DIGITAL DESIGN

LAB6 COMBINATORIAL CIRCUIT: ENCODER, DECODER, MULTIPLEXER, DEMULTIPLEXER

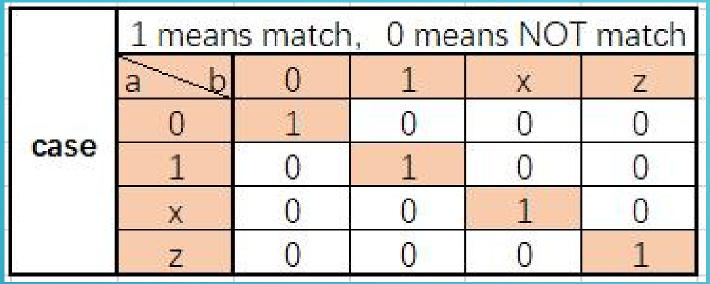
2022 FALL TERM @ CSE . SUSETCH

LAB6

- Verilog
  - behaviora description: case, casex, casez
- Combinational circuit
  - Encoder
  - Decoder
  - Multiplexer
  - Demultiplexer
- Practise

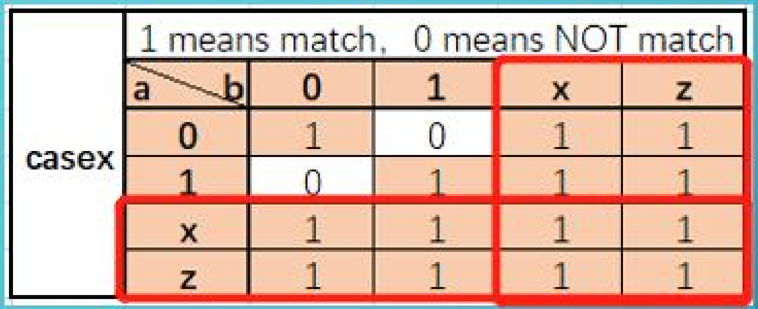
CASE VS CASEZ VS CASEX

- For example : using case, casez, casex to match ‘a’ and ‘b’



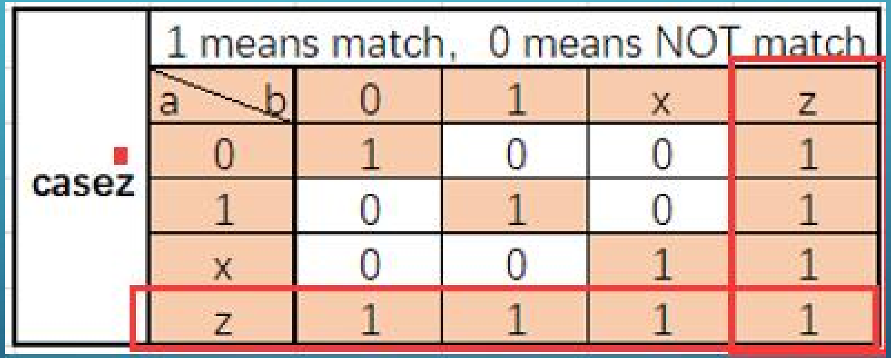
case

1 means match, 0 means NOT match				
a \ b	0	1	x	z
0	1	0	0	0
1	0	1	0	0
x	0	0	1	0
z	0	0	0	1



casex

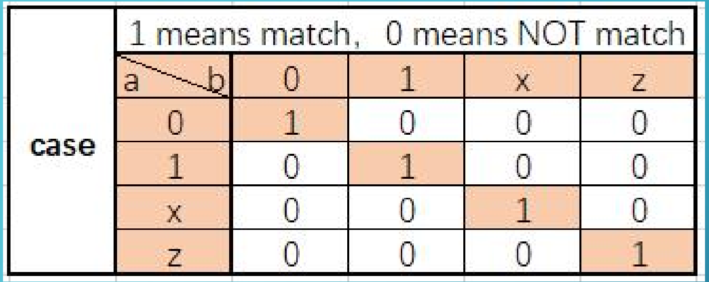
1 means match, 0 means NOT match				
a \ b	0	1	x	z
0	1	0	1	1
1	0	1	1	1
x	1	1	1	1
z	1	1	1	1



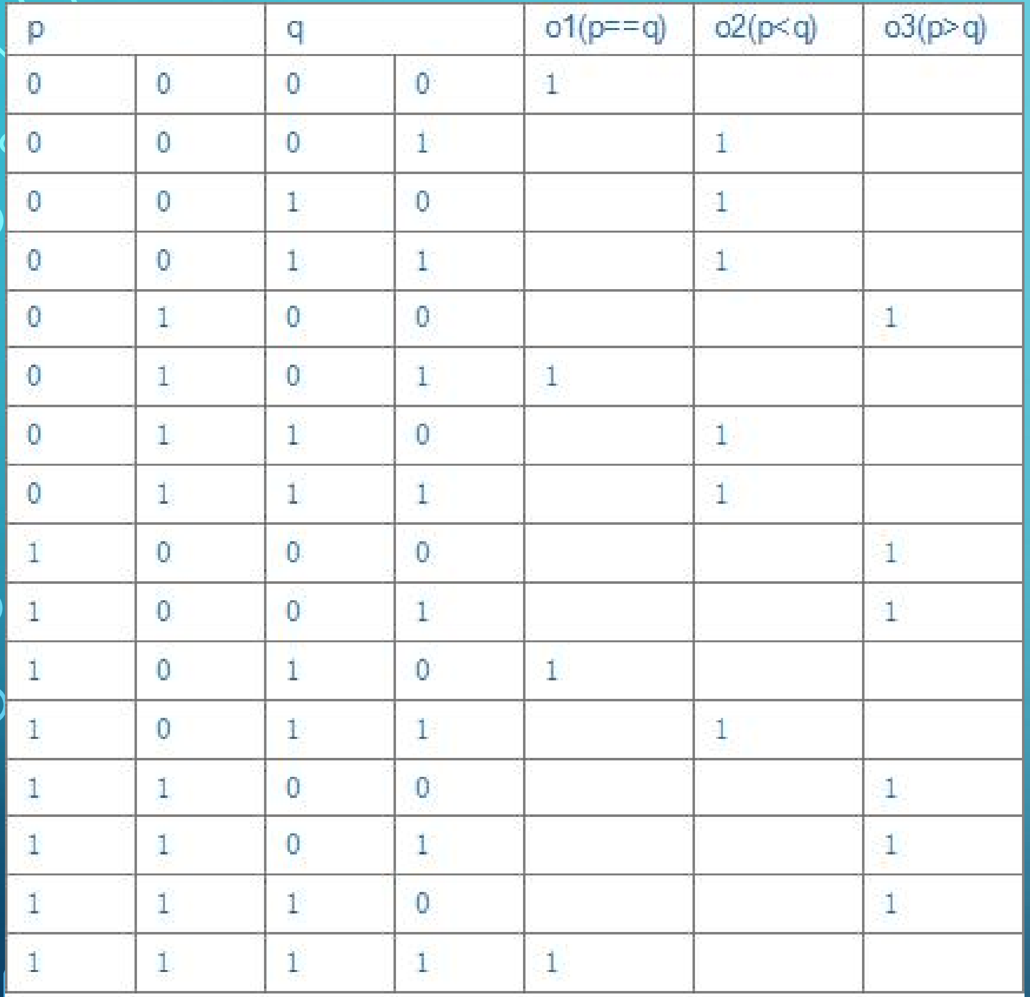
casez

1 means match, 0 means NOT match				
a \ b	0	1	x	z
0	1	0	0	1
1	0	1	0	1
x	0	0	1	1
z	1	1	1	1

