

**DIGITAL DESIGN**

**ASSIGNMENTREPORT**

**ASSIGNMENT ID : 03**

**Student Name: Weibao Fu**

**Student ID: 11812202**

PART 1: DIGITAL design THEORY

1. Y = AB(D + C) + (BC’ + DE’) + BD’

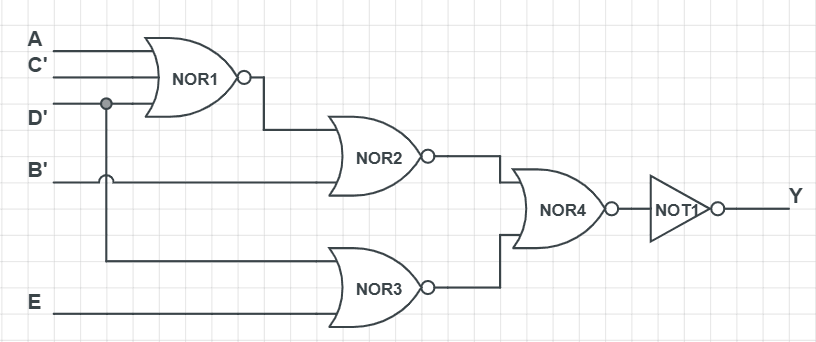
= ABD + ABC + BC’ + DE’ +BD’

= AB + BC’ + BD’ + DE’

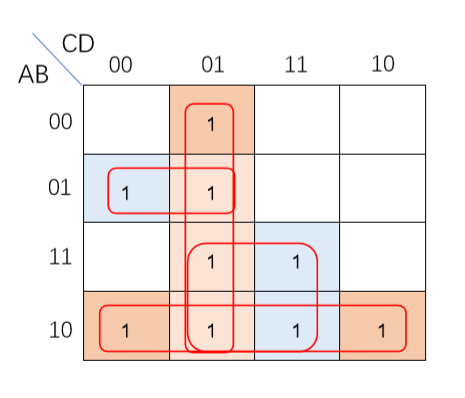
= B(A + C’ + D’) + (D’ + E)’

= (B’ + (A + C’ +D’)’)’ + (D’ + E)’

= ((B’ + (A + C’ +D’)’)’ + (D’ + E)’)’’



1. From the Boolean Function, we can get this Karnaugh maps.



We can get the simple function of F is F = AB’ + AD + C’D + A’BC’

1. F = AB’ + AD + C’D + A’BC’

= AB’ + AD + C’D + ((A’B)’ + C)’

= (((AB’ + AD) + C’D) + ((A’B)’ + C)’)

1. F = AB’ + AD + C’D + A’BC’

= (A’ + B)’ + (A’ + D’)’ + (C + D’)’ + (A + B’ + C)’

= ((A’ + B) (A’ + D’))’ + (C + D’)’ + ((A + B’)’C’)’’

= ((A’ + B) (A’ + D’))’ + ((C + D’) ((A + B’)’C’)’)’

= (((A’ + B) (A’ + D’))’’ ((C + D’) ((A + B’)’C’)’)’’)’

1. F = AB’ + AD + C’D + A’BC’

= (A’ + B)’ + (A’ + D’)’ + (C + D’)’ + (A + B’ + C)’

= ((A’ + B)’ + (A’ + D’)’)’’ + ((C + D’)’ + ((A + B’)’’ + C)’)’’

1. F = AB’ + AD + C’D + A’BC’

= ((AB’)’(AD)’(C’D)’(A’BC’)’)’

= (((AB’)’(AD)’)’’((C’D)’((A’B)’’C’)’)’’)’

1. F = AB’ + AD + C’D + A’BC’

= (A + C’)(A’ + B’ + D)(B’ + C’ + D)(A + B + D)

= ((A + C’)((A’ + B’) + D))(((B’ + C’) + D)((A + B) + D))

1. F’ = ∑(0, 2, 3, 6, 7, 12, 14)

= A’C + ABD’ + A’B’D’

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

1. F = AB’ + AD + C’D + A’BC’

= ((AB’)’(AD)’(C’D)’(A’BC’)’)’

= (((AB’)’(AD)’)’’(C’D)’(((A’B)’’C’)’)’’)’

Four-bit parity generator

\*Truth table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | Y(odd) | Y(even) |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

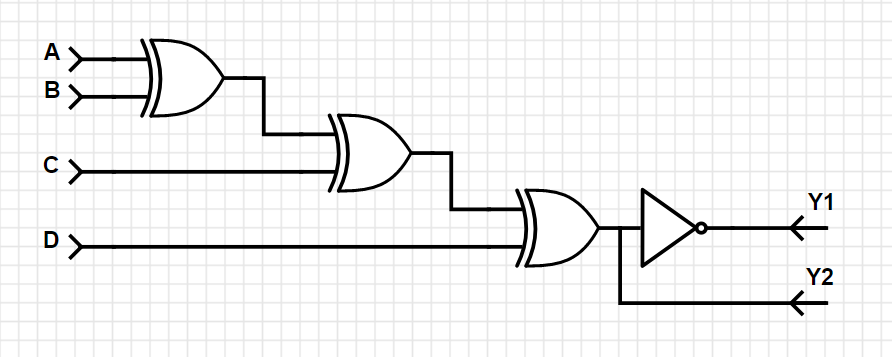
\*Y(odd)=A’B’C’D’+A’B’CD+A’BC’D+AB’C’D + A’BCD’ + AB’CD’ + ABC’D’ + ABCD

