

# 7-seg display

---

## DD Lab4

助教: 胡祐嘉、劉宸彥



Department of Electrical Engineering and SoC Research Center National Chung Cheng University

# Outline

---

- 課程目的
- 七段顯示器原理與架構
- 課程練習內容
- Lab作業
- 課程評分方式

# 課程目的

---

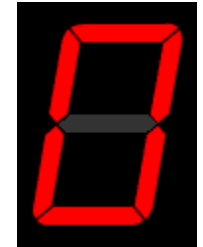
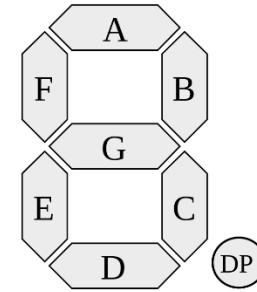
經過了先前的實驗課，我們已經了解到如何設計加法器以及將Verilog燒錄到Nexys-4 FPGA上，控制周邊電路，在這堂課則會教大家

1. 了解七段顯示器原理與架構
2. 學習透過Verilog控制FPGA上的七段顯示器
3. 能夠將先前課程的實作內容以七段顯示器顯示

# 七段顯示器原理與架構 (1/2)

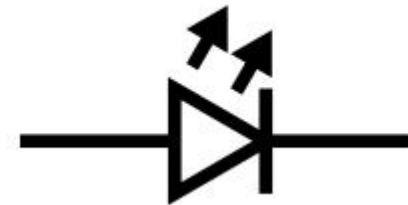
## ■ 什麼是七段顯示器

- 七段顯示器(seven-segment display)為一用來顯示數字的電子元件
- 藉由七個發光二極體(LED)以不同組合來顯示數字(若有小數點位則八個)
- 分為共陽極及共陰極兩種



## ■ 什麼是發光二極體

- 與其他二極體一樣，LED內的電流可以從陽極流向陰極，反之則不能，當電流流過時發光
- 因此可以透過控制陽極及陰極的電位來控制LED發光



發光二極體



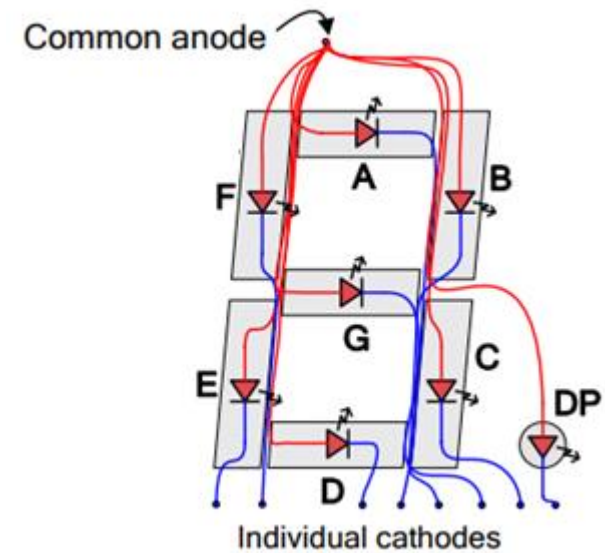
二極體

# 七段顯示器原理與架構 (2/2)

## ■ 共陽極與共陰極的差別

- 共陽極與共陰極指的是七段顯示器內的一端提供相同的電位
- 因此**共陽極**透過改變**陰極電位**來控制七段顯示器顯示
- 而**共陰極**透過改變**陽極電位**來控制七段顯示器顯示

- E.g:Nexys-4內的七段顯示器為共陽極，透過輸入控制訊號，告訴FPGA版LED陰極是否要接地來控制LED顯示(若訊號為0則接地，LED亮起)

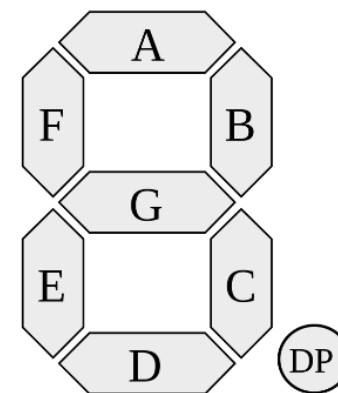


共陽極七段顯示器示意圖[1]

