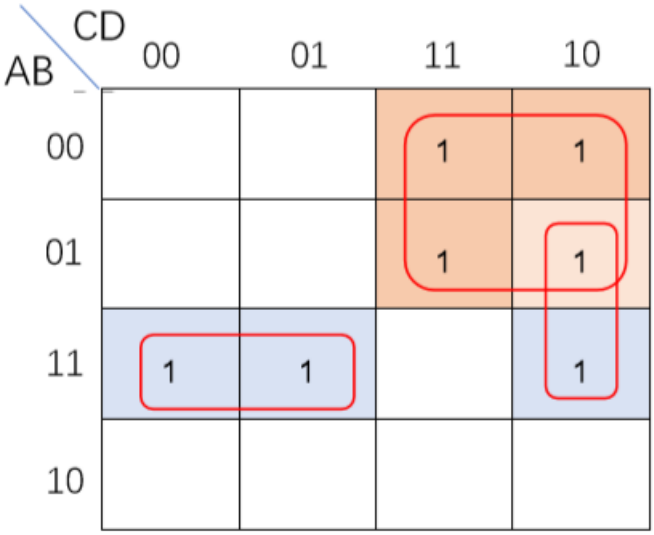
(b)



From the Karnaugh maps, it gives F(w, x, y, z) = wz’ + wxy’

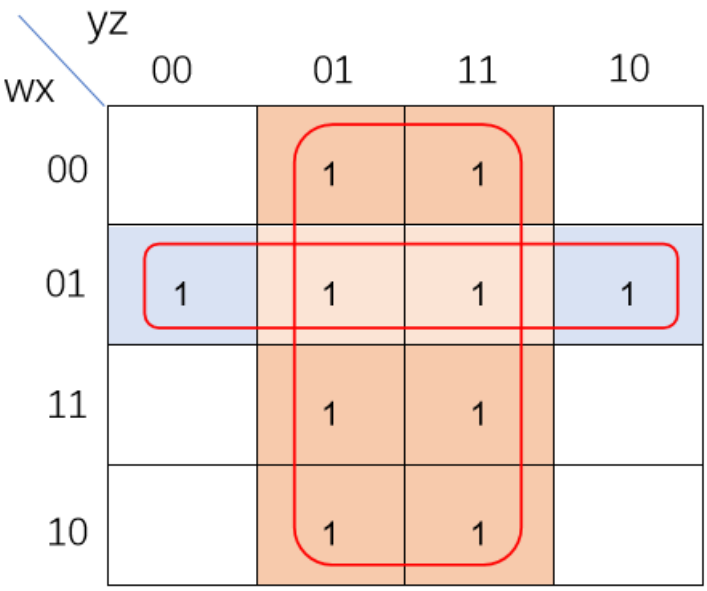
11.

(a)



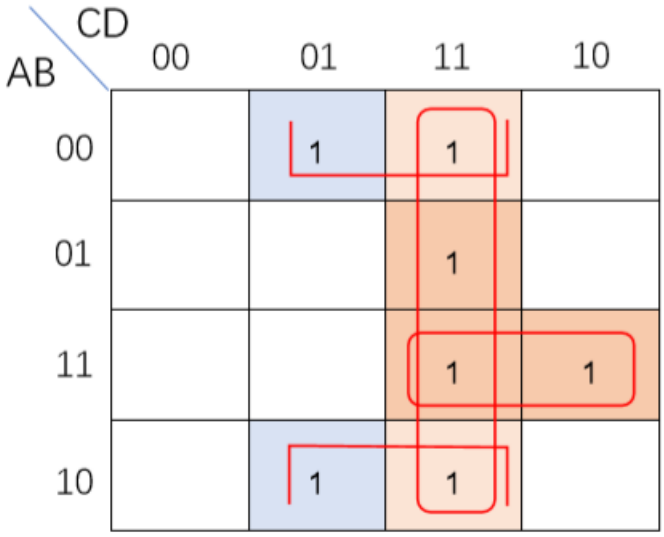
From the Karnaugh maps, it gives F(A, B, C, D) = A'C + ABC' + BCD'

(b)



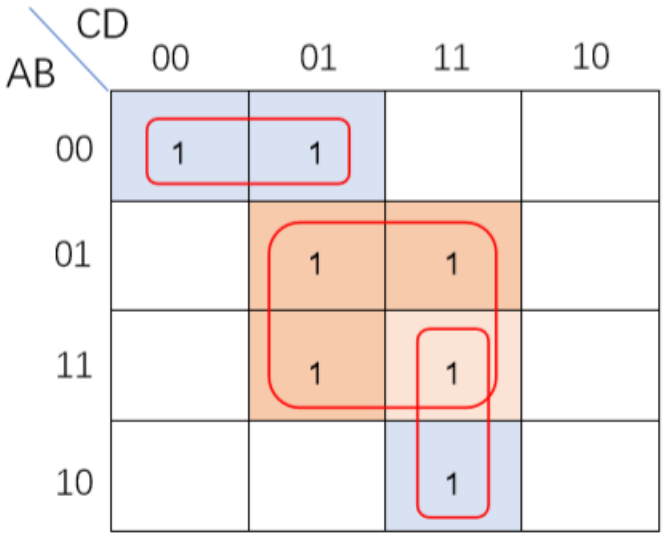
From the Karnaugh maps, it gives F(w, x, y, z) = z + w'x

(c)



From the Karnaugh maps, it gives F(A, B, C, D) = B'D + CD + ABC

(d)



From the Karnaugh maps, it gives F(A, B, C, D) = BD + A'B'C' + ACD

12. (a) F(A, B, C, D) = AD + BC'D + ABC + A'BC'D

= AD + ABC + BC'D

= ((AD)'(BC'D)'(ABC)')'

= ((((AD)'((BC')''D)')'')''((AB)''C)')')''

(b) F(A, B, C, D) = A'B'C'D + CD' + AC'D

= A'B'C'D + CD' + AC'D(1+B')

