# DIGITAL DESIGN

LAB8 COMBINATORIAL CIRCUIT AND 7-SEG-TUBE(S)

2022 FALL TERM @ CSE . SUSTECH

# DIGITAL DESIGN

LAB8 COMBINATORIAL CIRCUIT AND 7-SEG-TUBE(S)

2022 FALL TERM @ CSE . SUSTECH

# LAB8

- Combinatorial circuit
  - 1bit full adder, 2bits full adder
  - Lighting 7 segment digital tube
- Practices

# 1BIT FULL ADDER VS MULTI-BITS FULL ADDER

Another input except a and b is the previous carry

```
module full_add_1b(a, b, cin, sum, cout);
input a, b, cin;
output sum, cout;
assign {cout, sum}=a+b+cin;
endmodule
```

```
module full_add_2b(a, b, cin, sum, cout);
input [1:0]a, b;
input cin;
output[1:0] sum;
output cout;

wire cout1;
full_add_1b u0(a[0], b[0], cin, sum[0], cout1);
full_add_1b u1(a[1], b[1], cout1, sum[1], cout);
endmodule
```

As the inputs are multi-bits, the output is calculated bit by bit

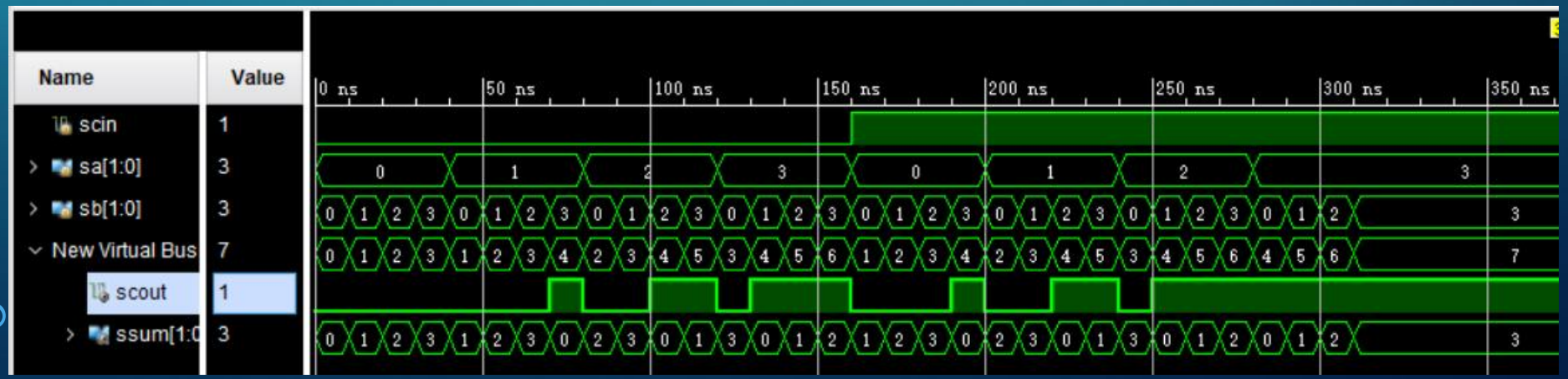
# 1BIT FULL ADDER VS MULTI-BITS FULL ADDER

```
module full_add_2b(a, b, cin, sum, cout);
input [1:0]a, b;
input cin;
output[1:0] sum;
output cout;

wire cout1;
full_add_1b u0(a[0], b[0], cin, sum[0], cout1);
full_add_1b u1(a[1], b[1], cout1, sum[1], cout);
endmodule
```