# Practice Questions 1

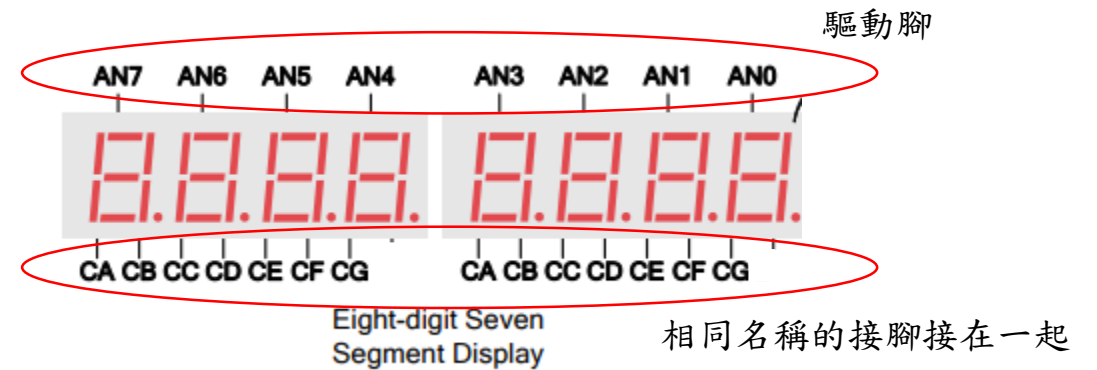
---

- 已知Nexys-4使用的七段顯示器為共陽極，且透過七位元訊號控制LED，則若要顯示數字4則需要輸入什麼訊號？
- ABCDEFG等七個LED的控制訊號分別放在七位元訊號的第0到6位，且訊號1代表高電位，訊號0代表低電位(接地)



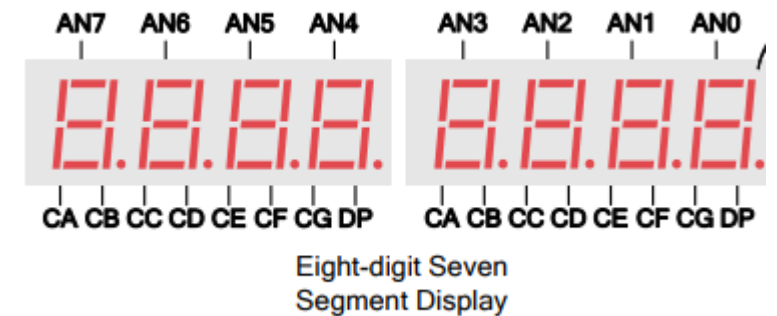
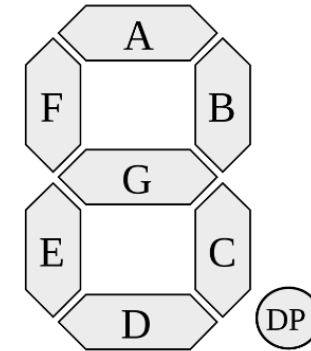
# 多個七段顯示器控制(1/2)

- 為了減少硬體的複雜度，通常會將多個七段顯示器相同名稱的接腳連接在一起，並且每一個七段顯示器有獨立的**驅動腳**(如右圖)，各自接收訊號決定七段顯示器是否顯示。



# Practice Questions 2

- 已知Nexys-4使用的七段顯示器為共陽極，且透過七位元訊號控制LED，並另外透過八位元訊號控制驅動腳，則若要使第3位的七段顯示器顯示數字5則需要輸入什麼訊號？
- CA、CB.....CG等七個LED的控制訊號分別放在七位元訊號的第0到6位，且訊號1代表高電位，訊號0代表低電位(接地)。
- AN7、AN6.....AN0等八個驅動腳的控制訊號由右至左放在八位元訊號的第0到第7位，且訊號1代表七段顯示器不顯示，0則代表該七段顯示器顯示。

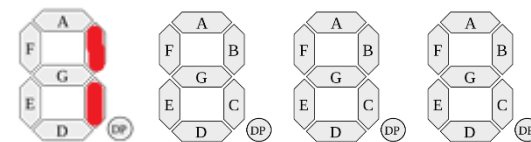




# 多個七段顯示器控制(2/2)

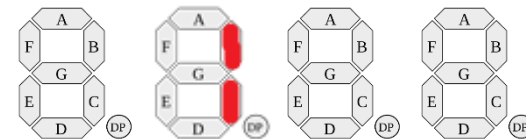
- 前述架構的七段顯示器無法在不同顯示器上同時顯示不同值，為了看起來像是多個七段顯示器顯示不同值，我們需以人眼無法察覺的速度去掃描多個七段顯示器。
- 人類的視覺暫留平均時間為1/16秒，因此每個七段顯示器的顯示時間必須小於 $1/(16 * N)$ 秒，其中N為七段顯示器的位數。
- 但若七段顯示器的掃描時間太短，則通過的電流會過小導致顯示器過暗甚至不亮。