= A'B'C'D + CD' + AC'D + AB'C'D

= B'C'D + AC'D + CD'

= ((B’C’D)’(AC’D)’(CD’)’)’

= ((((B’C’)’’D)’((AC’)’’D)’)’’(CD’)’)’

(c) F(A, B, C, D) = (A’ + C’ + D’)(A’ + C’)(C’ + D’)

= (ACD)’(AC)’(CD)’

= (ACD + AC + CD)’

= (AC + CD)’

= C’ + A’D’

= (C(A’D’)’)’

(d) F(A, B, C, D) = A’ + AB + B’C + ACD

= A’ + B + B’C + ACD

= A’ + B + C + ACD = A’ + B + C

= (AB’C’)’

= ((AB’)’’C’)’

PART 2: DIGITAL DESIGN LAB (TASK1)

DESIGN

Verilog Code:

```
module Priority_Encoder(wards,seg_out,seg_en);
input [3:0]wards;
output reg [7:0] seg_out;
```

```
output [7:0] seg_en;
assign seg_en = 8'b11111110;
always @ *
begin
casex(wards)
   4'b0001: seg_out = 8'b11111001;
   4'b001x: seg_out = 8'b10100100;
   4'b01xx: seg_out = 8'b10110000;
   4'b1xxx: seg_out = 8'b10011001;
   default: seg_out = 8'b11111111;
endcase
end
endmodule
```

Truth-table

wards[3]	*wards[2]*	*wards[1]*	*wards[0]*	*seg_out*	*number*
0	*0*	*0*	*0*	*1111 1111*	
0	*0*	*0*	*1*	*1111 1001*	*1*
0	*0*	*1*	*0*	*1010 0100*	*2*
0	*0*	*1*	*1*	*1010 0100*	*2*
0	*1*	*0*	*0*	*1011 0000*	*3*
0	*1*	*0*	*1*	*1011 0000*	*3*
0	*1*	*1*	*0*	*1011 0000*	*3*

0	1	1	1	1011 0000	3
1	0	0	0	1001 1001	4
1	0	0	1	1001 1001	4
1	0	1	0	1001 1001	4
1	0	1	1	1001 1001	4
1	1	0	0	1001 1001	4
1	1	0	1	1001 1001	4
1	1	1	1	1001 1001	4

SIMULATION

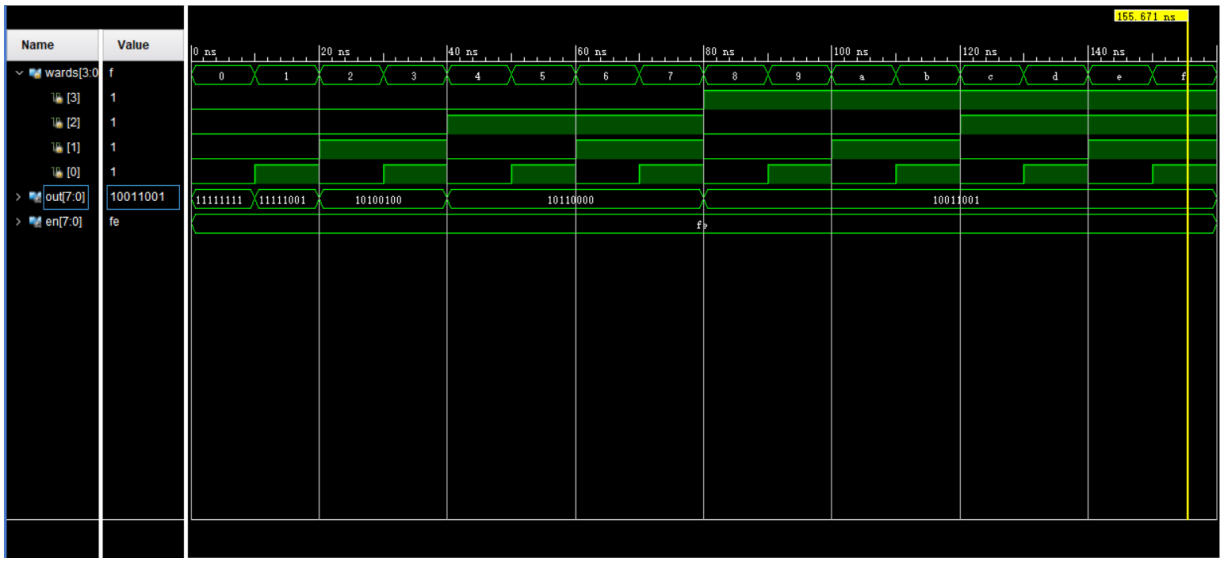
Verilog Code

```
module Pri_sim( );
reg [3:0]wards;
wire [7:0]out;
wire [7:0]en;
Priority_Encoder usim(wards,out,en);

initial
begin
   wards = 4'b0000;
    while(wards<4'b1111)
   begin
```

```
    #10 wards = wards+1;
    end
    #10 $finish(1);
end
endmodule
```

Wave form



From the Wave form:

When 0-10ns, it means no wards turn on the bell, such that the output is (1111 1111), which mean there will not show any number.

When 10-20ns, it means only the ward1 turn on the bell, such that the output is (1010 0100), which mean there will show the number 1.

When 20-40ns, it means there may exist one or more ward turn on the bell, but the highest priority ward is ward2, such that the output is (1010 0100), which mean there will show the number2.

When 40-80ns, it means there may exist one or more wards turn on the bell, but the highest priority ward is ward3, such that the output is (1011 0000), which mean there will show the number3.

When 80-160ns, it means there may exist one or more wards turn on the bell, but the highest priority ward is ward4, such that the output is (1001 1001), which mean there will show the number4.