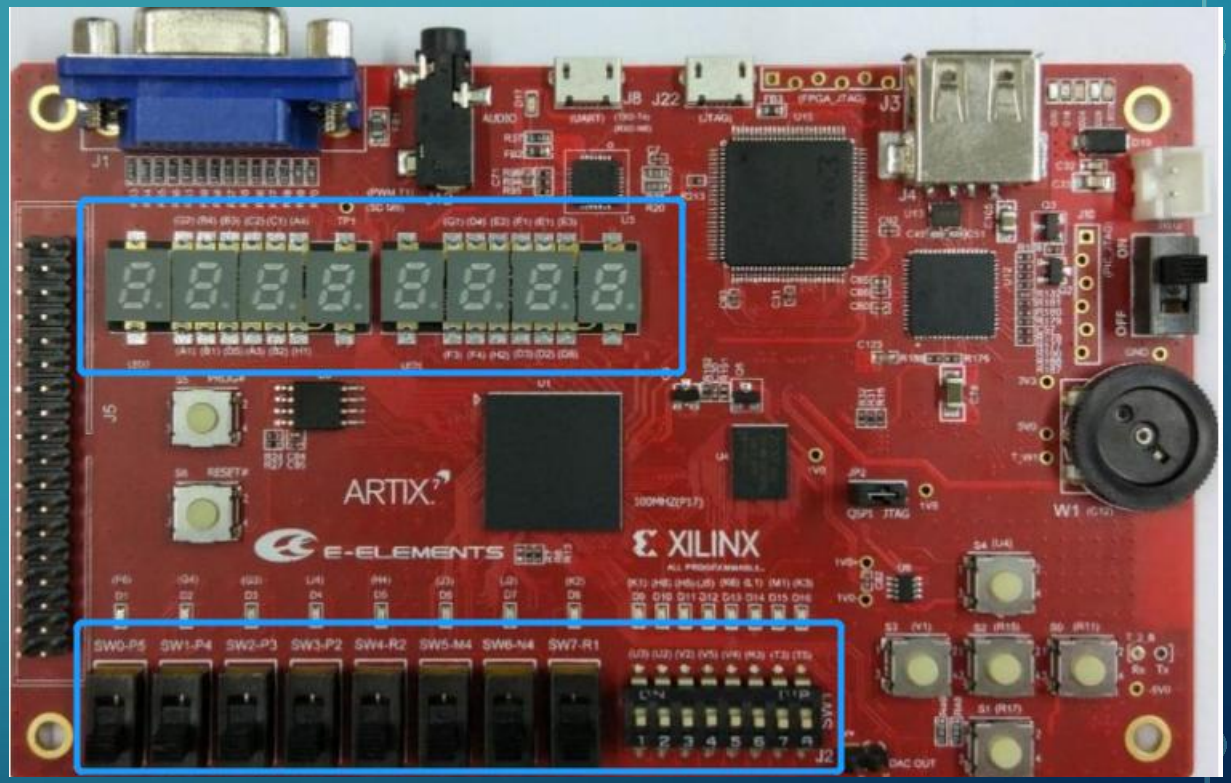
```
module full_add_sm(    );
reg scin;
reg [1:0]sa, sb;
wire [1:0]ssum;
wire scout;
full_add_2b u2(sa, sb, scin, ssum, scout);
initial
{scin, sa, sb} = 5'd0;

initial
begin
    repeat(31)
    #10 {scin, sa, sb} ={scin, sa, sb} +1;
end
endmodule
```