BEHAVIORAL MODELING(1) CASE

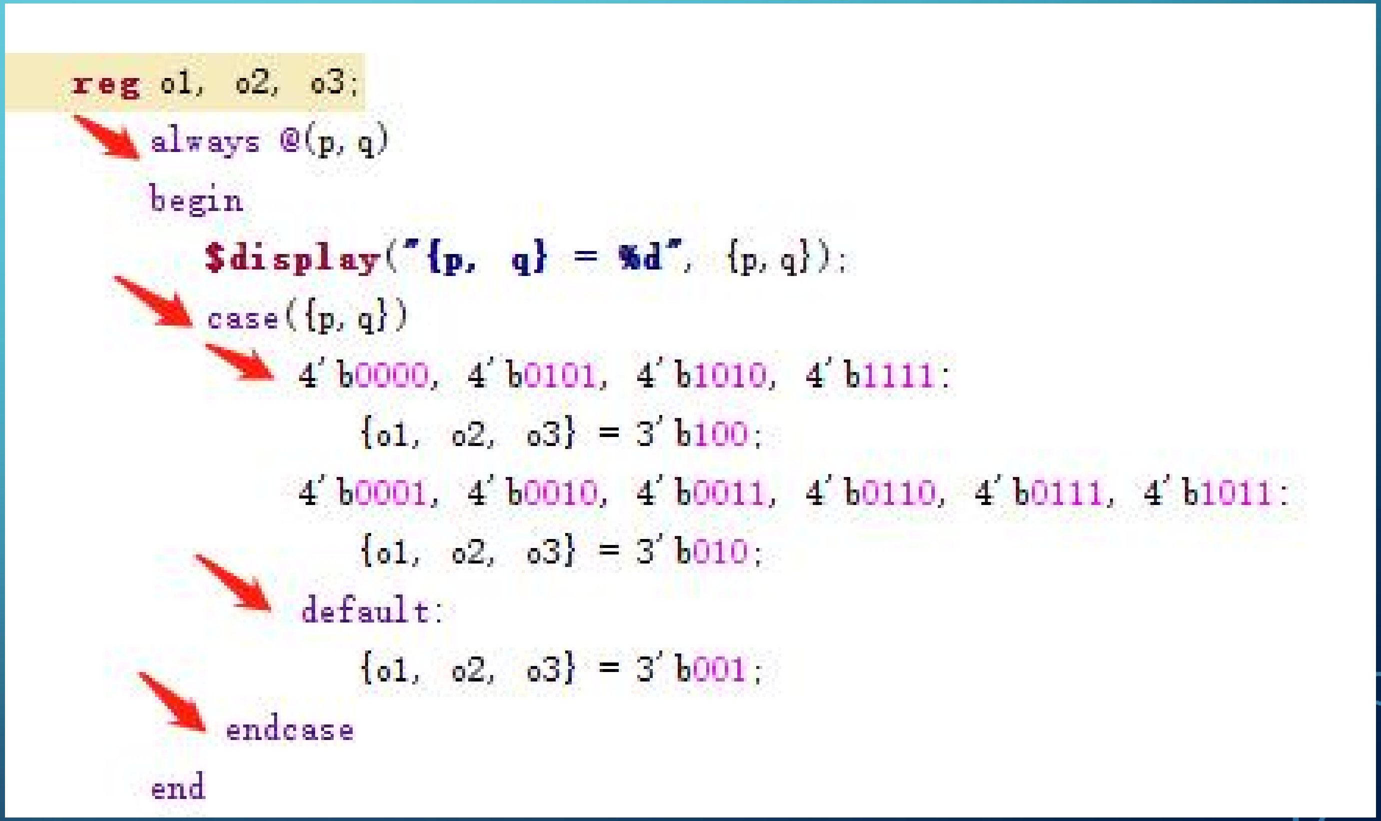


case	1 means match, 0 means NOT match				
	a \ b	0	1	x	z
	0	1	0	0	0
	1	0	1	0	0
	x	0	0	1	0
	z	0	0	0	1



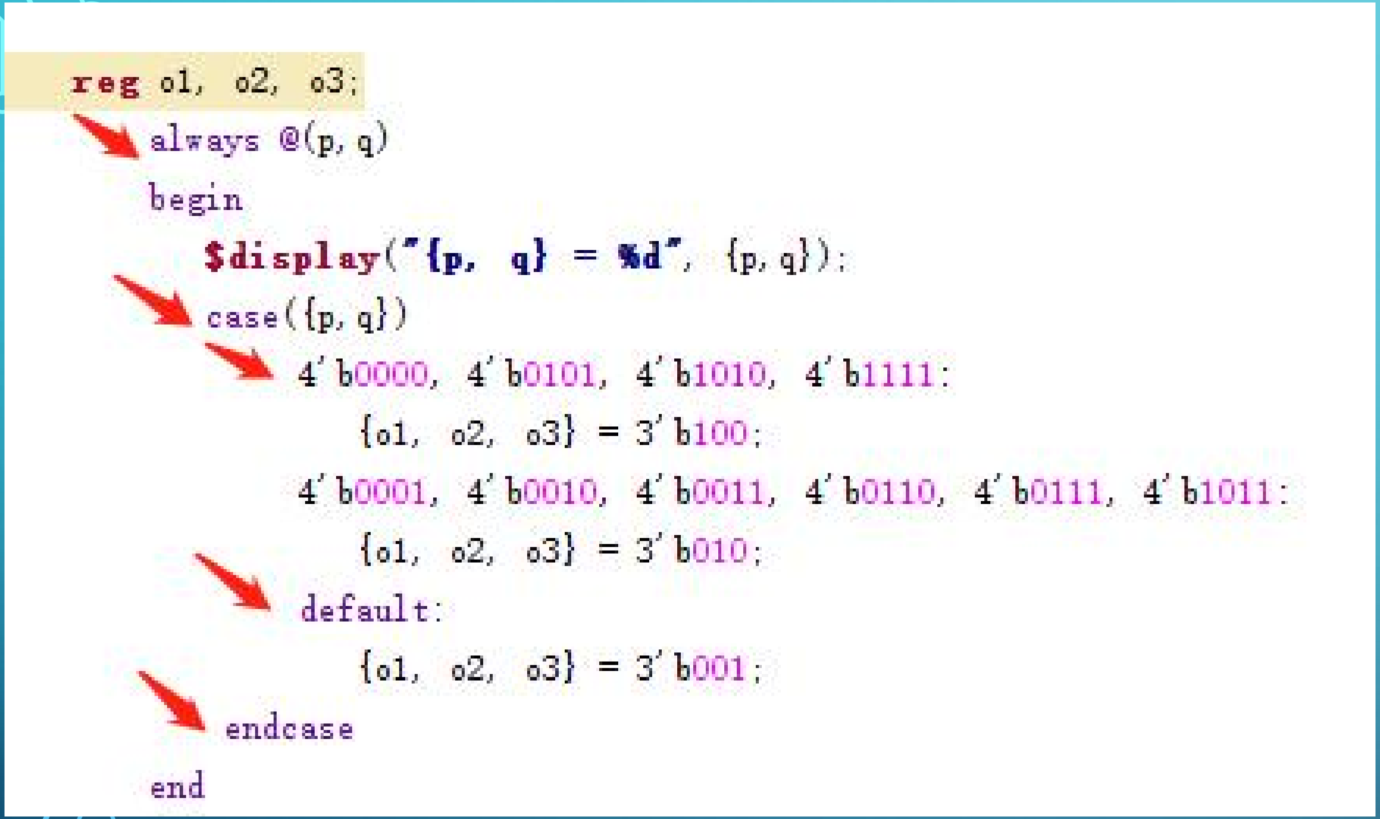
p		q		o1(p==q)	o2(p<q)	o3(p>q)
0	0	0	0	1		
0	0	0	1		1	
0	0	1	0		1	
0	0	1	1		1	
0	1	0	0			1
0	1	0	1	1		
0	1	1	0		1	
0	1	1	1		1	
1	0	0	0			1
1	0	0	1			1
1	0	1	0	1		
1	0	1	1		1	
1	1	0	0			1
1	1	0	1			1
1	1	1	0			1
1	1	1	1	1		