CONSTRAINT FILE AND THE TESTING

Constraint File

```
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_en[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg_out[7]}]
set_property PACKAGE_PIN A18 [get_ports {seg_en[7]}]
set_property PACKAGE_PIN A20 [get_ports {seg_en[6]}]
set_property PACKAGE_PIN B20 [get_ports {seg_en[5]}]
set_property PACKAGE_PIN E18 [get_ports {seg_en[4]}]
set_property PACKAGE_PIN F18 [get_ports {seg_en[3]}]
set_property PACKAGE_PIN D19 [get_ports {seg_en[2]}]
set_property PACKAGE_PIN E19 [get_ports {seg_en[1]}]
set_property PACKAGE_PIN C19 [get_ports {seg_en[0]}]
set_property PACKAGE_PIN E13 [get_ports {seg_out[7]}]
set_property PACKAGE_PIN C15 [get_ports {seg_out[6]}]
set_property PACKAGE_PIN C14 [get_ports {seg_out[5]}]
set_property PACKAGE_PIN E17 [get_ports {seg_out[4]}]
set_property PACKAGE_PIN F16 [get_ports {seg_out[3]}]
set_property PACKAGE_PIN F14 [get_ports {seg_out[2]}]
set_property PACKAGE_PIN F13 [get_ports {seg_out[1]}]
set_property PACKAGE_PIN F15 [get_ports {seg_out[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {wards[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {wards[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {wards[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {wards[3]}]
set_property PACKAGE_PIN Y9 [get_ports {wards[0]}]
set_property PACKAGE_PIN W9 [get_ports {wards[1]}]
set_property PACKAGE_PIN Y7 [get_ports {wards[2]}]
set_property PACKAGE_PIN Y8 [get_ports {wards[3]}]
```

Analysis

0 0 0 0

Testcase #1:

When no wards turn on the bell,

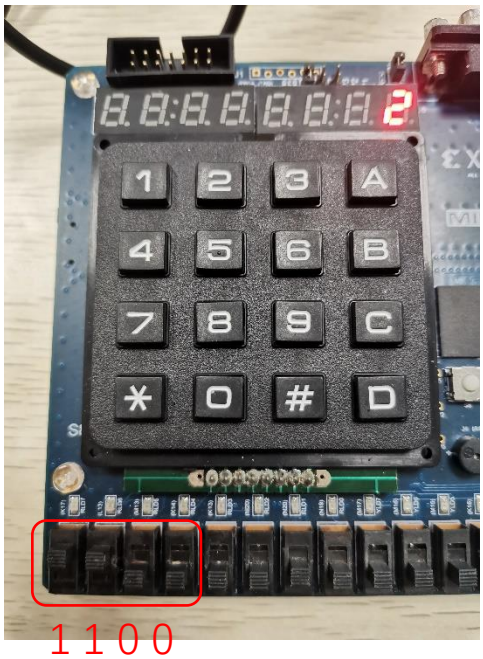
Then the 7-seg tube will not appear anything

1 0 0 0

Testcase #2:

When only the ward1 turn on the bell,

Then the 7-seg tube will appear number 1

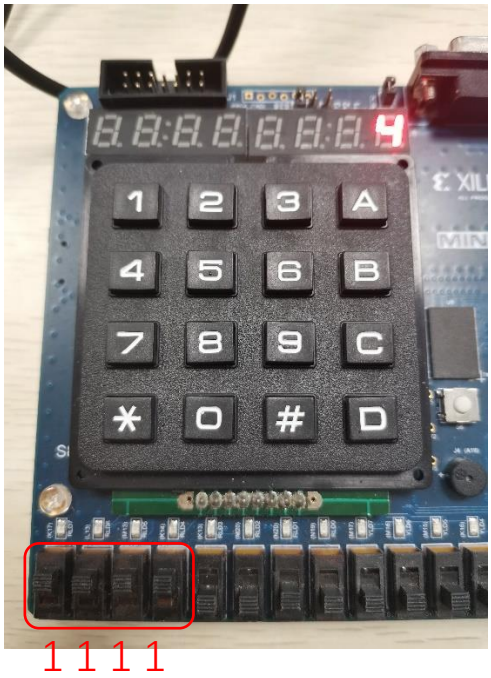


Testcase #3:

When both the ward1 and ward2 turn on the bell, then the 7-seg tube will appear number 2

Testcase #4:

When the ward1, ward2 and ward3 all turn on the bell, then the 7-seg tube will appear number 3



Testcase #5:

When the ward1, ward2, ward3 and ward4 all turn on the bell, then the 7-seg tube will appear number 4.

THE DESCRIPTION OF OPERATION

Problem and Solution

1. *During the task, I found all of the 8 tubes will appear the same value. After I tried sometimes, I set seg_en to 8'b11111110, and then only the most right tube will show the number.*
2. *Another problem is when I do simulation, the result will appear in decimal form, and then I set the radix to the binary form, such that I can get information directly.*

PART 2: DIGITAL DESIGN LAB (TASK2)

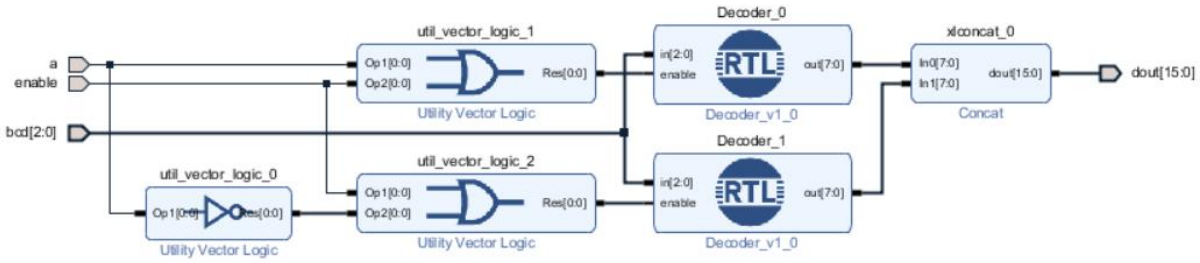
DESIGN

Verilog Code (3-8 Decoder)

```
module Decoder(in,out,enable);
input [2:0]in;
input enable;
```

```
output reg [7:0]out;
always @*
begin
   if(~enable)
   case (in)
        3'b000: out=8'b00000001;
        3'b001: out=8'b00000010;
        3'b010: out=8'b00000100;
        3'b011: out=8'b00001000;
        3'b100: out=8'b00010000;
        3'b101: out=8'b00100000;
        3'b110: out=8'b01000000;
        3'b111: out=8'b10000000;
     endcase
     else
        out=8'b00000000;
end
endmodule
```