控制訊號: 1111001, 0111

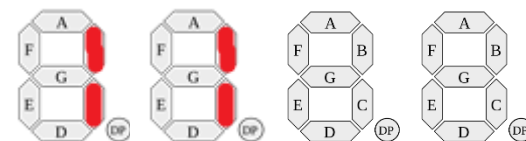


第一次掃描

控制訊號: 1111001, 1011



第二次掃描



人眼看到的

# 課程教材及練習內容

## ■ 課程練習項目

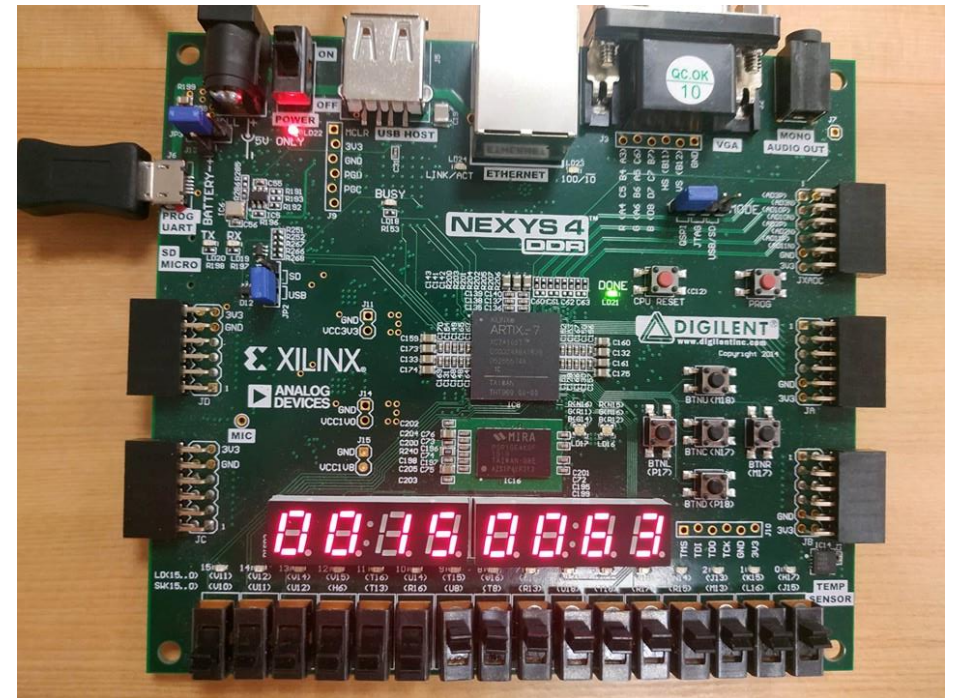
- 課程提供的RTL Code 會將Switch輸入以十進制顯示在七段顯示器上。
- 修改提供的RTL Code，結合LAB1使用的加法器，將Switch輸入相加並顯示在七段顯示器上。