= (A⊕B⊕C⊕D)’

Y(even)=A’B’C’D+A’B’CD’+A’BC’D’+AB’C’D’+ A’BCD + AB’CD + ABC’D + ABCD’

= A⊕B⊕C⊕D

\*The circuit diagram



Three-bit parity checker

Four-bit parity generator

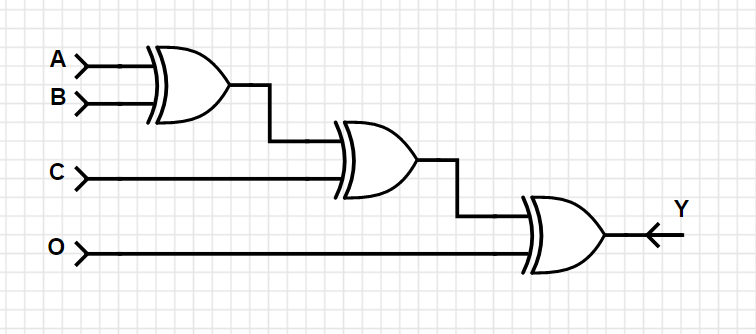
\*Truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | O | Y |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

\*Y=A’B’C’O+A’B’CO’+A’BC’O’+AB’C’O’+ A’BCO + AB’CO + ABC’O + ABCO’

= A⊕B⊕C⊕O

Circuit diagram



##### From the given problem, we can get the truth table.

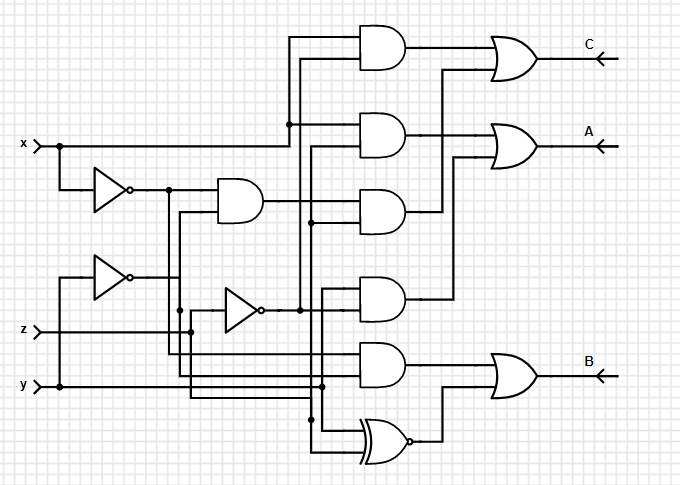
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | z | A | B | C |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