Block Design Screen shots



Verilog Code (Design Code)

```
`timescale 1 ps / 1 ps

module decoder38_wrapper(a,bcd,dout,enable);
```

```
input a;
input [2:0]bcd;
output [15:0]dout;
input enable;

wire a;
wire [2:0]bcd;
wire [15:0]dout;
wire enable;

decoder38 decoder38_i
        (.a(a),
        .bcd(bcd),
        .dout(dout),
        .enable(enable));
endmodule
```

Truth table

En	*Input*				*Output*																*Num*
0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*
0	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*1*
0	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*2*
0	*0*	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*3*
0	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*4*
0	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*5*
0	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*6*

0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	7
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	8
0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	9
0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	10
0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	11
0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	12
0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	13
0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14
0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
1	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIMULATION

Verilog Code

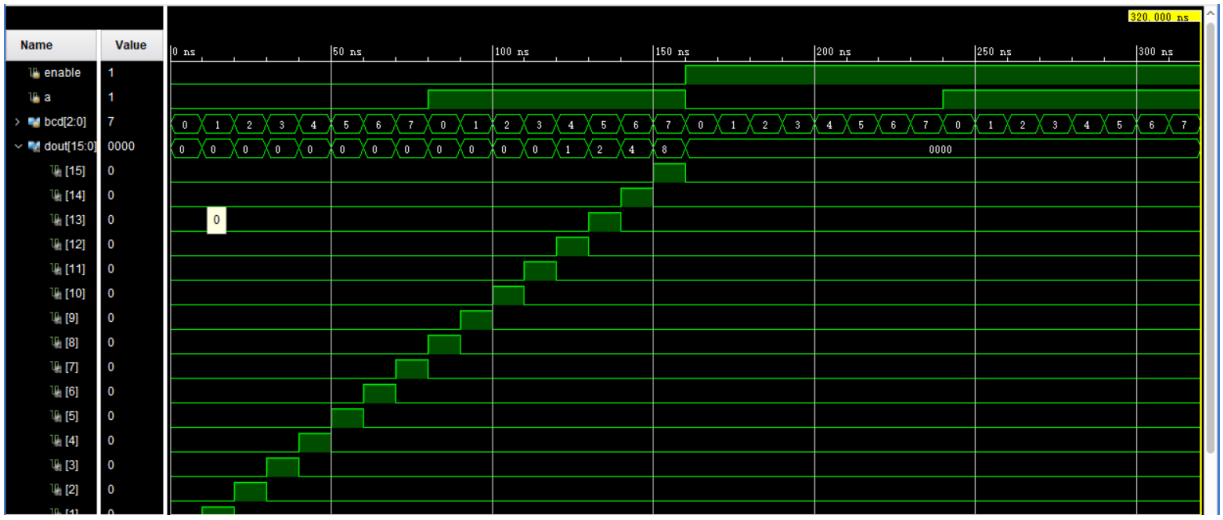
```
`timescale 1ns / 1ps

module decode_sim( );
reg enable;
reg a;
reg [2:0]bcd;
wire [15:0]dout;
decoder38_wrapper simu(a,bcd,dout,enable);

initial
begin
```

```
    {enable,a,bcd}=5'b00000;
    while({enable,a,bcd}<5'b11111)
    begin
    #10 {enable,a,bcd}={enable,a,bcd}+1;
    end
    #10 $finish(1);
end
endmodule
```

Wave Form



From the wave form (the result is in hexadecimal form), we can get:

When 0-150ns, enable=0:

When 0-10ns, the input is 0000, and then the result is (0000 0000 0000 0001)

When 10-20ns, the input is 0001, and then the result is (0000 0000 0000 0010)

When 20-30ns, the input is 0010, and then the result is (0000 0000 0000 0100)

When 30-40ns, the input is 0011, and then the result is (0000 0000 0000 1000)

When 40-50ns, the input is 0100, and then the result is (0000 0000 0001 0000)

When 50-60ns, the input is 0101, and then the result is (0000 0000 0010 0000)

When 60-70ns, the input is 0110, and then the result is (0000 0000 0100 0000)

When 70-80ns, the input is 0111, and then the result is (0000 0000 1000 0000)

When 80-90ns, the input is 1000, and then the result is (0000 0001 0000 0000)

When 90-100ns, the input is 1001, and then the result is (0000 0010 0000 0000)

When 100-110ns, the input is 1010, and then the result is (0000 0100 0000 0000)

When 110-120ns, the input is 1011, and then the result is (0000 1000 0000 0000)

When 120-130ns, the input is 1100, and then the result is (0001 0000 0000 0000)

When 130-140ns, the input is 1101, and then the result is (0010 0000 0000 0000)

When 140-150ns, the input is 1110, and then the result is (0100 0000 0000 0000)

When 150-160ns, the input is 1111, and then the result is (1000 0000 0000 0000)

When 160-320ns, enable=1: then the result is (0000 0000 0000 0000)