## ■ 實驗教學內容

- 學習如何使用控制訊號控制七段顯示器顯示

## ■ 教材內容

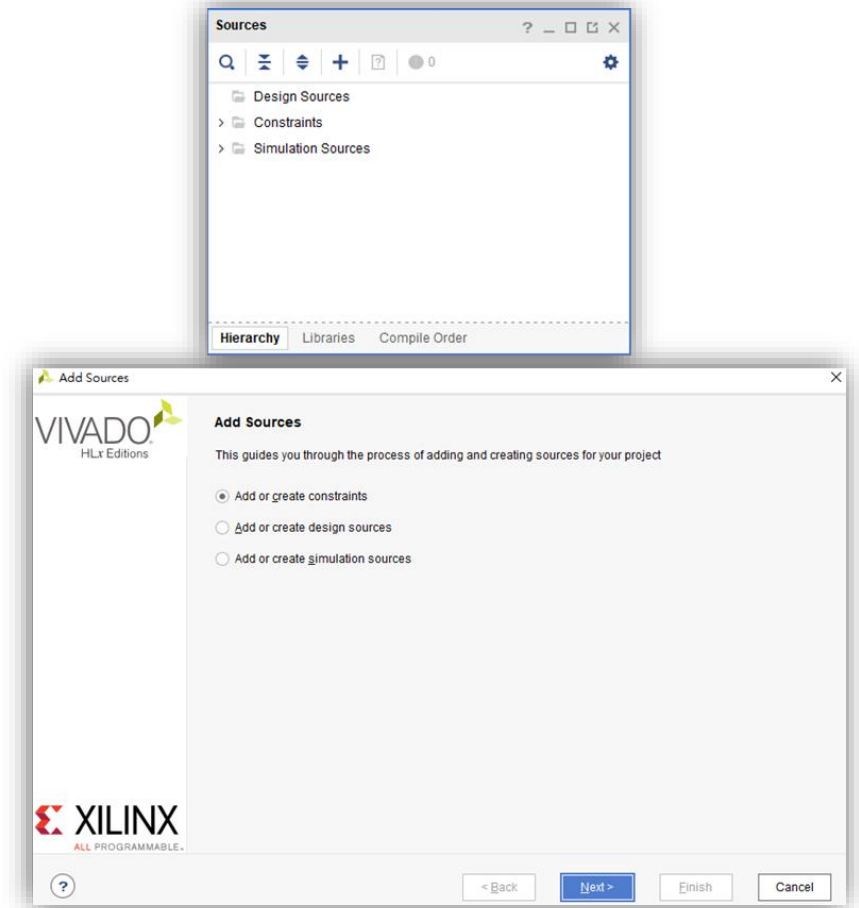
- 一份.v檔以及一份.xdc檔
  - 7-seg.v用來描述輸入輸出
  - Nexys4.xdc用來描述.v與實體線路的連接關係



教材提供的RTL燒錄後示意圖，會將輸入顯示在七段顯示器上

# 建立專案

- 如同LAB3所教，建立專案，並透過Add Sources匯入.v檔與.xdc檔，在此就不贅述



# 教材說明(1/3)

## ■ Xilinx Design Constraints file(xdc file)

□ T10、R10.....L18為LED控制PIN腳

□ J17、J18.....U13為七段顯示器驅動腳

七段顯示器LED控制

```
##7 segment display

set_property -dict { PACKAGE_PIN T10      IOSTANDARD LVCMOS33 } [get_ports { CA }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10      IOSTANDARD LVCMOS33 } [get_ports { CB }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16      IOSTANDARD LVCMOS33 } [get_ports { CC }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13      IOSTANDARD LVCMOS33 } [get_ports { CD }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15      IOSTANDARD LVCMOS33 } [get_ports { CE }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11      IOSTANDARD LVCMOS33 } [get_ports { CF }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18      IOSTANDARD LVCMOS33 } [get_ports { CG }]; #IO_L4P_T0_D04_14 Sch=cg

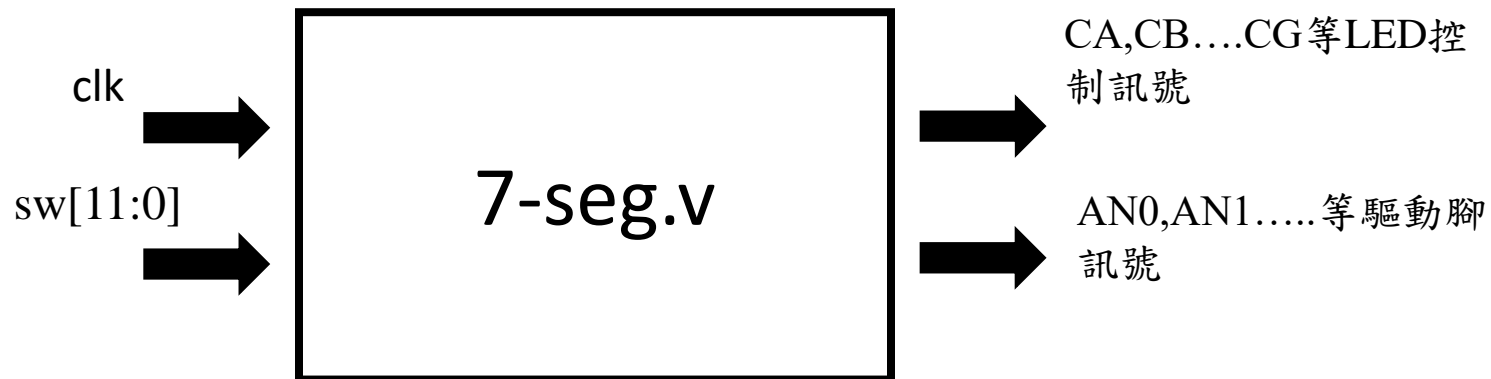
#set_property -dict { PACKAGE_PIN H15      IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp

set_property -dict { PACKAGE_PIN J17      IOSTANDARD LVCMOS33 } [get_ports { AN0 }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18      IOSTANDARD LVCMOS33 } [get_ports { AN1 }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9       IOSTANDARD LVCMOS33 } [get_ports { AN2 }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14      IOSTANDARD LVCMOS33 } [get_ports { AN3 }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14      IOSTANDARD LVCMOS33 } [get_ports { AN4 }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14      IOSTANDARD LVCMOS33 } [get_ports { AN5 }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2       IOSTANDARD LVCMOS33 } [get_ports { AN6 }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13      IOSTANDARD LVCMOS33 } [get_ports { AN7 }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
```