##### Such that, we can get A(x, y, z) = yz' + xz

##### B(x, y, z) = x’y’ + y⊙z

##### C(x, y, z) = xz’ + x’y’z

##### Circuit diagram



##### 5.(Here we assume that the 7-seg tube is high level active)

##### Truth table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

##### Such that a(A, B, C, D) = ∑(0,2,3,5,6,7,8,9) = A’C + B’C’D’ + A’BD + AB’C’

##### b(A, B, C, D) = ∑(0,1,2,3,4,7,8,9) = A’CD + A’C’D’ + A’B’ + B’C’

##### c(A, B, C, D) = ∑(0,1,3,4,5,6,7,8,9) = A’B + A’C’ + A’D + B’C’

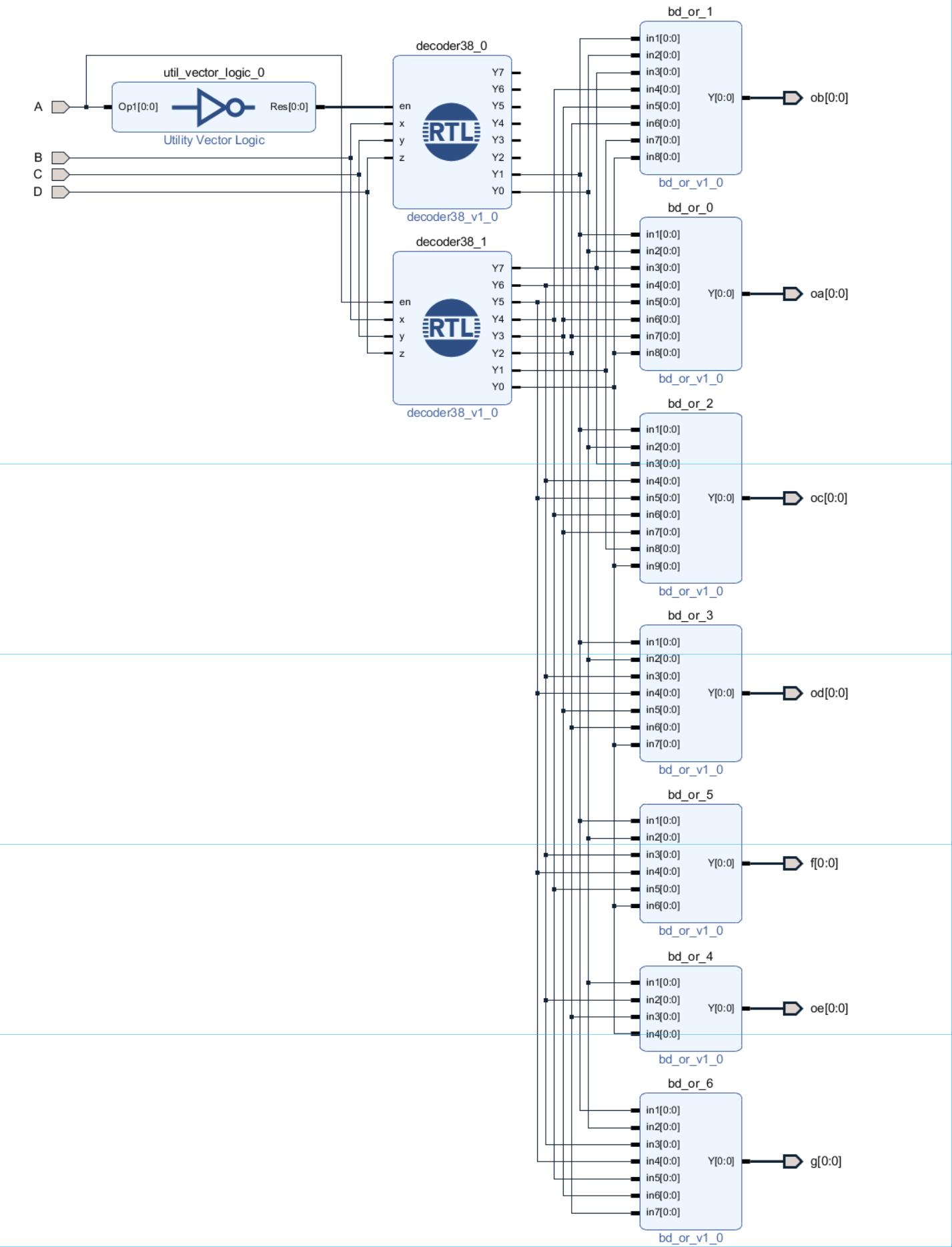
##### d(A, B, C, D) = ∑(0,2,3,5,6,8,9)

##### = A’B’C + A’B’D’ + A’BC’D + A’CD’ + AB’C’

##### e(A, B, C, D) = ∑(0,2,6,8) = A’CD’ + B’C’D’

##### f(A, B, C, D) = ∑(0,4,5,6,8,9) = AB’C’ + A’BC’ + A’BD’ + A’C’D’

##### g(A, B, C, D) = ∑(2,3,4,5,6,8,9) = AB’C’ + A’BC’ + A’B’C + A’CD’



##### 6. Assume the unsigned number are A4A3A2A1, B4B3B2B1.

Then we can get We set Cin as carry. From this, if

we multiple Bi with A4A3A2A1 one by one, and we add the result, then we set the highest as Cin, and participate the next adder. As a result, we can get the answer.

The logic formula is that:

Assume Adder(abcd, edfg) = rmnpq, such that Ci = q and Cin = r.

Then the next step is Add(a’b’c’d’, rd’f’g’).

The circuit diagram:

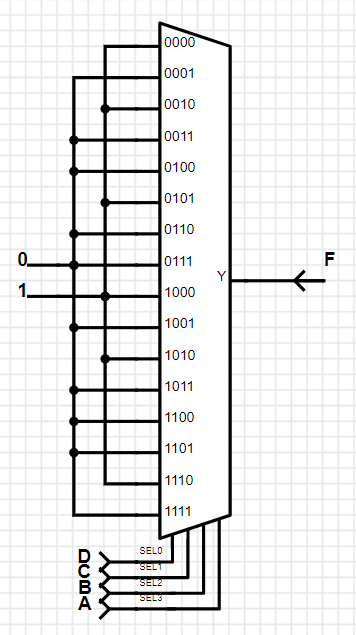
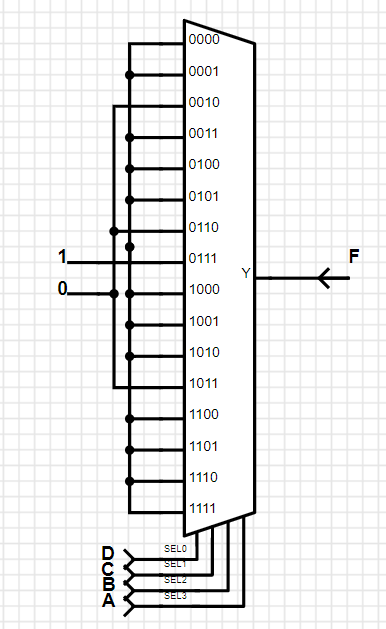


7. （a） Since F (A, B, C, D) =∑ (0, 2, 5, 8, 10, 14)

(b) Since F (A, B, C, D) = ∏ (2, 6, 11)

= ∑ (0, 1, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15)

We can implement the Boolean function as below.