CONSTRAINT FILE AND THE TESTING

Constraint File

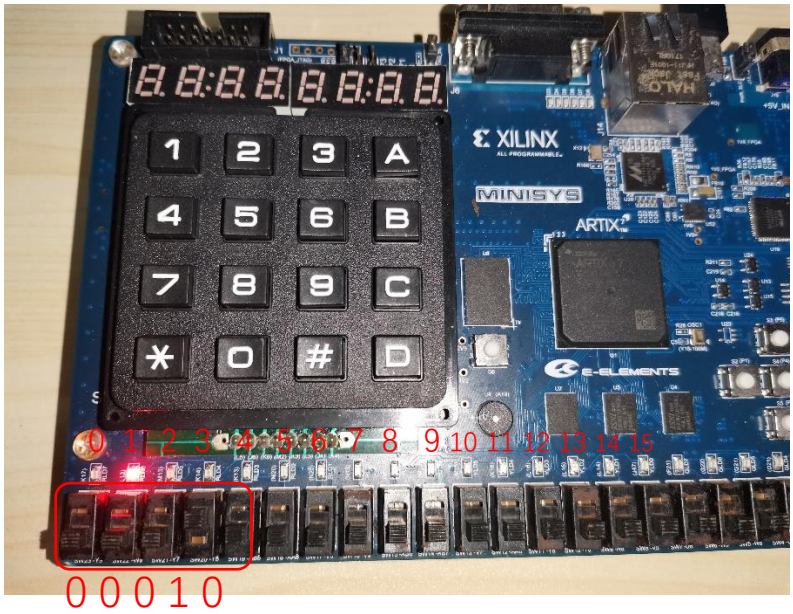
```
set_property IOSTANDARD LVCMOS33 [get_ports {bcd[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {bcd[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[15]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[14]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[13]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[12]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports a]
set_property IOSTANDARD LVCMOS33 [get_ports enable]
```

```
set_property PACKAGE_PIN W9 [get_ports {bcd[2]}]
set_property PACKAGE_PIN Y7 [get_ports {bcd[1]}]
set_property PACKAGE_PIN Y8 [get_ports {bcd[0]}]
set_property PACKAGE_PIN K17 [get_ports {dout[0]}]
set_property PACKAGE_PIN L13 [get_ports {dout[1]}]
set_property PACKAGE_PIN M13 [get_ports {dout[2]}]
set_property PACKAGE_PIN K14 [get_ports {dout[3]}]
set_property PACKAGE_PIN K13 [get_ports {dout[4]}]
set_property PACKAGE_PIN M20 [get_ports {dout[5]}]
set_property PACKAGE_PIN N20 [get_ports {dout[6]}]
set_property PACKAGE_PIN N19 [get_ports {dout[7]}]
set_property PACKAGE_PIN M17 [get_ports {dout[8]}]
set_property PACKAGE_PIN M18 [get_ports {dout[9]}]
set_property PACKAGE_PIN M15 [get_ports {dout[10]}]
set_property PACKAGE_PIN K16 [get_ports {dout[11]}]
set_property PACKAGE_PIN L16 [get_ports {dout[12]}]
set_property PACKAGE_PIN L15 [get_ports {dout[13]}]
set_property PACKAGE_PIN L14 [get_ports {dout[14]}]
set_property PACKAGE_PIN J17 [get_ports {dout[15]}]
set_property PACKAGE_PIN Y9 [get_ports a]
set_property PACKAGE_PIN AB8 [get_ports enable]
```