truth table for 2-bit comparator

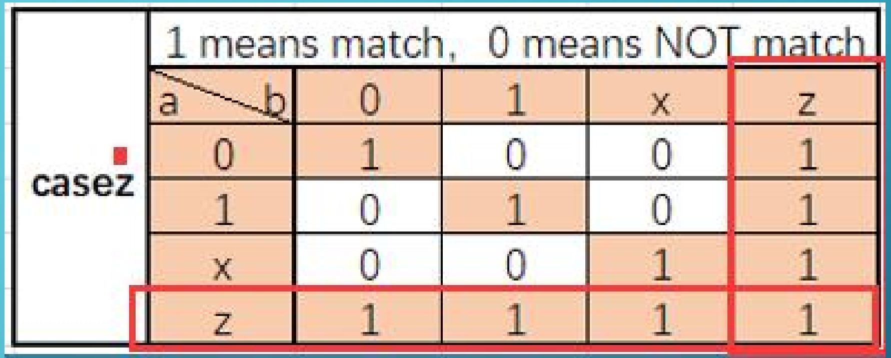


```
reg o1, o2, o3;
  always @(p, q)
  begin
      $display("{p, q} = %d", {p, q});
      case({p, q})
          4'b0000, 4'b0101, 4'b1010, 4'b1111:
              {o1, o2, o3} = 3'b100;
          4'b0001, 4'b0010, 4'b0011, 4'b0110, 4'b0111, 4'b1011:
              {o1, o2, o3} = 3'b010;
          default:
              {o1, o2, o3} = 3'b001;
       endcase
  end
```

BEHAVIORAL MODELING(1) CASZ



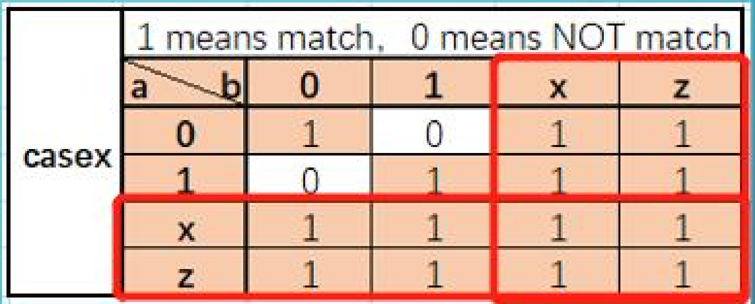
```
reg o1, o2, o3;
always @(p, q)
begin
    $display("{p, q} = %d", {p,q});
    case({p,q})
        4'b0000, 4'b0101, 4'b1010, 4'b1111:
            {o1, o2, o3} = 3'b100;
        4'b0001, 4'b0010, 4'b0011, 4'b0110, 4'b0111, 4'b1011:
            {o1, o2, o3} = 3'b010;
        default:
            {o1, o2, o3} = 3'b001;
    endcase
end
```



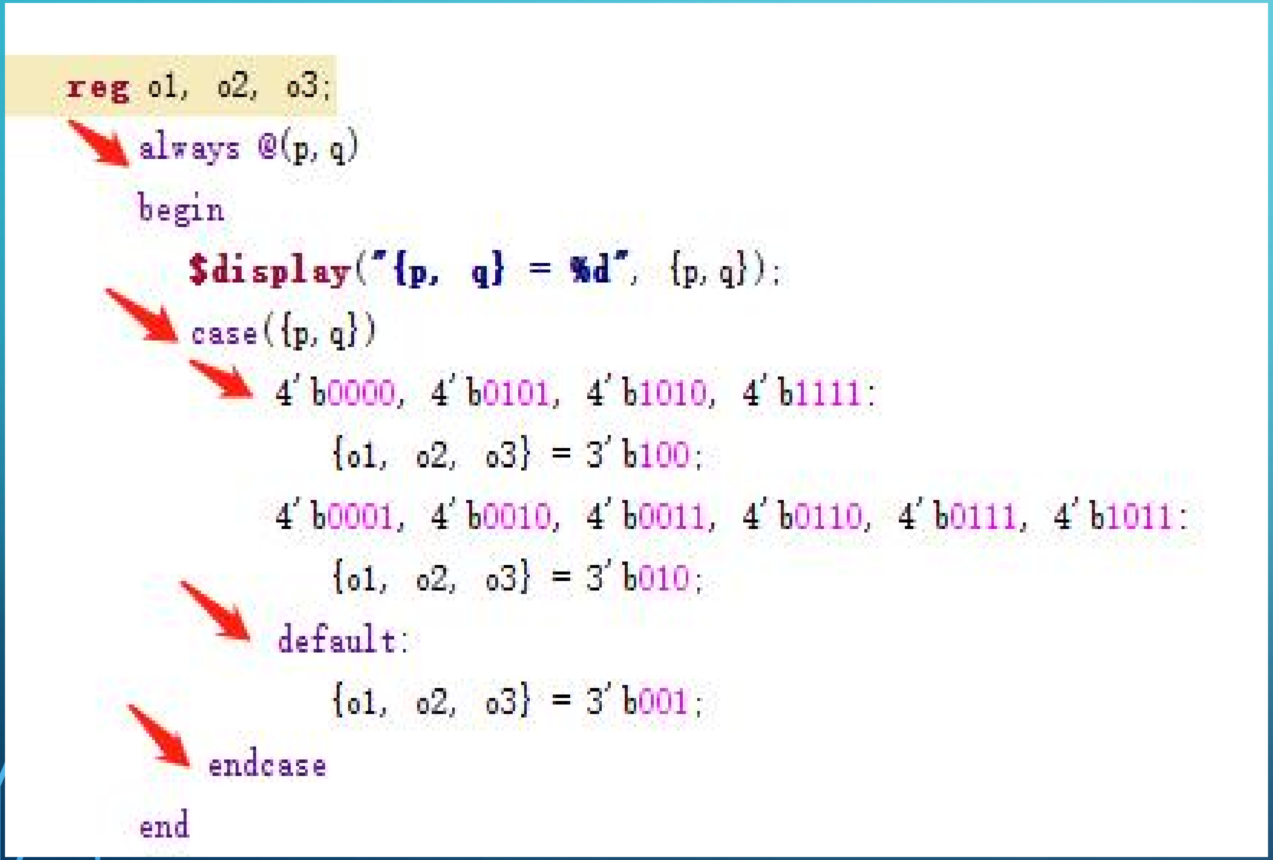
	1 means match, 0 means NOT match				
casez	a \ b	0	1	x	z
	0	1	0	0	1
	1	0	1	0	1
	x	0	0	1	1
	z	1	1	1	1

```
casez({p,q})
      4'b0000,4'b0101,4'b1010,4'b1111:
            {o1, o2, o3} = 3'b100;
       4'b0001, 4'b001z, 4'b011z, 4'b1011:
             {o1, o2, o3} = 3'b010;
       defalut:   {o1, o2, o3} = 3'b001;
endcase
```

BEHAVIORAL MODELING(1) CASEX



casex	1 means match, 0 means NOT match				
	a \ b	0	1	x	z
	0	1	0	1	1
	1	0	1	1	1
	x	1	1	1	1
	z	1	1	1	1



```
reg o1, o2, o3;
always @(p, q)
begin
  $display("{p, q} = %d", {p, q});
  case({p, q})
      4'b0000, 4'b0101, 4'b1010, 4'b1111:
          {o1, o2, o3} = 3'b100;
      4'b0001, 4'b0010, 4'b0011, 4'b0110, 4'b0111, 4'b1011:
          {o1, o2, o3} = 3'b010;
      default:
          {o1, o2, o3} = 3'b001;
  endcase
end
```