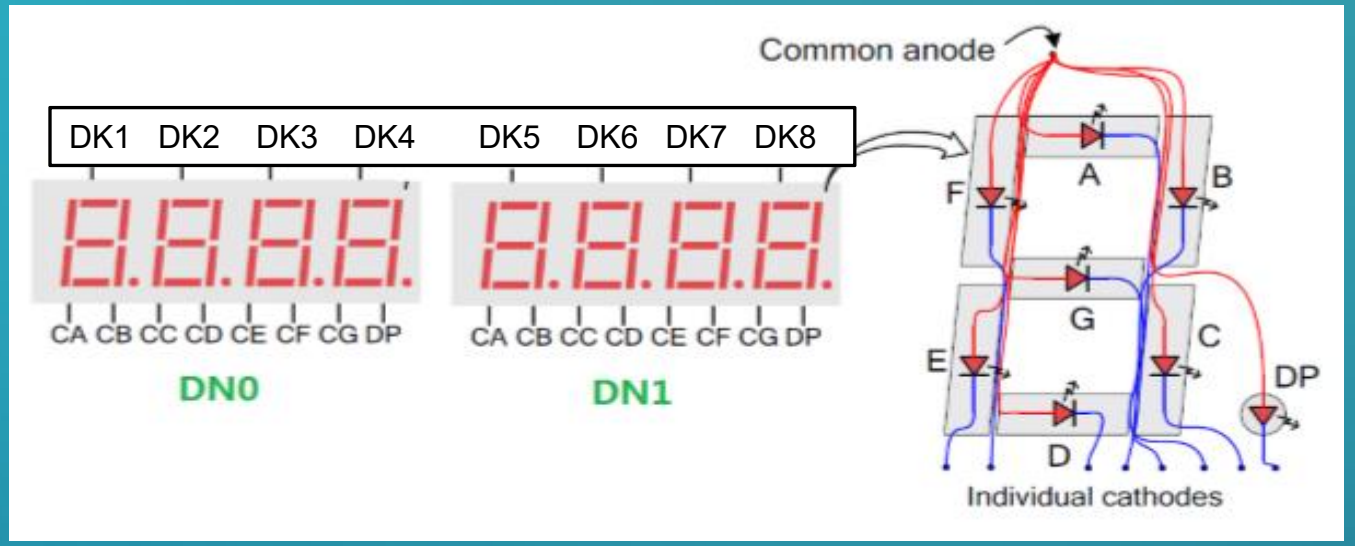


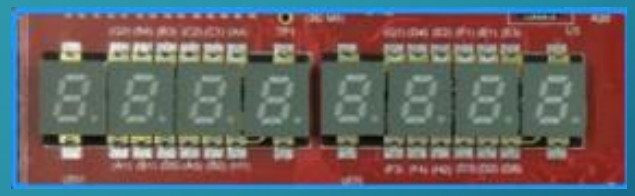
# LIGHTING THE 7-SEG-TUBE(S) ON EGO1

- Get an 4-bits width binary number from the input ports, show its hexadecimal number on the **7-seg-tube(s)**.
- Using **EGO1** develop board.
- 4*dial switch are the inputs while 7seg-tube(s) are the outputs



# LIGHTING THE 7-SEG-TUBE(S) ON EGO1

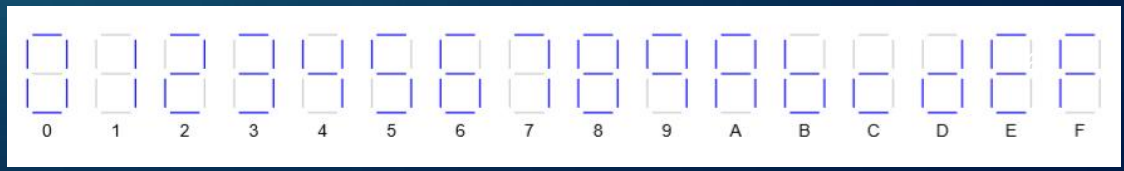


The ENABLE pin(8 in total) of each 7-seg-tube is different from the CONTROL pin(2*7 in total) of each set

While using the **EGO1 board**, follow the following rules:

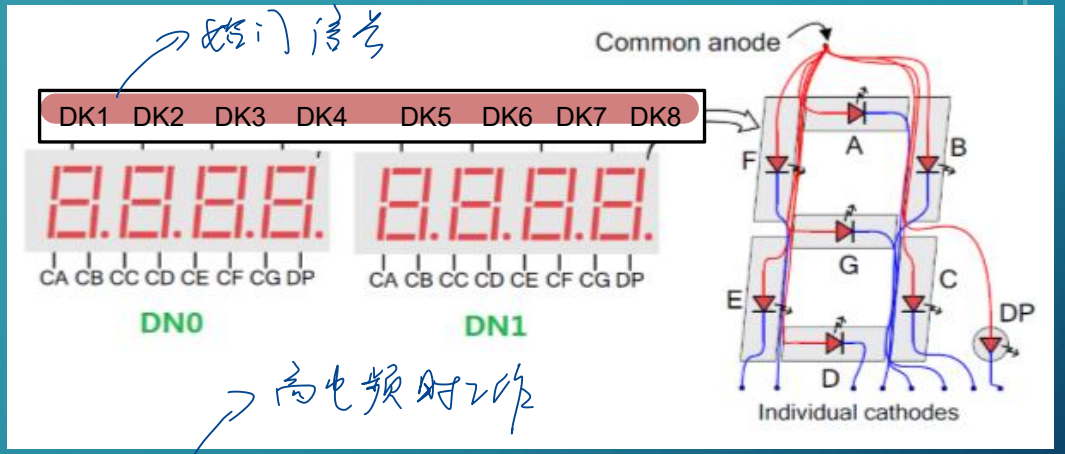
- Eight 7-seg-tubes, each one has an **enable pin**, all are **high level enable**(**1** means enable the tube while 0 means disable the tube)
- Eight 7-seg-tubes are divided into **2 groups: DN0 and DN1**.
- Every 4 7-seg-tubes in the same group share the same group of **control pins(CA ~CG and DP)**, which are **all high level enable**(1 means light the seg while 0 means NOT light the seg).