Analysis

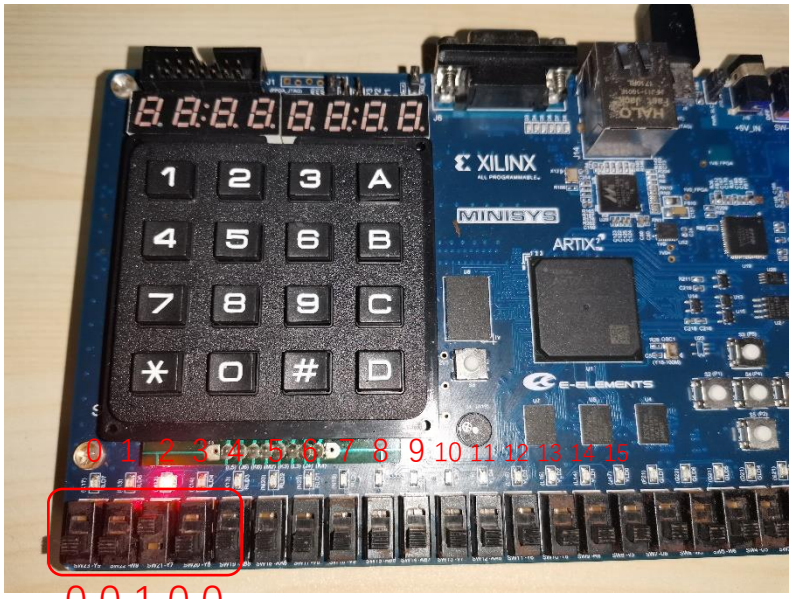
When the input is 0000,

The No.0 light turn on



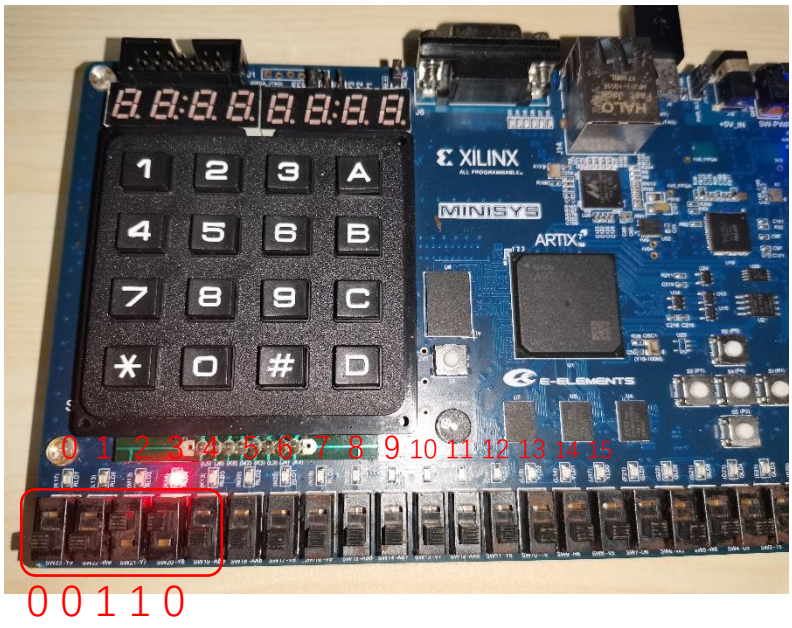
When the input is 0001,

The No.1 light turn on



When the input is 0010,

The No.2 light turn on

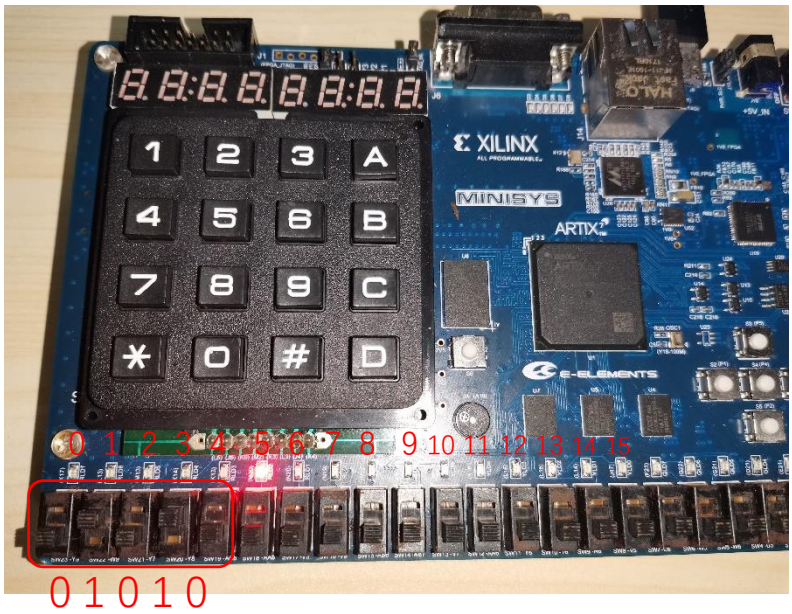


When the input is 0011,

The No.3 light turn on

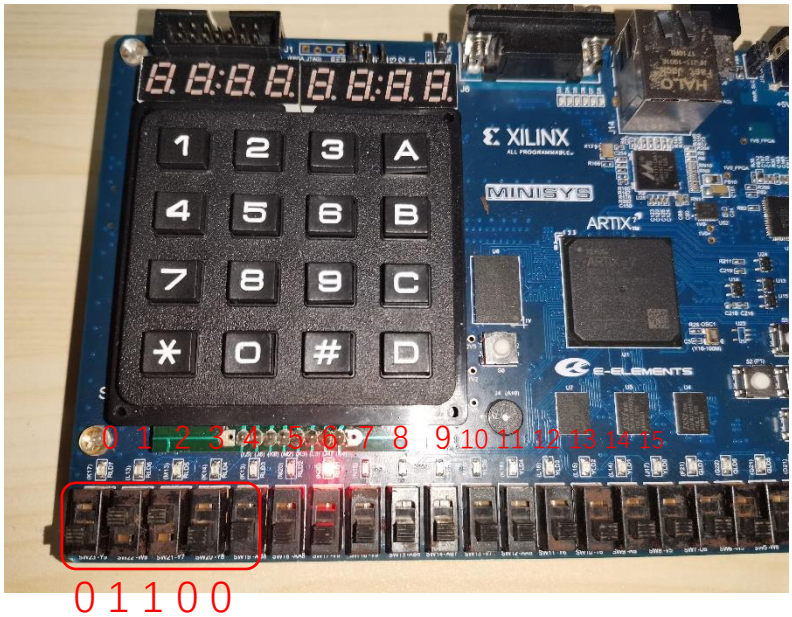
When the input is 0100,

The No.4 light turn on



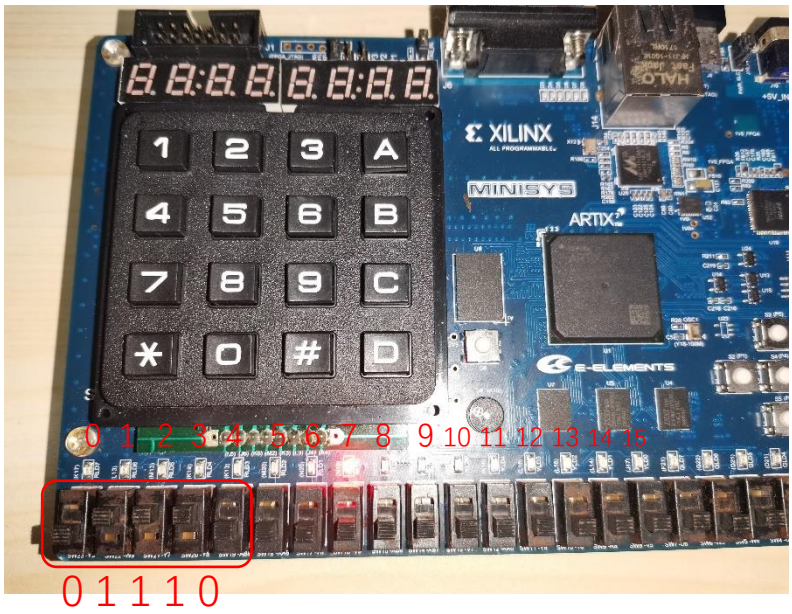
When the input is 0101,

The No.5 light turn on



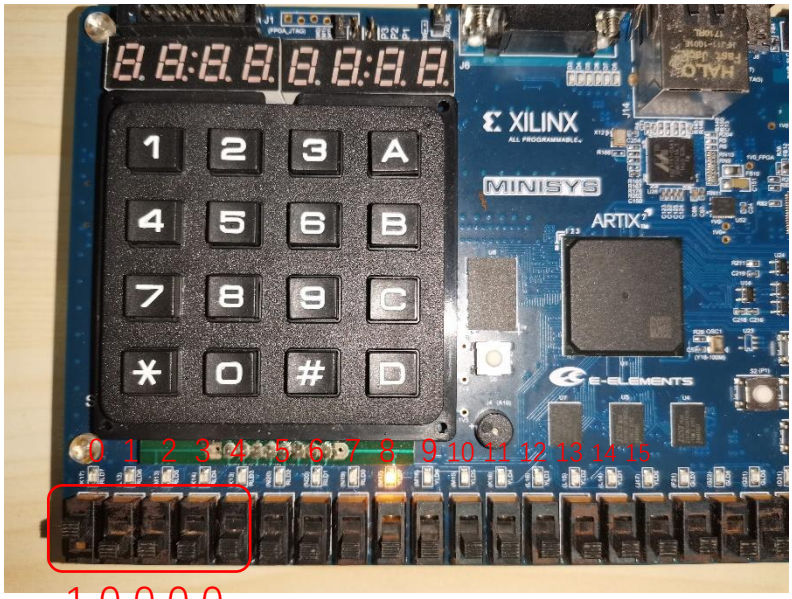
When the input is 0110,