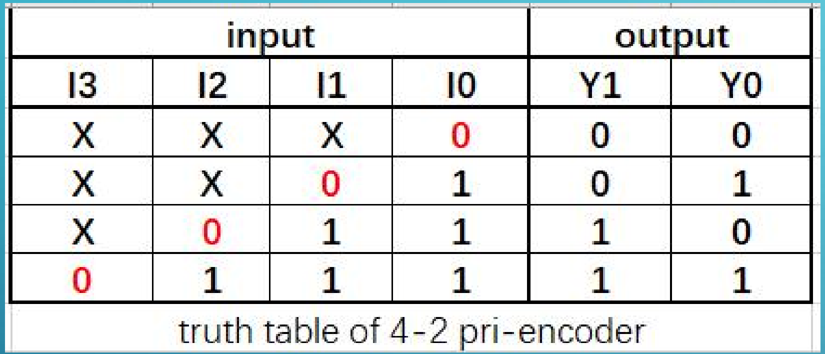
```
casex({p,q})
        4'b0000,4'b0101,4'b1010,4'b1111:
                {o1, o2, o3} = 3'b100;
         4'b0001, 4'b001x, 4'b011x, 4'b1011:
                {o1, o2, o3} = 3'b010;
         defalut:   {o1, o2, o3} = 3'b001;
endcase
```

```
casex({p,q})
        4'b0000,4'b0101,4'b1010,4'b1111:
                {o1, o2, o3} = 3'b100;
         4'b0001, 4'b001z, 4'b011z, 4'b1011:
                {o1, o2, o3} = 3'b010;
         defalut:   {o1, o2, o3} = 3'b001;
endcase
```

ENCODER

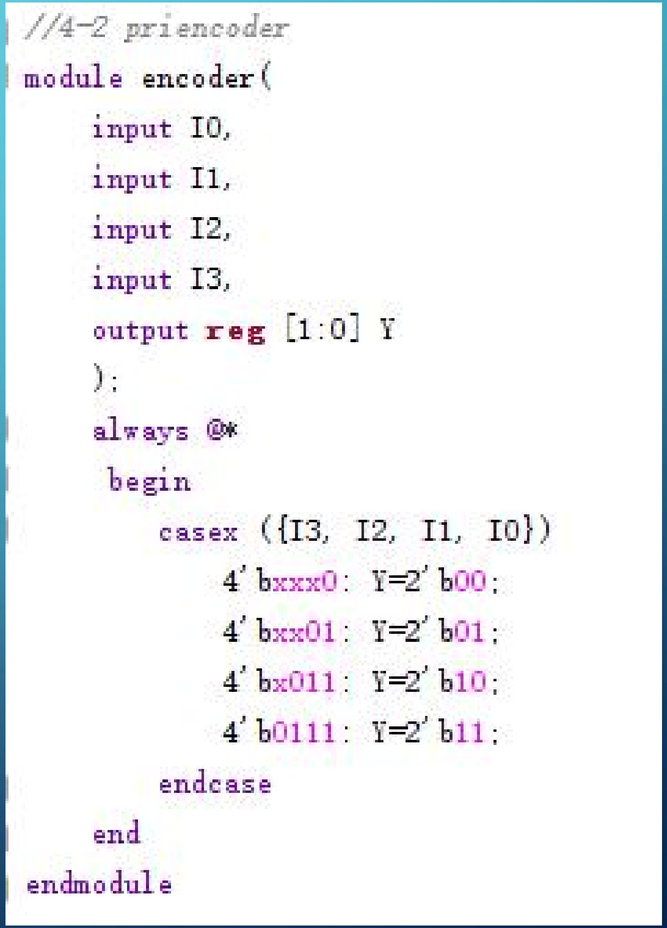
An encoder is a device that converts information from one format or code to another, for the purposes of **standardization**, **speed** or **compression**.

Priority encoder LSB's priority is the highest



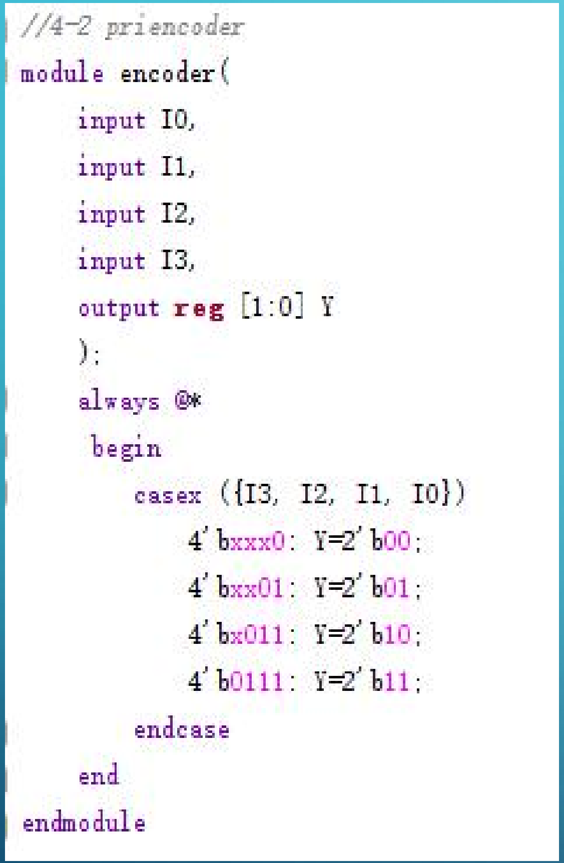
input				output	
I3	I2	I1	I0	Y1	Y0
X	X	X	0	0	0
X	X	0	1	0	1
X	0	1	1	1	0
0	1	1	1	1	1

truth table of 4-2 pri-encoder

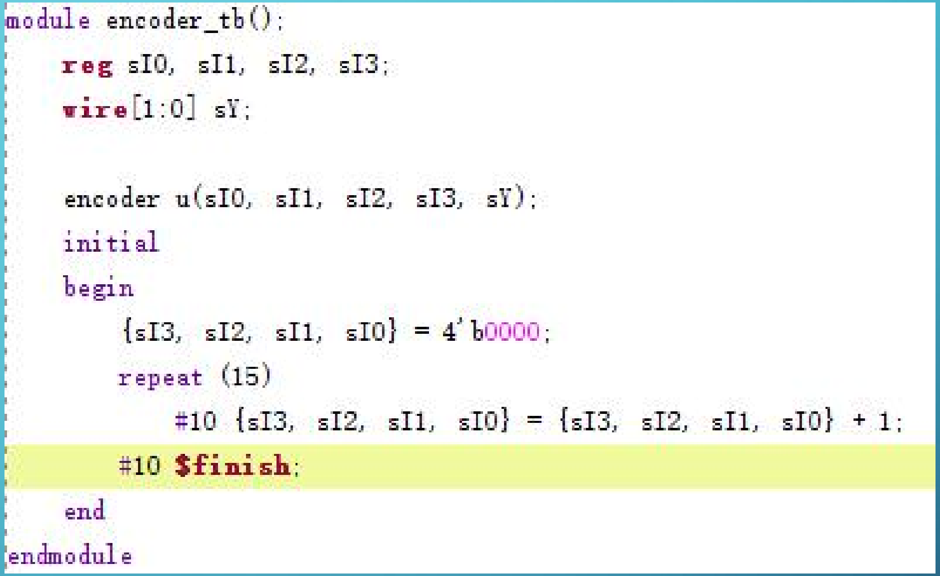


```
//4-2 priencoder
module encoder(
    input I0,
    input I1,
    input I2,
    input I3,
    output reg [1:0] Y
    );
    always @*
     begin
       casex ({I3, I2, I1, I0})
          4'bxxx0: Y=2'b00;
          4'bxx01: Y=2'b01;
          4'bx011: Y=2'b10;
          4'b0111: Y=2'b11;
       endcase
    end
endmodule
```