(a) (b)

8. Assume A is subtrahend, B is minuend, C is a bit that borrow from the lower position, D is the result, E is a bit that borrow from the higher position.

Truth table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Such that, D (A, B, C) = ∑(1, 2, 4, 7) E(A, B, C) = ∑(1, 2, 3, 7)

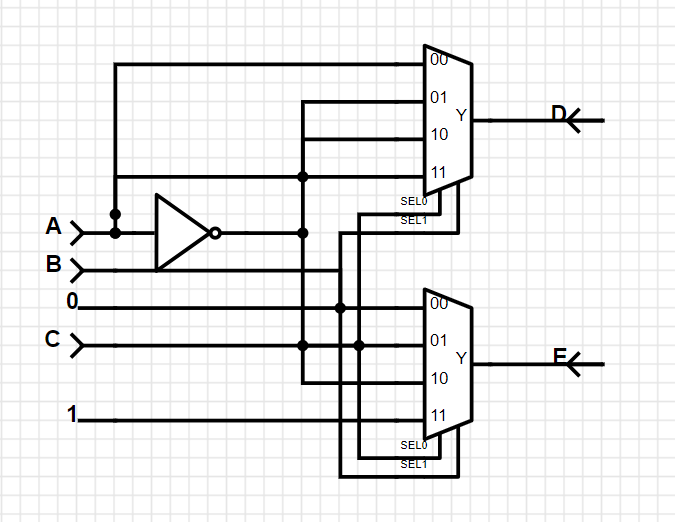
Since D = D10A1’A0’ + D11A1’A0 + D12A1A0’ + D13A1A0

LET A1 = B, A0 = C, THEN D10=D13=A, D11=D12=A’

Since E = D20A1’A0’ + D21A1’A0 + D22A1A0’ + D23A1A0

LET A1 = B, A0 = C, THEN D20=0, D21=D22=A’, D23=1

Circuit diagram:



9. Since F = I0D0 + I1D1 + I2D2 + I3D3 + I4D4 + I5D5 + I6D6 + I7D7

= C’B’A’+DC’B’A+C’BA’+D’C’BA+D’CB’A+DCBA’

= (D+D’)C’B’A’+DC’B’A+(D+D’)C’BA’+D’C’BA+D’CB’A+DCBA’

F(A, B, C, D) = ∑(0,1,4,5,7,9,10,12)

PART 2: DIGITAL design LAB (Task1)

##### Design

***Multiplexer***

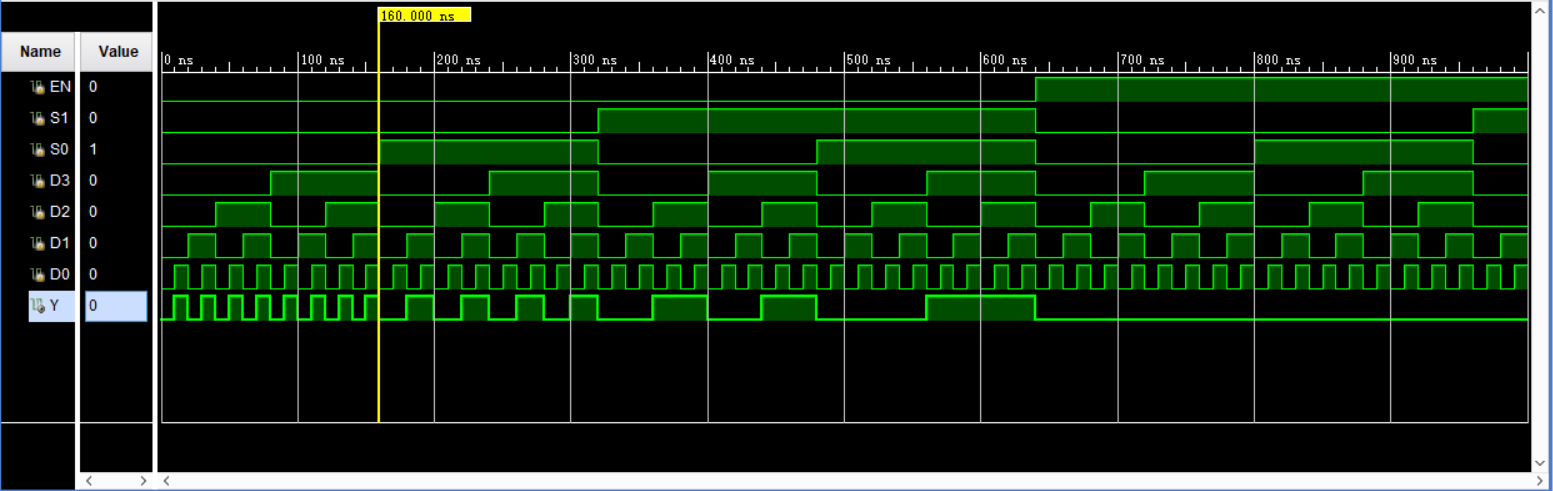
***\*****Design Verilog Code*

`timescale 1ns / 1ps  
module Multiplexer74151(EN,S1,S0,D3,D2,D1,D0,Y);  
input EN,S1,S0,D3,D2,D1,D0;  
output reg Y;  
always @\*  
if(~EN)  
   case({S1,S0})  
       2'b00: Y=D0;  
      2'b01: Y=D1  
       2'b10: Y=D2;  
       2'b11: Y=D3;  
   endcase  
else  
   Y=1'b0;  
endmodule