# LIGHTING THE 7-SEG-TUBE(S) ON EGO1

Practice1：
Complete the table with pin(s) according to the information printed on the EGO1 board.

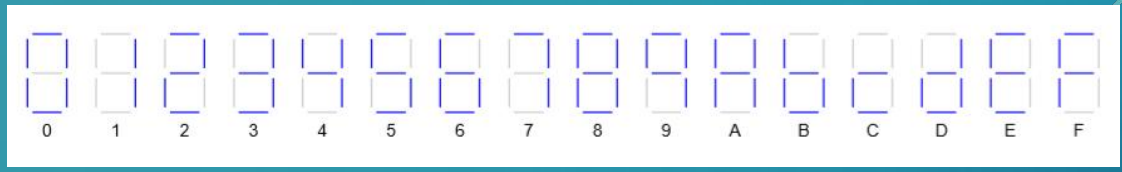
| GROUP | PORT | PIN | GROUP | PORT | PIN |
|---|---|---|---|---|---|
| DN0 | CA0 | B4 | DN1 | CA1 | D4 |
| | CB0 | A4 | | CB1 | E3 |
| | CC0 | A3 | | CC1 | D3 |
| | CD0 | B1 | | CD1 | F4 |
| | CE0 | A1 | | CE1 | F3 |
| | CF0 | B3 | | CF1 | E2 |
| | CG0 | B2 | | CG1 | D2 |
| | DP0 | D5 | | DP1 | H2 |




| GROUP | PORT | PIN | GROUP | PORT | PIN |
|---|---|---|---|---|---|
| DN0 | DK1 | G2 | DN1 | DK5 | G1 |
| | DK2 | C2 | | DK6 | F1 |
| | DK3 | C1 | | DK7 | E1 |
| | DK4 | H1 | | DK8 | G6 |

# LIGHTING THE 7-SEG-TUBE(S) ON EGO1 (DESIGN)

```
module light_7seg_ego1(input [3:0]sw, output reg [7:0] seg_out, output [7:0] seg_en);
    assign seg_en = 8'hff;
    always @ *
        case(sw)
            4'h0: seg_out = 8'b1111_1100; //0
            4'h1: seg_out = 8'b0110_0000; //1
            4'h2: seg_out = 8'b1101_1010; //2
            4'h3: seg_out = 8'b1111_0010; //3
            4'h4: seg_out = 8'b0110_0110; //4
            4'h5: seg_out = 8'b1011_0110; //5
            4'h6: seg_out = 8'b1011_1110; //6
            4'h7: seg_out = 8'b1110_0000; //7
            4'h8: seg_out = 8'b1111_1110; //8
            4'h9: seg_out = 8'b1110_0110; //9
            4'ha: seg_out = 8'b1110_1110; //A
            4'hb: seg_out = 8'b0011_1110; //B
            4'hc: seg_out = 8'b1001_1100; //C
            4'hd: seg_out = 8'b0111_1010; //D
            4'he: seg_out = 8'b1001_1110; //E
            4'hf: seg_out = 8'b1000_1110; //F
            default: seg_out = 8'b0000_0001;
        endcase
endmodule
```