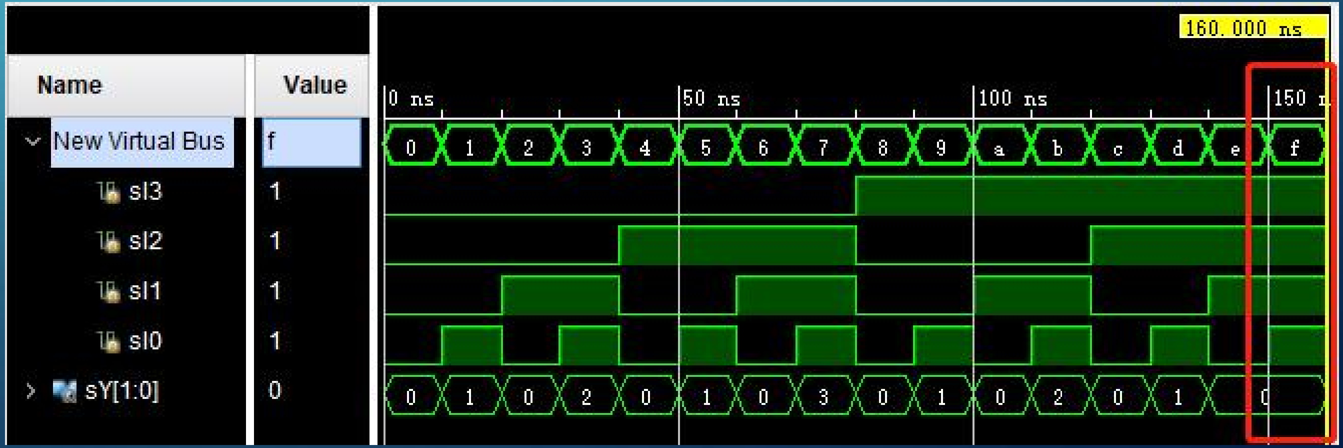
ENCODER(PRIORITY ENCODER)



```
//4-2 priencoder
module encoder(
    input I0,
    input I1,
    input I2,
    input I3,
    output reg [1:0] Y
    );
    always @*
     begin
        casex ({I3, I2, I1, I0})
            4'bxxx0: Y=2'b00;
            4'bxx01: Y=2'b01;
            4'bx011: Y=2'b10;
            4'b0111: Y=2'b11;
        endcase
    end
endmodule
```

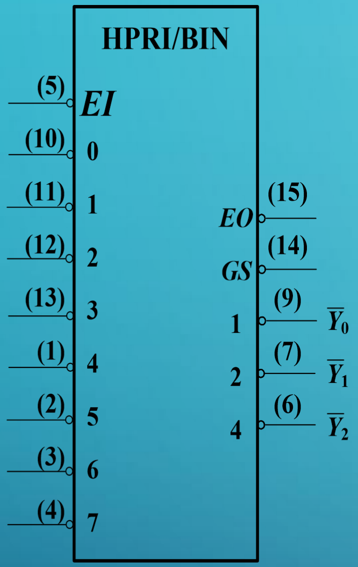


```
module encoder_tb();
    reg sI0, sI1, sI2, sI3;
    wire[1:0] sY;

    encoder u(sI0, sI1, sI2, sI3, sY);
    initial
    begin
        {sI3, sI2, sI1, sI0} = 4'b0000;
        repeat (15)
            #10 {sI3, sI2, sI1, sI0} = {sI3, sI2, sI1, sI0} + 1;
        #10 $finish;
    end
endmodule
```

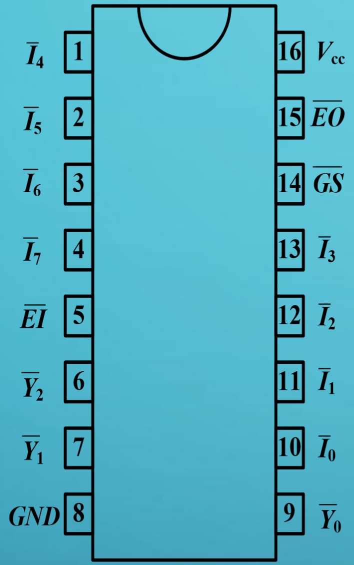


Name	Value
New Virtual Bus	f
sI3	1
sI2	1
sI1	1
sI0	1
sY[1:0]	0

ENCODER(74148)



Logic diagram



Pin diagram

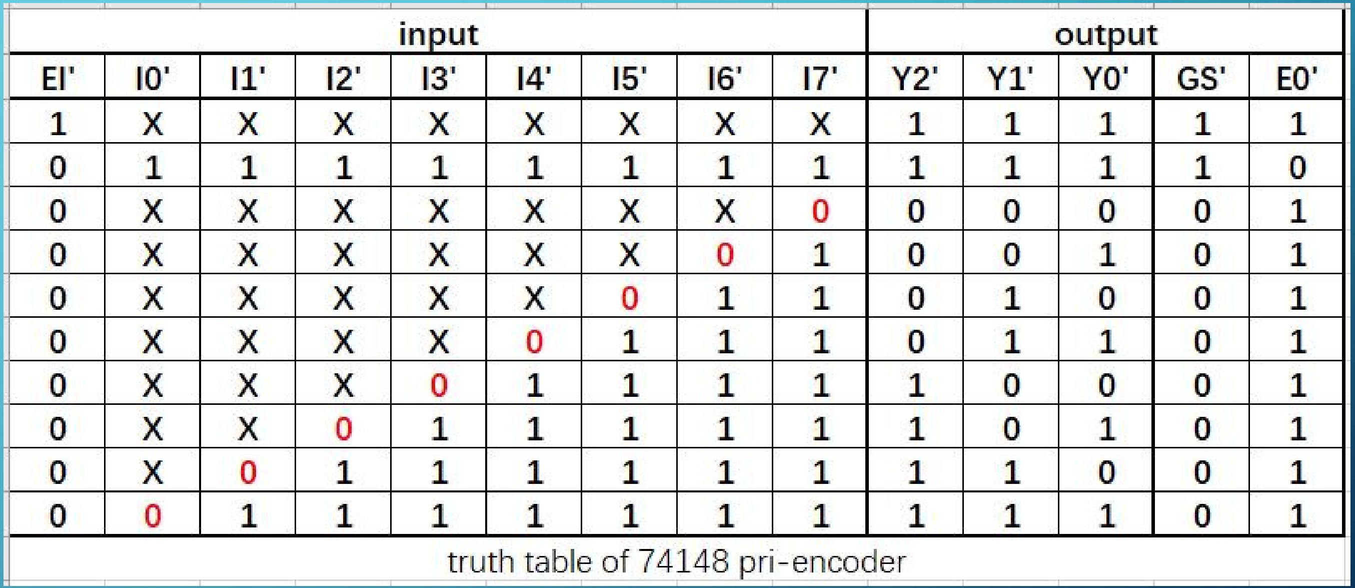
- **74148**: 8-3 priority encoder
- The input is low level effective, and the output is 3 bit one's complement.
- HPRI illustrates that the MSB's priority is the highest

ENCODER(74148)

- **EI**: Enable input
- **EO**: Enable output
- **GS:** Group select

$$\overline{\mathrm{EO}} = \overline{\mathrm{EI}\ \overline{I_0}\,\overline{I_1}\,\overline{I_2}\,\overline{I_3}\,\overline{I_4}\,\overline{I_5}\,\overline{I_6}\,\overline{I_7}}$$

$$\overline{\mathrm{GS}} = \overline{\mathrm{EI}(I_0 + I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7)}$$