***\*****Simulation Verilog Code*

`timescale 1ns / 1ps  
module sim\_1( );  
reg EN,S1,S0,D3,D2,D1,D0;  
wire Y;  
Multiplexer74151 usim(EN,S1,S0,D3,D2,D1,D0,Y);  
initial  
begin  
   {EN,S1,S0,D3,D2,D1,D0} = 7'b0000000;  
   while({EN,S1,S0,D3,D2,D1,D0}<7'b1111111)  
   begin  
       #10 {EN,S1,S0,D3,D2,D1,D0} = {EN,S1,S0,D3,D2,D1,D0}+1;  
   end  
   #10 $finish(1);  
end  
endmodule

*\*Wave Form*



*From the wave form, we can get:*

*When 0-160ns: EN=0, and {S1,S0}=2’b00, such that the wave of Y is same as D0.*

*When 170-320ns: EN=0, and {S1,S0}=2’b01, such that the wave of Y is same as D1.*

*When 330-480ns: EN=0, and {S1,S0}=2’b10, such that the wave of Y is same as D2.*

*When 490-640ns: EN=0, and {S1,S0}=2’b11, such that the wave of Y is same as D3.*

*When 650-1280ns: EN=1, such that Y=0;*

***Full subtractor***

Assume A is subtrahend, B is minuend, C is a bit that borrow from the lower position, D is the result, E is a bit that borrow from the higher position.

\*Truth table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

*\*Design Verilog Code*

`timescale 1ns / 1ps  
module Sub(A,B,C,D,E);  
input A,B,C;  
output D,E;  
wire zero = 0 ;  
wire one = 1;  
Multiplexer74151 mult1(zero,B,C,A,~A,~A,A,D);  
Multiplexer74151 mult2(zero,B,C,one,~A,~A,zero,E);  
endmodule

##### simulation

*Simulation Verilog Code*

`timescale 1ns / 1ps  
module sim\_sub( );  
reg A,B,C;  
wire D,E;  
Sub usim(A,B,C,D,E);  
initial  
begin  
   {A,B,C} = 3'b000;  
   while({A,B,C}<3'b111)  
   begin  
       #10 {A,B,C}={A,B,C}+1;  
   end  
   #10 $finish();  
end  
endmodule

*Wave Form*



*From the Wave Form, we can get:*

*When 0-10ns, {A,B,C}=3’b000, then D=0, E=0;*

*When 10-20ns, {A,B,C}=3’b001, then D=1, E=1;*

*When 20-30ns, {A,B,C}=3’b010, then D=1, E=1;*

*When 30-40ns, {A,B,C}=3’b011, then D=0, E=1;*

*When 40-50ns, {A,B,C}=3’b100, then D=1, E=0;*

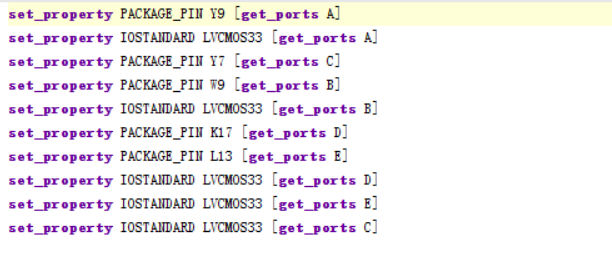
*When 50-60ns, {A,B,C}=3’b101, then D=0, E=0;*

*When 60-70ns, {A,B,C}=3’b110, then D=0, E=0;*

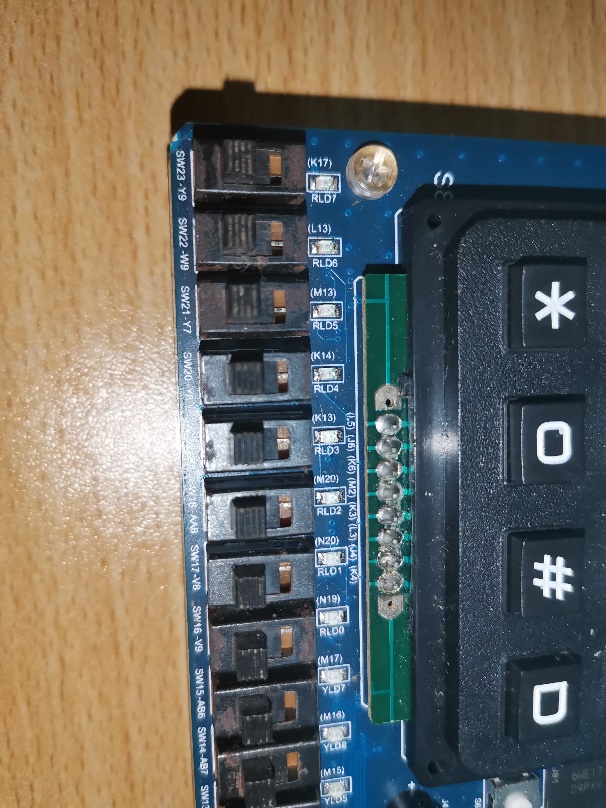
*When 70-80ns, {A,B,C}=3’b111, then D=1, E=1;*