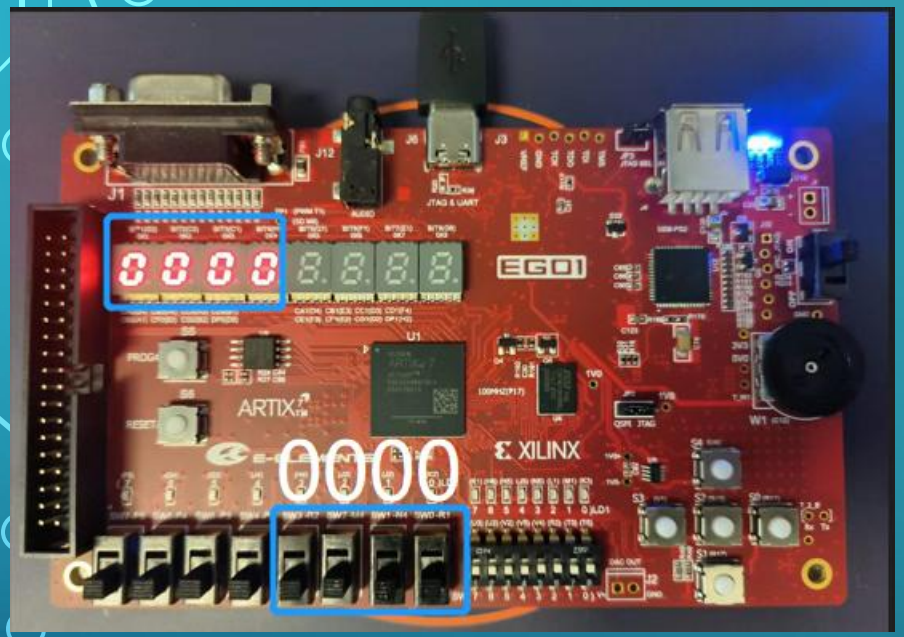


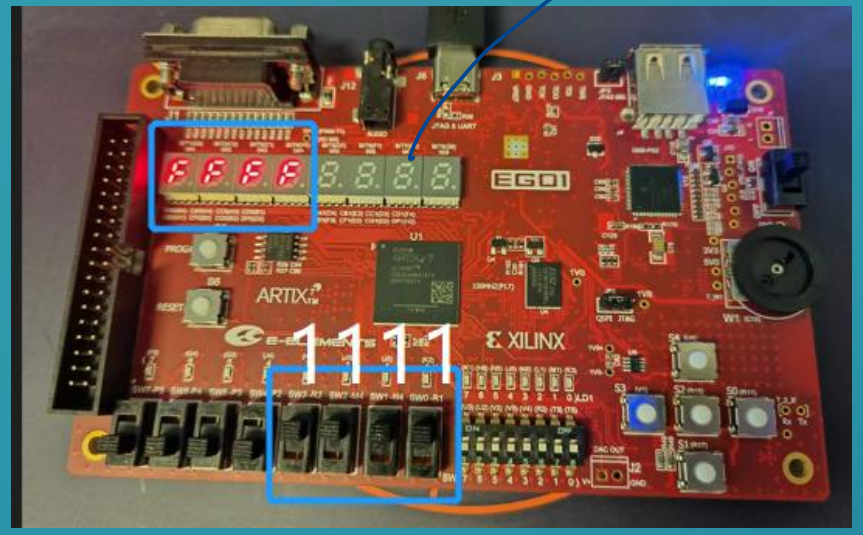

| GROUP | PORT | PIN |
|---|---|---|
| DN0 | CA0 | B4 |
| | CB0 | A4 |
| | CC0 | A3 |
| | CD0 | B1 |
| | CE0 | A1 |
| | CF0 | B3 |
| | CG0 | B2 |
| | DP0 | D5 |

Practice2：
change the design code to make “c” shown on the 7-seg-tube(s) as the above figure.