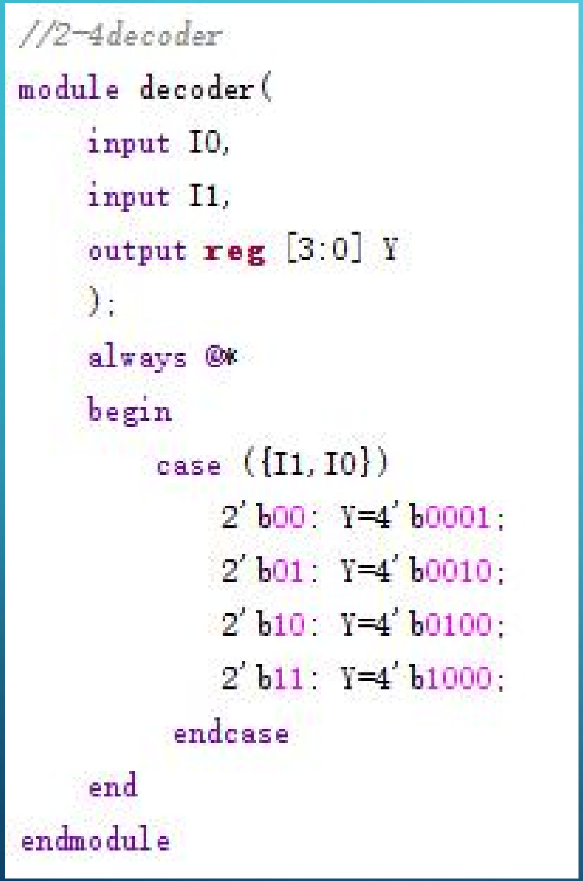
input									output				
EI'	I0'	I1'	I2'	I3'	I4'	I5'	I6'	I7'	Y2'	Y1'	Y0'	GS'	E0'
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

truth table of 74148 pri-encoder

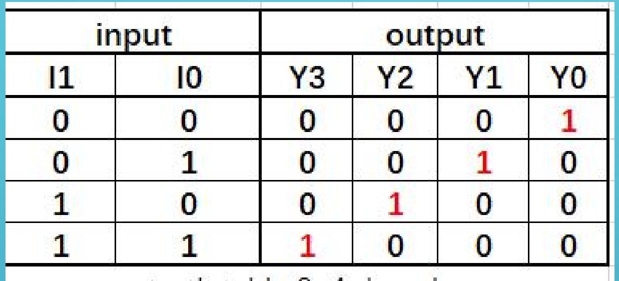
DECODER

- In digital electronics, a **binary Decoder** is a combinational logic circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs. They are used in a wide variety of applications, including data du-multiplexing, seven segment displays, and memory address decoding.
- There are several types of binary decoders, but in all cases a decoder is an electronic circuit with multiple input and multiple output signals, which converts every unique combination of input states to a specific combination of output states.
- In addition to integer data inputs, some decoders also have one or more "enable" inputs. When the enable input is negated (disabled), all decoder outputs are forced to their inactive states.

DECODER （2-4 DECODER）

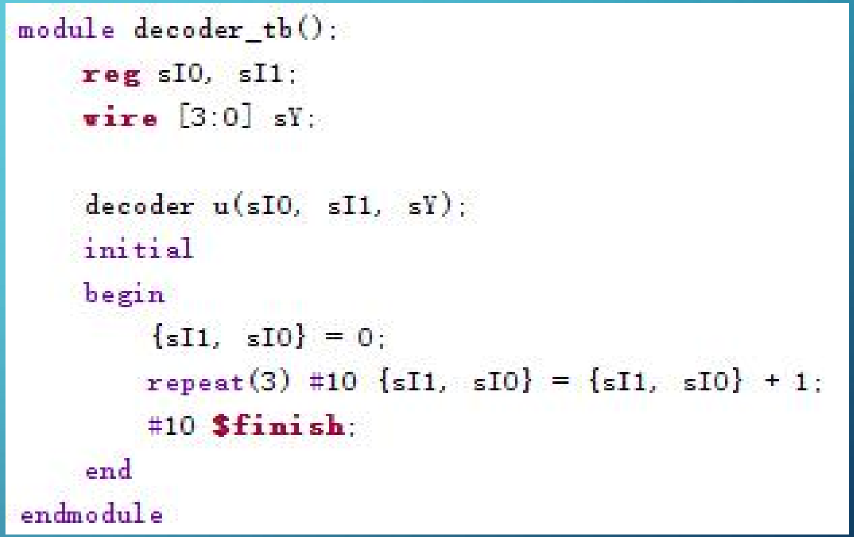


```
//2-4decoder
module decoder(
    input I0,
    input I1,
    output reg [3:0] Y
    );
    always @*
    begin
        case ({I1,I0})
            2'b00: Y=4'b0001;
            2'b01: Y=4'b0010;
            2'b10: Y=4'b0100;
            2'b11: Y=4'b1000;
        endcase
    end
endmodule
```

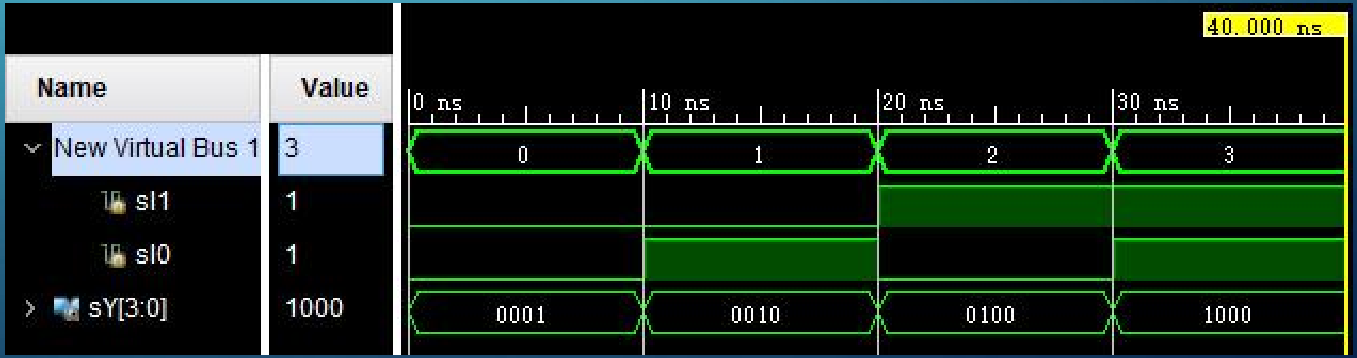


input		output			
I1	I0	Y3	Y2	Y1	Y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

truth table 2-4 decoder



```
module decoder_tb();
    reg sI0, sI1;
    wire [3:0] sY;

    decoder u(sI0, sI1, sY);
    initial
    begin
        {sI1, sI0} = 0;
        repeat(3) #10 {sI1, sI0} = {sI1, sI0} + 1;
        #10 $finish;
    end
endmodule
```



```
Name                Value
New Virtual Bus 1   3
    sI1             1
    sI0             1
sY[3:0]             1000
```

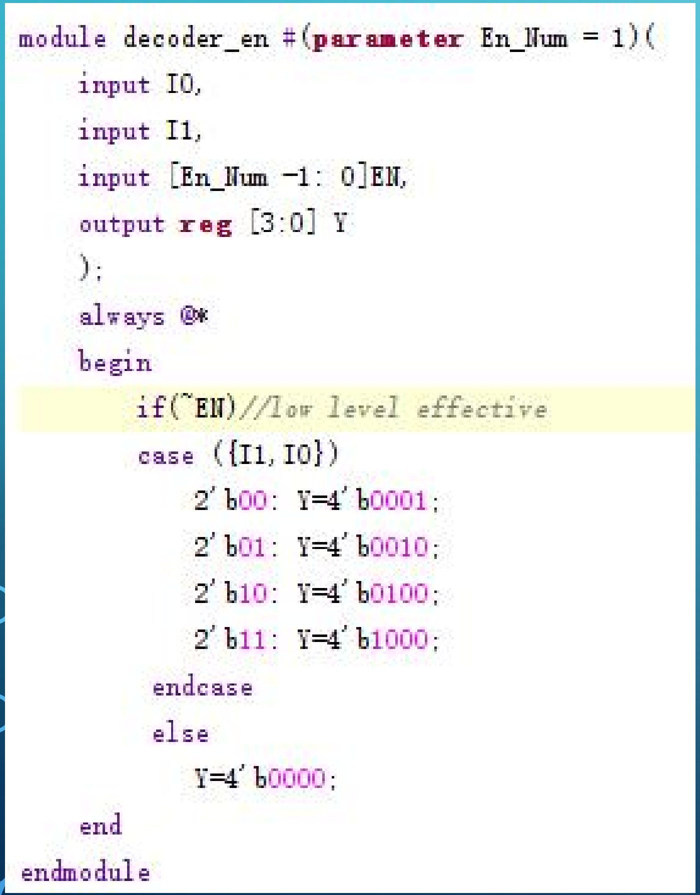
ONE HOT CODING

- **One hot coding**, also known as one bit effective coding
  - use n-bit status register to code n states.
  - Each state has its own register bits, and at any time, only one of them is valid.