##### Constraint file and the testing

*Constraint File*



*Analysis*

**

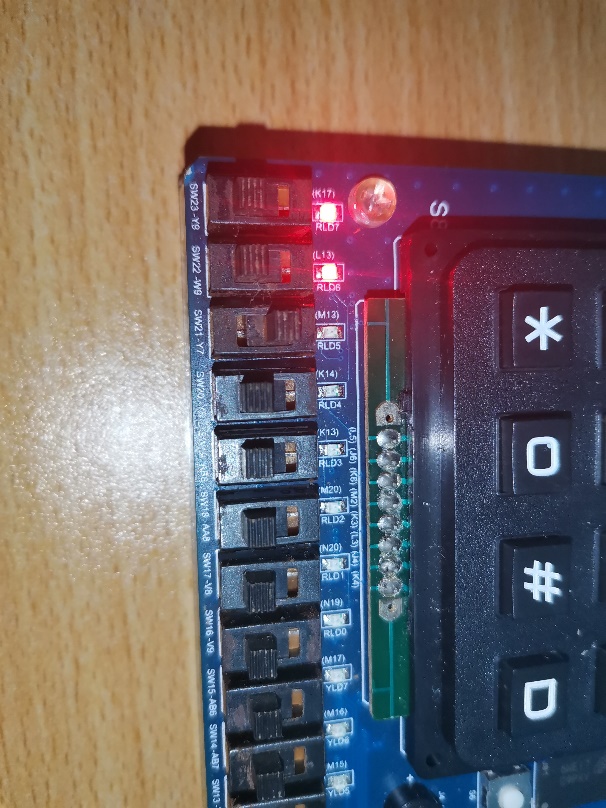
*Testcase #1*

*When input is 0 - 0,*

*And it borrow from lower position is 0*

*Then the output is 0, the borrow from higher position is 0*

0 0 0

**

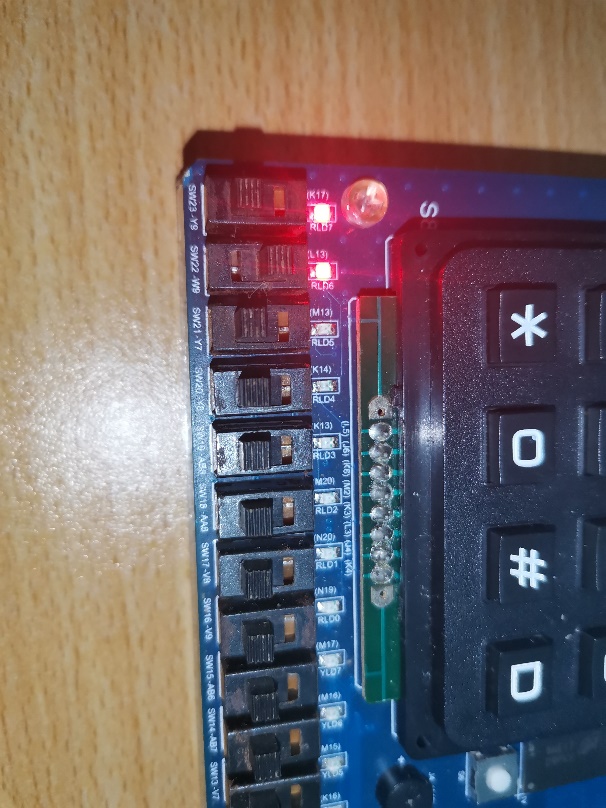
*Testcase #2*

*When input is 0 - 0,*

*And it borrow from lower position is 1*

*Then the output is 1, the borrow from higher position is 1*

0 0 1

**

*Testcase #3*

*When input is 0 - 1,*

*And it borrow from lower position is 0*

*Then the output is 1, the borrow from higher position is 1*

0 1 0

**

*Testcase #4*

*When input is 0 - 1,*

*And it borrow from lower position is 1*

*Then the output is 0, the borrow from higher position is 1*

0 1 1

**

*Testcase #5*

*When input is 1 - 0,*

*And it borrow from lower position is 0*

*Then the output is 1, the borrow from higher position is 0*

1 1 0 0

**

*Testcase #6*

*When input is 1 - 0,*

*And it borrow from lower position is 1*

*Then the output is 0, the borrow from higher position is 0*

1 0 1

**

*Testcase #7*

*When input is 1 - 1,*

*And it borrow from lower position is 0*

*Then the output is 0, the borrow from higher position is 0*

1 1 0

**

*Testcase #8*

*When input is 1 - 1,*

*And it borrow from lower position is 1*

*Then the output is 1, the borrow from higher position is 1*

1 1 1

##### the description of operation

*No Problem*

PART 2: DIGITAL design LAB (Task2)