七段顯示器驅動腳控制

# 教材說明(2/3)

- 先定義輸入輸出與宣告，輸入為clk和Switch元件訊號，輸出為驅動腳控制訊號及LED控制訊號。
  - clk為時脈訊號，在同步電路中扮演計時器的角色。
- 將sw[11:6]與sw[5:0]分別放在不同的wire，表示兩筆不同的數字。



```
1 module top(  
2     clk,  
3     sw,  
4     CA,  
5     CB,  
6     CC,  
7     CD,  
8     CE,  
9     CF,  
10    CG,  
11    AN0,  
12    AN1,  
13    AN2,  
14    AN3,  
15    AN4,  
16    AN5,  
17    AN6,  
18    AN7  
19 );  
20  
21 input clk;  
22 input [11:0]sw;  
23 output CA,CB,CC,CD,CE,CF,CG;  
24 output AN0,AN1,AN2,AN3,AN4,AN5,AN6,AN7;  
25  
26 wire [5:0]num1;  
27 wire [5:0]num2;  
28 reg [2:0] state;  
29 reg [6:0] seg_number,seg_data;  
30 reg [20:0] counter;  
31 reg [7:0] scan;  
32  
33  
34  
35 assign num1 = sw[11:6];  
36 assign num2 = sw[5:0];  
37
```

# 教材說明(3/3)

```
38 assign {AN7,AN6,AN5,AN4,AN3,AN2,AN1,AN0} = scan;
39 always@(posedge clk) begin
40     counter <= (counter <= 100000) ? (counter + 1) : 0;
41     state <= (counter == 100000) ? (state + 1) : state;
42     case(state)
43     0:begin
44         seg_number <= num1 / 1000;
45         scan <= 8'b0111_1111;
46     end
47     1:begin
48         seg_number <= (num1 / 100) % 10;
49         scan <= 8'b1011_1111;
50     end
51     2:begin
52         seg_number <= (num1 / 10) % 10;
53         scan <= 8'b1101_1111;
54     end
55     3:begin
56         seg_number <= num1 % 10;
57         scan <= 8'b1110_1111;
58     end
59     4:begin
60         seg_number <= num2 / 1000;
61         scan <= 8'b1111_0111;
62     end
63     5:begin
64         seg_number <= (num2 / 100) % 10;
65         scan <= 8'b1111_1011;
66     end
67     6:begin
68         seg_number <= (num2 / 10) % 10;
69         scan <= 8'b1111_1101;
70     end
71     7:begin
72         seg_number <= num2 % 10;
73         scan <= 8'b1111_1110;
74     end
75     default: state <= state;
76 endcase
77 end
```

每100000個clk改變一次state

```
79
80 assign {CG,CF,CE,CD,CC,CB,CA} = seg_data;
81
82 always@(posedge clk) begin
83     case(seg_number)
84         16'd0: seg_data <= 7'b100_0000;
85         16'd1: seg_data <= 7'b111_1001;
86         16'd2: seg_data <= 7'b010_0100;
87         16'd3: seg_data <= 7'b011_0000;
88         16'd4: seg_data <= 7'b001_1001;
89         16'd5: seg_data <= 7'b001_0010;
90         16'd6: seg_data <= 7'b000_0010;
91         16'd7: seg_data <= 7'b101_1000;
92         16'd8: seg_data <= 7'b000_0000;
93         16'd9: seg_data <= 7'b001_0000;
94         default: seg_data <= seg_data;
95     endcase
96
97 end
```