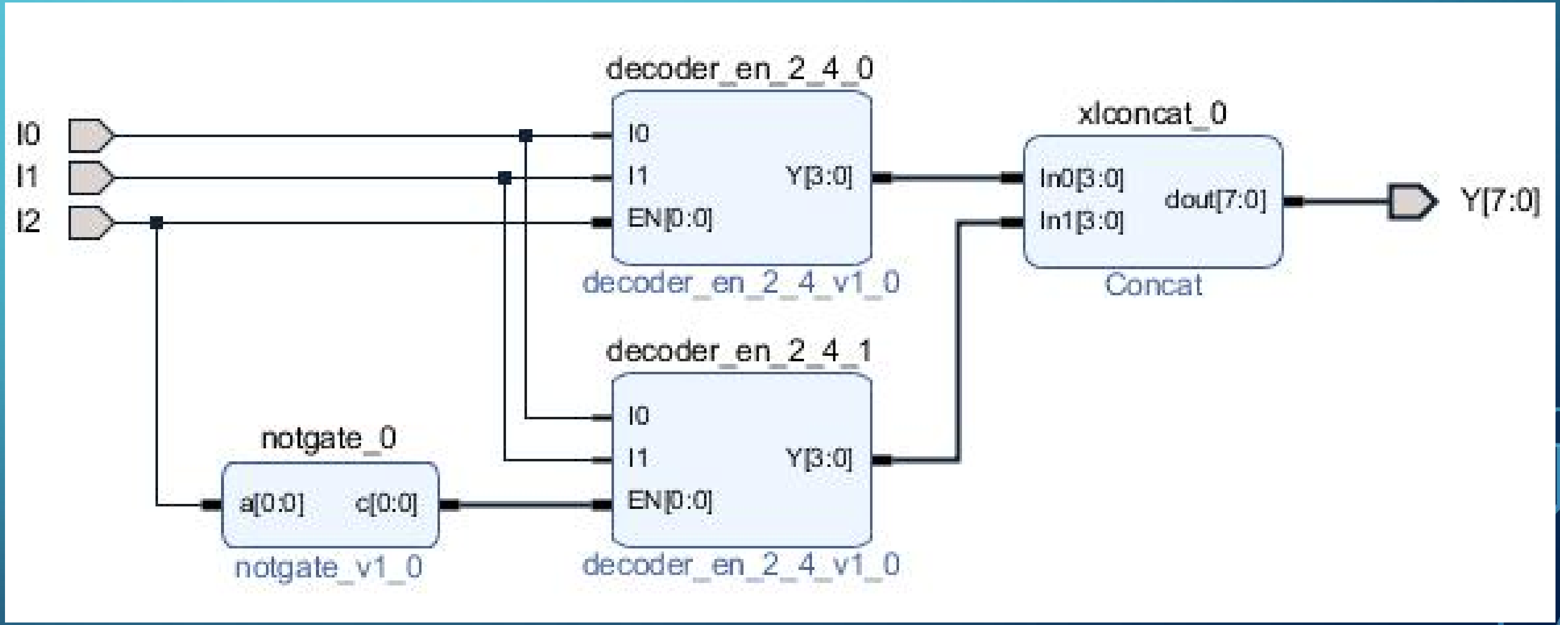
DECODER （3-8 DECODER）

Enable input port

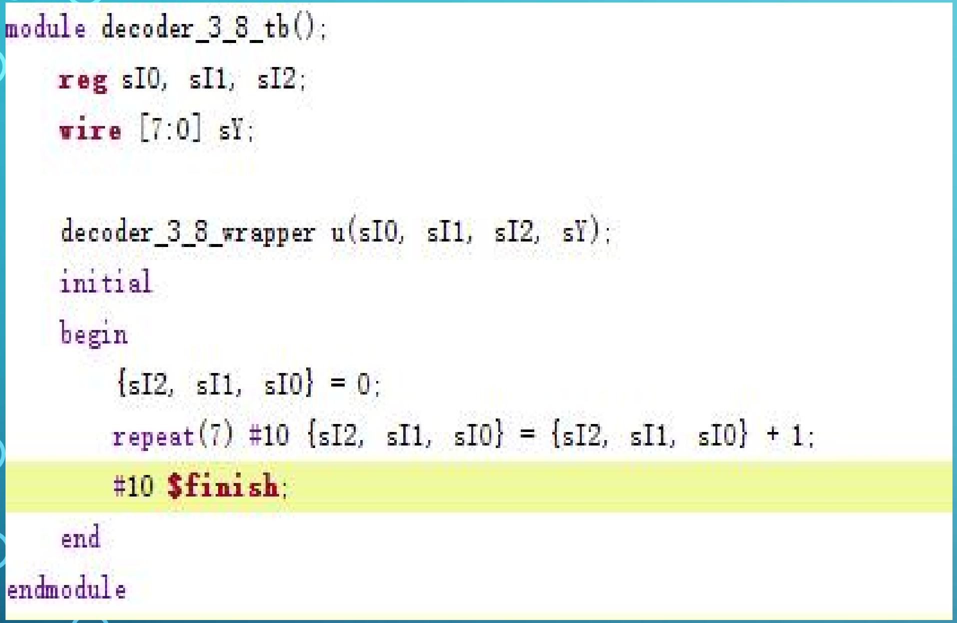
- How to implement an 3-8 decoder by using two 2-4 decoders?



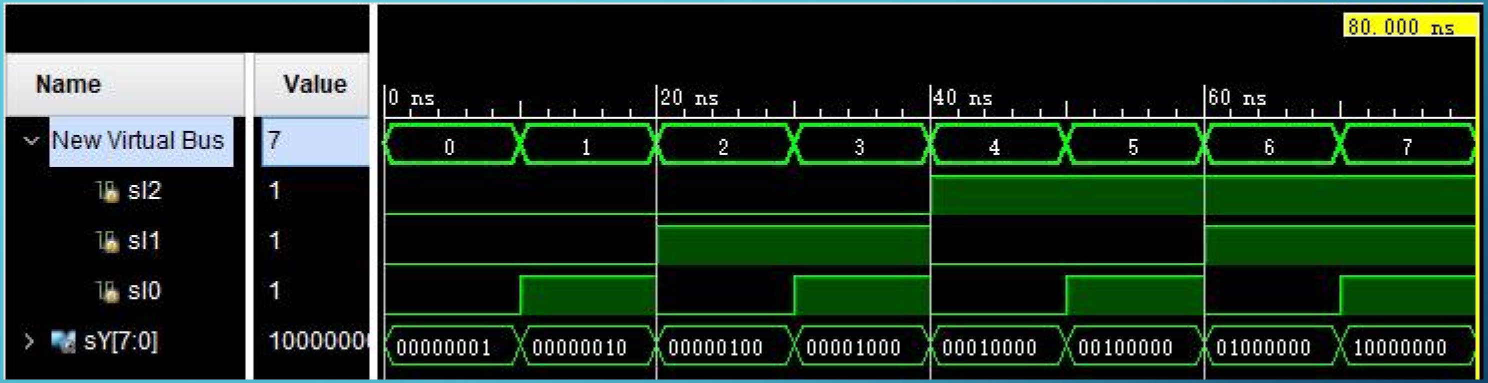
```
module decoder_en #(parameter En_Num = 1)(
    input I0,
    input I1,
    input [En_Num -1: 0]EN,
    output reg [3:0] Y
    );
    always @*
    begin
        if(~EN)//low level effective
        case ({I1,I0})
            2'b00: Y=4'b0001;
            2'b01: Y=4'b0010;
            2'b10: Y=4'b0100;
            2'b11: Y=4'b1000;
         endcase
         else
            Y=4'b0000;
    end
endmodule
```