##### Design

***3-8 Decoder***

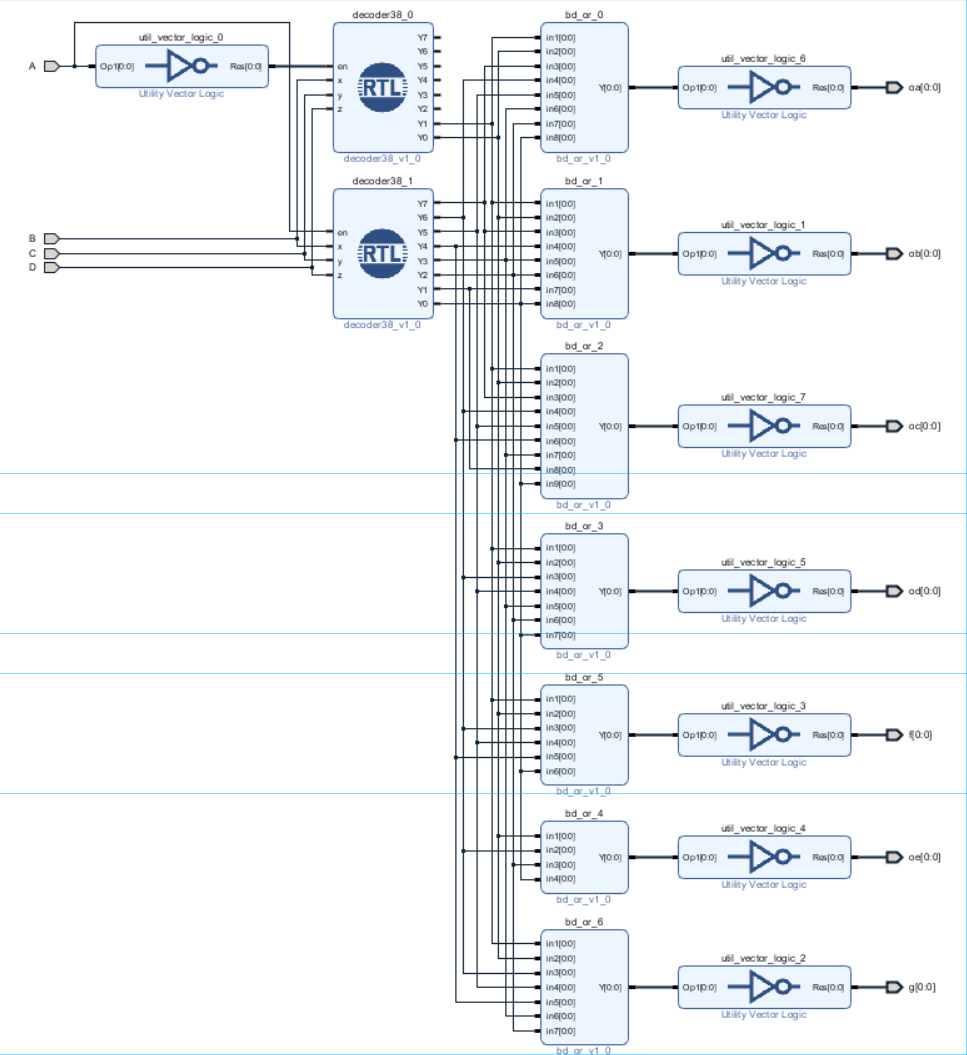
*\*Verilog Code*

`timescale 1ns / 1ps  
module decoder38(en,x,y,z,Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0);  
input en,x,y,z;  
output reg Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0;  
always @\*  
begin  
   if(~en)  
       case({x,y,z})  
           3'b000: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b00000001;  
           3'b001: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b00000010;  
           3'b010: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b00000100;  
           3'b011: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b00001000;  
           3'b100: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b00010000;  
           3'b101: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b00100000;  
           3'b110: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b01000000;  
           3'b111: {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b10000000;  
        endcase  
    else {Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0} = 8'b00000000;  
end  
endmodule

*\*Truth table (The 7-seg tube is low active)*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

***ABCD-to-7 segment***



*\*Design Verilog Code*

`timescale 1 ps / 1 ps  
module ABCD\_seg\_wrapper(A,B,C,D,f,g,oa,ob,oc,od,oe);  
 input wire A,B,C,D;  
 output wire f,g,oa,ob,oc,od,oe;  
 ABCD\_seg ABCD\_seg\_i(.A(A),.B(B),.C(C),.D(D),.f(f),.g(g),  
.oa(oa),.ob(ob),.oc(oc),.od(od),.oe(oe));  
endmodule