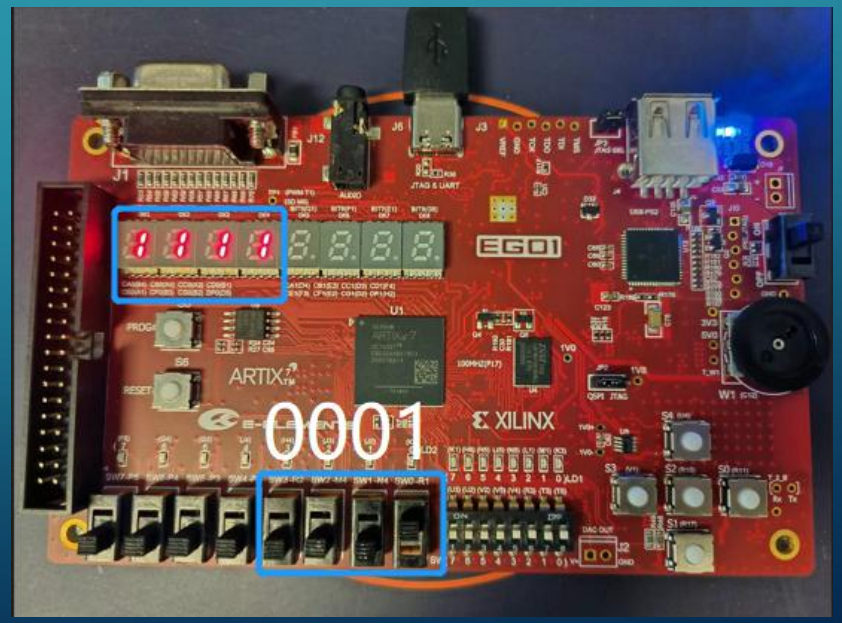
```
set_property PACKAGE_PIN B4 [get_ports {seg_out[7]}]
set_property PACKAGE_PIN A4 [get_ports {seg_out[6]}]
set_property PACKAGE_PIN A3 [get_ports {seg_out[5]}]
set_property PACKAGE_PIN B1 [get_ports {seg_out[4]}]
set_property PACKAGE_PIN A1 [get_ports {seg_out[3]}]
set_property PACKAGE_PIN B3 [get_ports {seg_out[2]}]
set_property PACKAGE_PIN B2 [get_ports {seg_out[1]}]
set_property PACKAGE_PIN D5 [get_ports {seg_out[0]}]
```