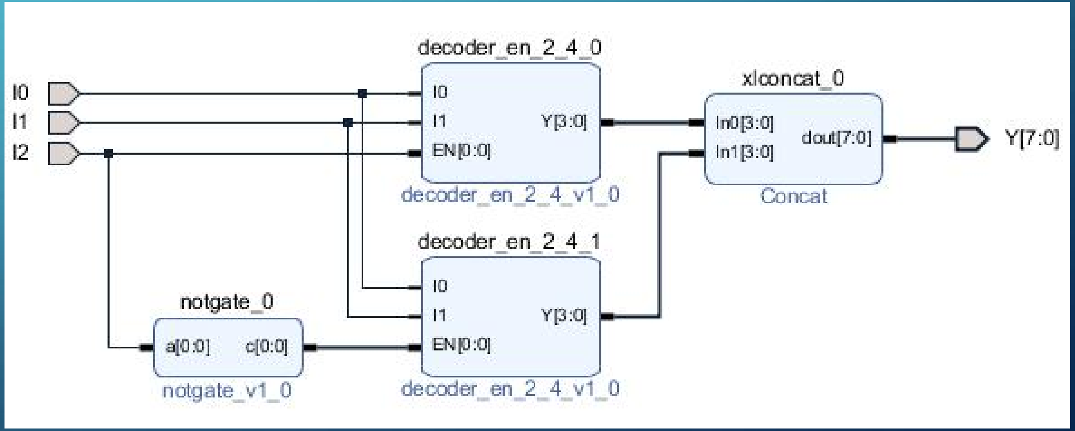
DECODER （3-8 DECODER）



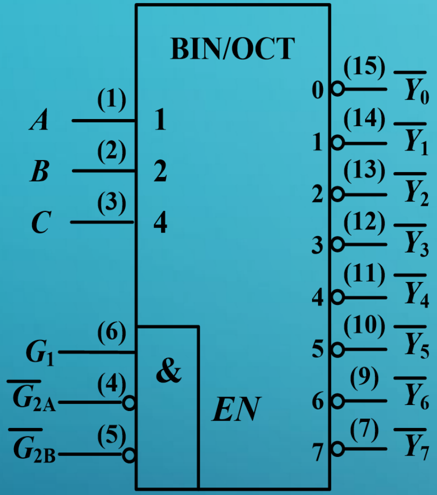
```
module decoder_3_8_tb();
    reg sI0, sI1, sI2;
    wire [7:0] sY;

    decoder_3_8_wrapper u(sI0, sI1, sI2, sY);
    initial
    begin
        {sI2, sI1, sI0} = 0;
        repeat(7) #10 {sI2, sI1, sI0} = {sI2, sI1, sI0} + 1;
        #10 $finish;
    end
endmodule
```



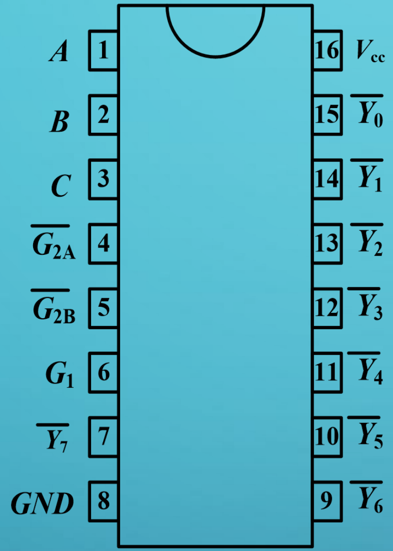
```
Name               Value      0 ns        20 ns        40 ns        60 ns      80.000 ns
New Virtual Bus    7          0     1     2     3      4     5      6     7
  sI2              1
  sI1              1
  sI0              1
sY[7:0]            10000000   00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000
```



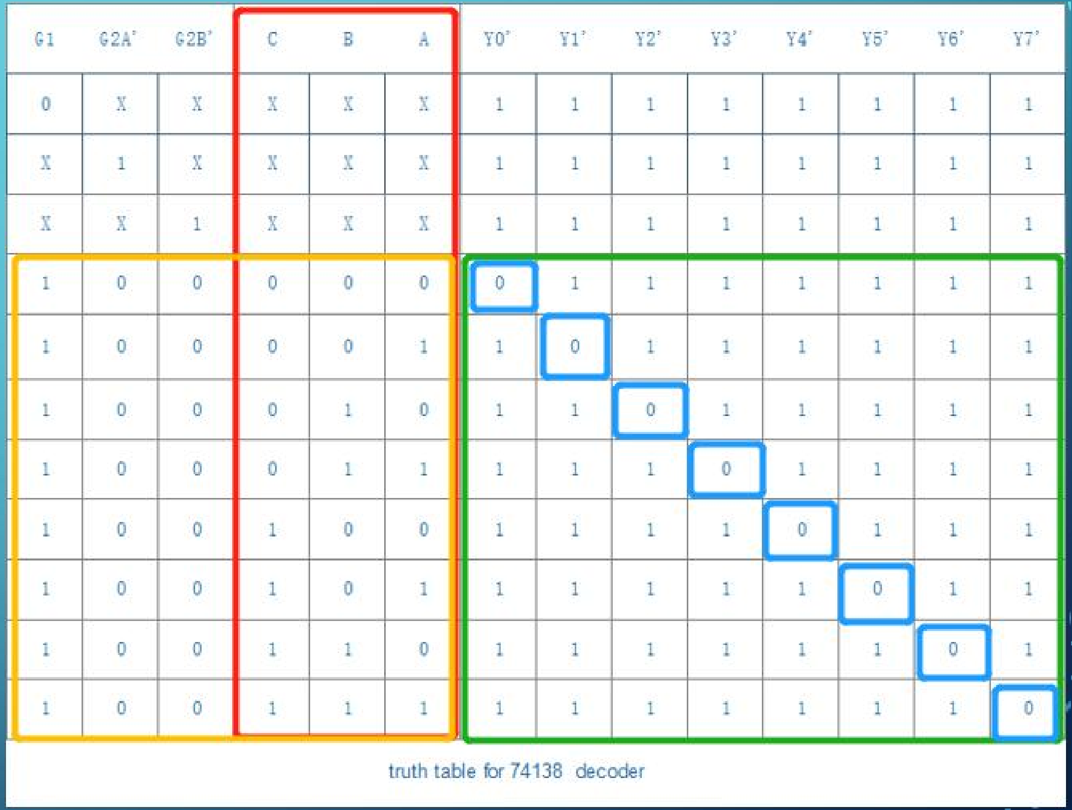
DECODER （74138）



Logic diagram



Pin diagram



G1	G2A'	G2B'	C	B	A	Y0'	Y1'	Y2'	Y3'	Y4'	Y5'	Y6'	Y7'
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

truth table for 74138 decoder

PRACTICES(1)

1. Design a 4-2 Programmable priority encoder in which the bit of input which has the highest priority is determined by another input signal, the priority is successively reduced from this bit to the right.

   1) ports:

      a. Input port X is the encoded object which is encoded to Y, Y is the output port;

      b. Another input port P which is used to indicate the index of the highest priority bit in X. for example: if the value of input which indicate the highest priority is 2, it means the priority bit from high to low is : 2 1 0 3

      Ps: in this circuit, X is 4-bit width, the index of LSB is 0, the index of MSB is 3.

2. Build a testbench, do the simulation and verify the function of your design.

PRACTICES(2)

- Implement a 4-16 decoder by two 3-8 decoders. You can either modify the provided 3-8 decoder or design 74138 decoder
  - Do the design and verify the function of your design.
  - Create the constraint file, do the synthetic and implementation, generate the bitstream file and program the device, then test on the develop board.