##### simulation

*Simulation Verilog Code*

`timescale 1ns / 1ps  
module sim\_u( );  
reg A,B,C,D;  
wire a,b,c,d,e,f,g;  
ABCD\_seg\_wrapper u(A,B,C,D,f,g,a,b,c,d,e);  
initial  
begin  
   {A,B,C,D}=4'b0000;  
   while({A,B,C,D}<4'b1111)  
   begin  
       #10 {A,B,C,D}={A,B,C,D}+1;  
   end  
   #10 $finish();  
end  
endmodule

*Wave Form*



*From the Wave Form*

*0-10ns, {A,B,C,D}=0000, {a,b,c,d,e,f,g}=0000001, which display 0 in 7-seg tube*

*10-20ns, {A,B,C,D}=0001, {a,b,c,d,e,f,g}=1001111, which display 1 in 7-seg tube;*

*20-30ns, {A,B,C,D}=0010, {a,b,c,d,e,f,g}=0010010, which display 2 in 7-seg tube;*

*30-40ns, {A,B,C,D}=0011, {a,b,c,d,e,f,g}=0000110, which display 3 in 7-seg tube;*

*40-50ns, {A,B,C,D}=0100, {a,b,c,d,e,f,g}=1001100, which display 4 in 7-seg tube;*

*50-60ns, {A,B,C,D}=0101, {a,b,c,d,e,f,g}=0100100, which display 5 in 7-seg tube;*

*60-70ns, {A,B,C,D}=0110, {a,b,c,d,e,f,g}=0100000, which display 6 in 7-seg tube;*

*70-80ns, {A,B,C,D}=0111, {a,b,c,d,e,f,g}=0001111, which display 7 in 7-seg tube;*

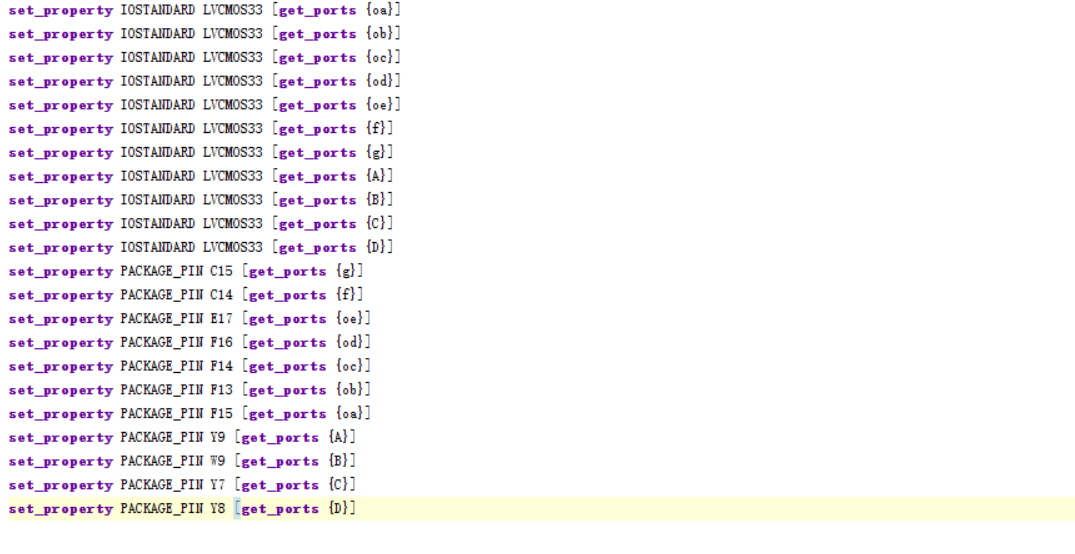
*80-90ns, {A,B,C,D}=1000, {a,b,c,d,e,f,g}=00000000, which display 8 in 7-seg tube;*

*90-100ns, {A,B,C,D}=1001, {a,b,c,d,e,f,g}=0000100, which display 9 in 7-seg tube;*

*100-160ns, {a,b,c,d,e,f,g} = 1111111, which display nothing in 7-seg tube.*

##### Constraint file and the testing

*Constraint File*



*Analysis*

**

*Testcase #1*

*When input is 0000,*

*It will display 0*

0 0 0 0

**

*Testcase #2*

*When input is 0001,*

*It will display 1*

0 0 0 1

**

*Testcase #3*

*When input is 0010,*

*It will display 2*

0 0 1 0

**

*Testcase #4*

*When input is 0011,*

*It will display 3*

0 0 1 1

**

*Testcase #5*

*When input is 0100,*

*It will display 4*

0 1 0 0

**

*Testcase #6*

*When input is 0101,*

*It will display 5*

0 1 0 1

**

*Testcase #7*

*When input is 0110,*

*It will display 6*

0 1 1 0

**

*Testcase #8*

*When input is 0111,*

*It will display 7*

0 1 1 1

**

*Testcase #9*

*When input is 1000,*

*It will display 8*

1 0 0 0

**

*Testcase #10*

*When input is 1001,*

*It will display 9*

1 0 0 1

**

*Testcase #11*

*When input is 1010,*

*It will display nothing*

1 0 1 0

**

*Testcase #12*

*When input is 1111,*

*It will display nothing*

1 1 1 1

##### the description of operation

*No Problem*