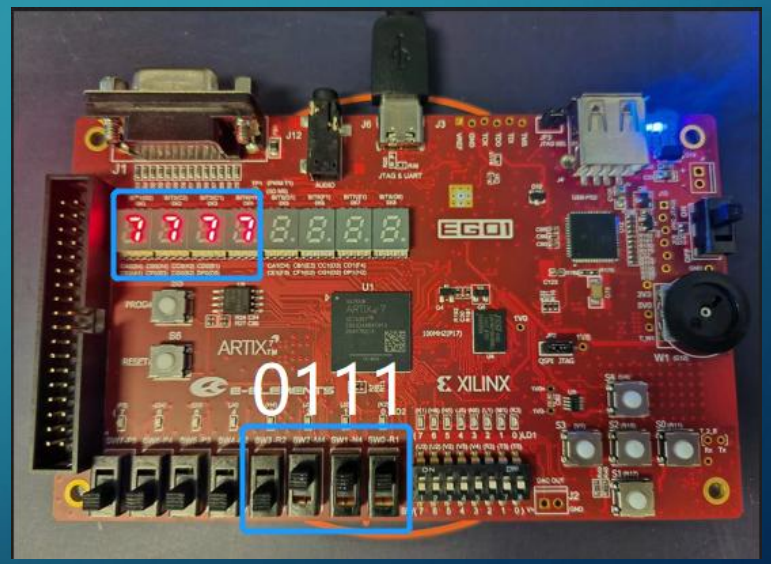
# TESTING




无控制信号. 所以不亮.
当然 seg_en = 8'hff



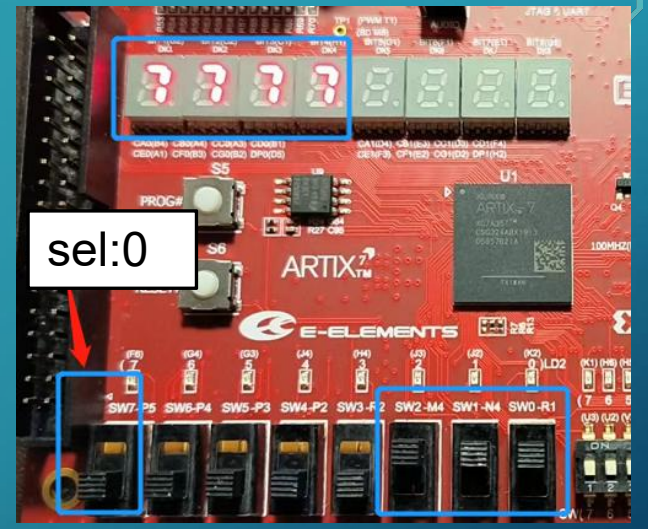
# LIGHTING THE 7-SEG-TUBE(S) ON EGO1

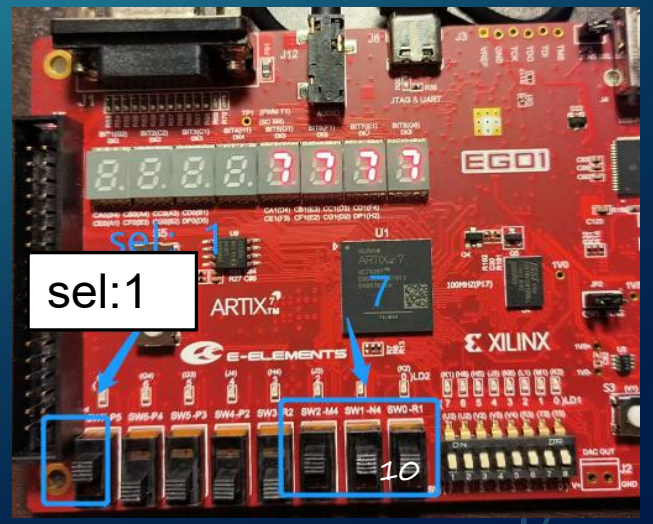
```
module light_half_7seg(input sel, input [3:0]sw, output [7:0] seg_out0, seg_out1, output reg [7:0] seg_en);

//module light_7seg_ego1(input [3:0]sw, output reg [7:0] seg_out, output [7:0] seg_en);
wire [7:0] seg_en0, seg_en1;
light_7seg_ego1 u0(sw, seg_out0, seg_en0);
light_7seg_ego1 u1(sw, seg_out1, seg_en1);

always @*
    if(sel==1'b0)
        seg_en = {4'hf, 4'h0};
    else
        seg_en = {4'h0, 4'hf};

endmodule
```




Select one of the groups of 7-seg-tubes to work

NOTE：

There should be new constraint file works with the new design.

# PRACTICE(3)

- There are sixteen wards, which are numbered from 0 to F respectively, among which #0 ward has the highest priority, #F has the lowest priority(Priority decreases as the number increases).
- Each room has an call bell, it could be turn on or turn off. In the main control room there is a display screen which shows the ID of the room whose bell is on with highest priority of all the bell on room.
- Please write a circuit to implement this function and test.
  - Do the design and verify the function of your design.
  - Create the constrain file, do the synthetic and implementation, generate the bitstream file and program the device, then test on the minisys develop board.

# PRACTICES(4)

Design a 3bit width full adder with 3 inputs (a, b, cin) and 2 outputs(sum, cout), the width of a,b and sum are both 3bit width while cin and cout are both 1bit width

1. Do the design by using structure design.
2. Make a testbench to verify its function.
3. Create the constrain file, do the synthetic and implementation, generate the bitstream file and program the device, then test on the minisys develop board.
   1) Switches are suggested to be used as input device
   2) Select one of the following three options as the output device:
      a) one led as the 'cout' and 7-segment tubes as the 'sum'
      b) 7-segment tubes as the 'cout' and 'sum'