The No.6 light turn on



When the input is 0111,

The No.7 light turn on

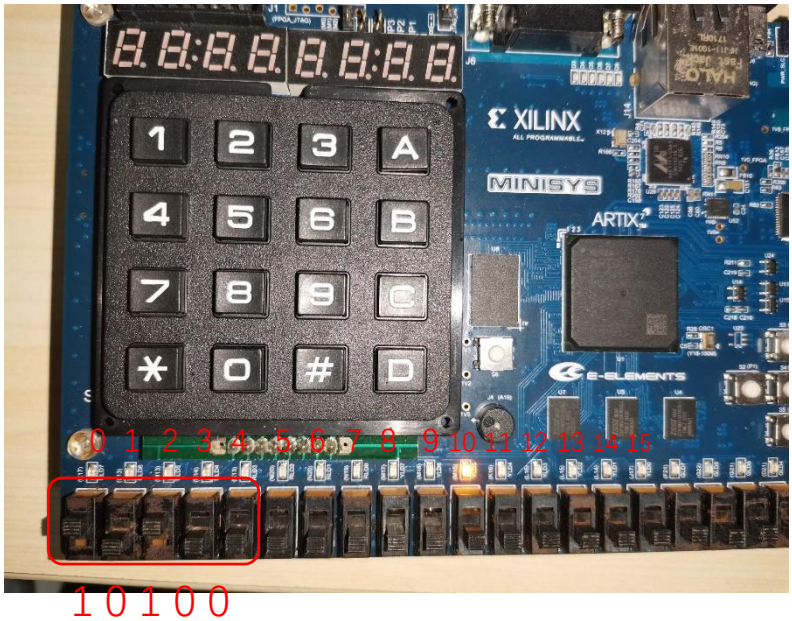


When the input is 1000,

The No.8 light turn on

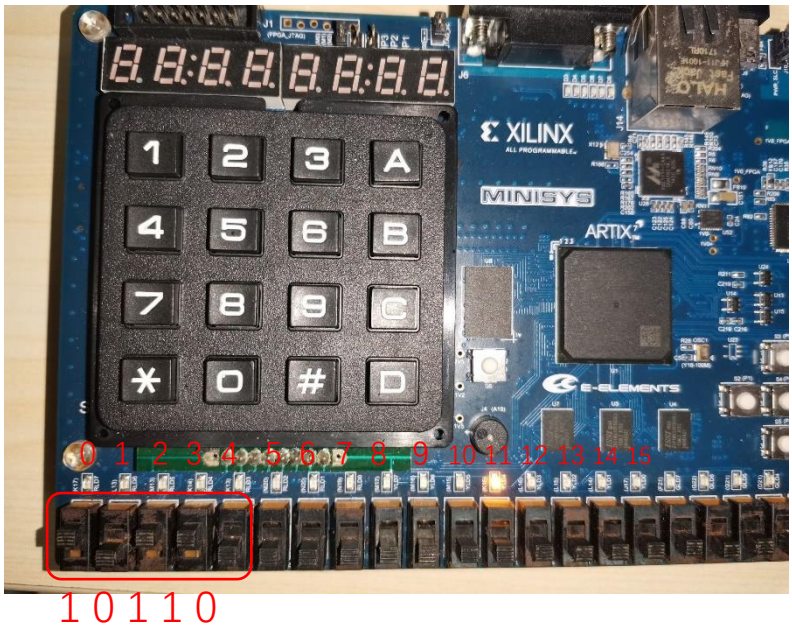
When the input is 1001,

The No.9 light turn on



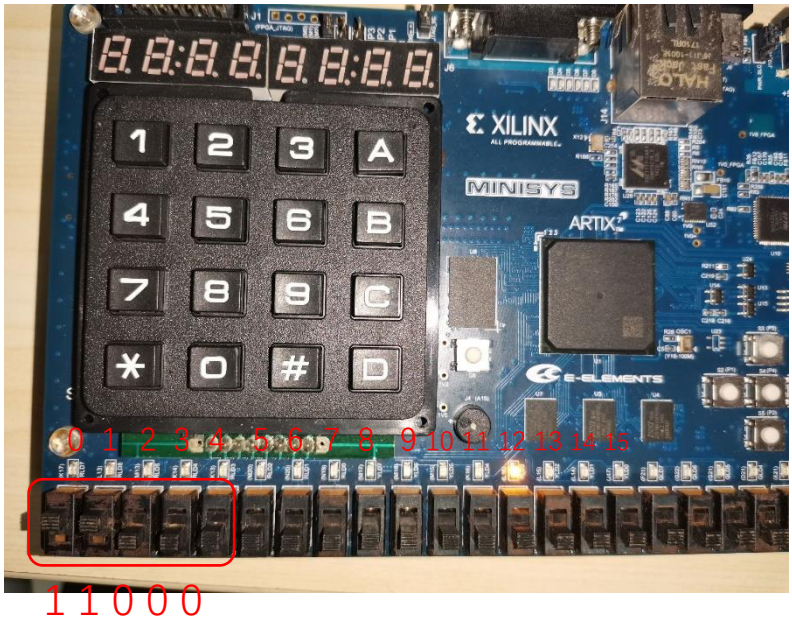
When the input is 1010,

The No.10 light turn on



When the input is 1011,

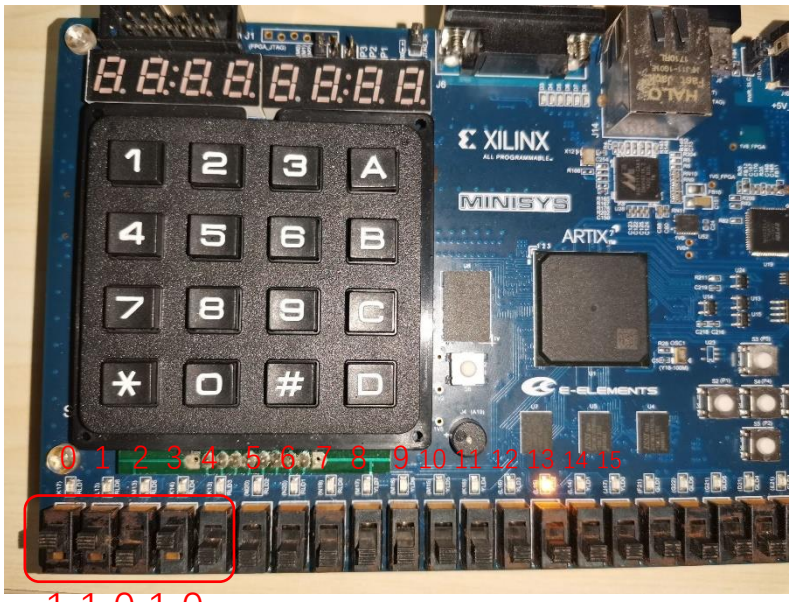
The No.11 light turn on



1 1 0 0 0

When the input is 1100,

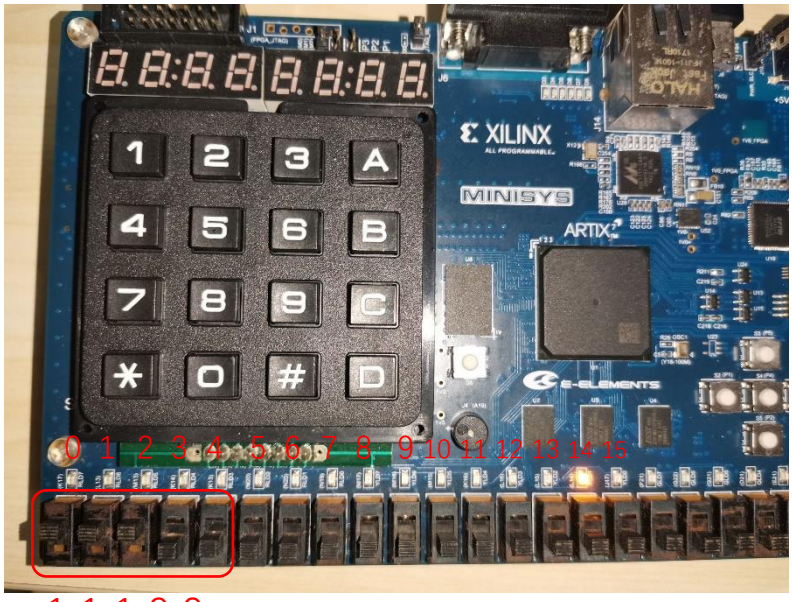
The No.12 light turn on



1 1 0 1 0

When the input is 1101,

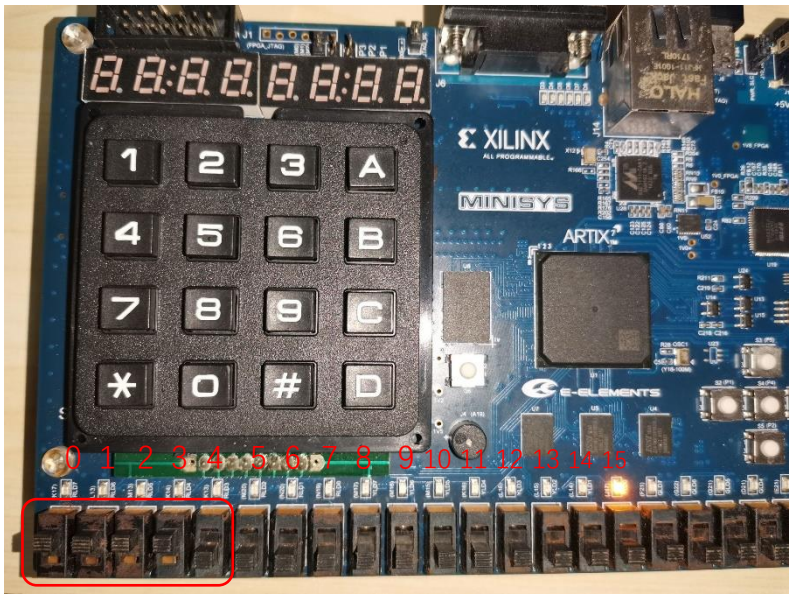
The No.13 light turn on



1 1 1 0 0

When the input is 1110,

The No.14 light turn on



1 1 1 1 0

When the input is 1111,

The No.15 light turn on

THE DESCRIPTION OF OPERATION

1. *First, I made some mistakes in my simulation files. This resulted in the simulation error. But I found mistakes quickly, and then simulated successfully.*
2. *Another problem is that it reported error when I generate bitstream, and finally I found I made some mistakes in my constraint file.*
3. *During this task, I mastered the usage of block design and learned the principle of the decoder.*