根據算出的值控制LED的顯示

分別將num1, num2轉成十進位下不同位數應顯示的值  
並根據state控制驅動腳



# 課程練習

1. 修改先前課程的4bit加法器，使其可以進行6bit加法(不考慮overflow)。
2. 在7-seg.v內額外宣告sum接線，並將num1,num2接到加法器中。
3. 將決定LED輸出數字的地方，從num1,num2改成sum

1.

```
D: > googledrive > Seminar > 助教課程 > DD > NEV
1  module adder4bit(A,B,sum);
2  input [5:0] A;
3  input [5:0] B;
4  output [6:0] sum;
5
6  assign sum = A + B;
7
8  endmodule
9
```

2.

```
wire sum;

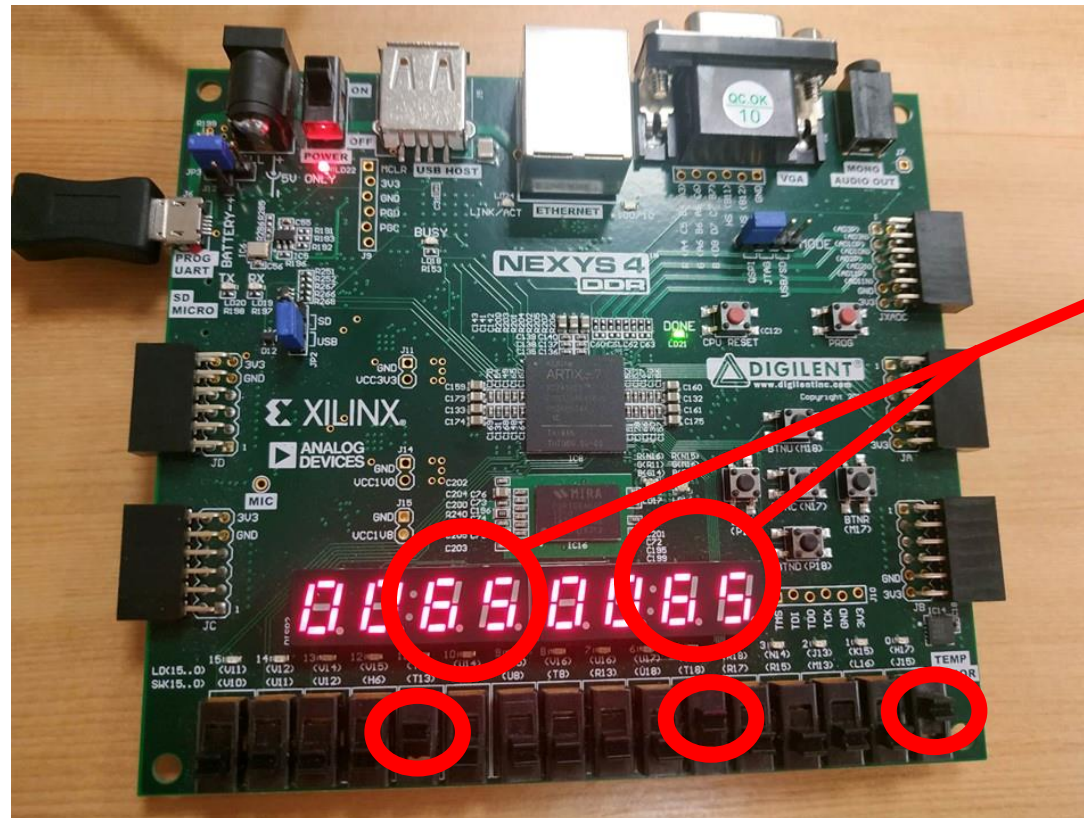
adder4bit ad6(num1,num2,sum);
```

3.

```
case(state)
0:begin
    seg_number <= sum / 1000;
    scan <= 8'b0111_1111;
end
1:begin
    seg_number <= (sum / 100) % 10;
    scan <= 8'b1011_1111;
end
2:begin
    seg_number <= (sum / 10) % 10;
    scan <= 8'b1101_1111;
end
3:begin
    seg_number <= sum%10;
    scan <= 8'b1110_1111;
```

# 課程練習成果

- 根據輸入的switch，將sw[11:6]與sw[5:0]相加並以十進位表示在七段顯示器(如下圖範例)



輸出值皆為相加結果65

第11,5,0個switch打開，相加。

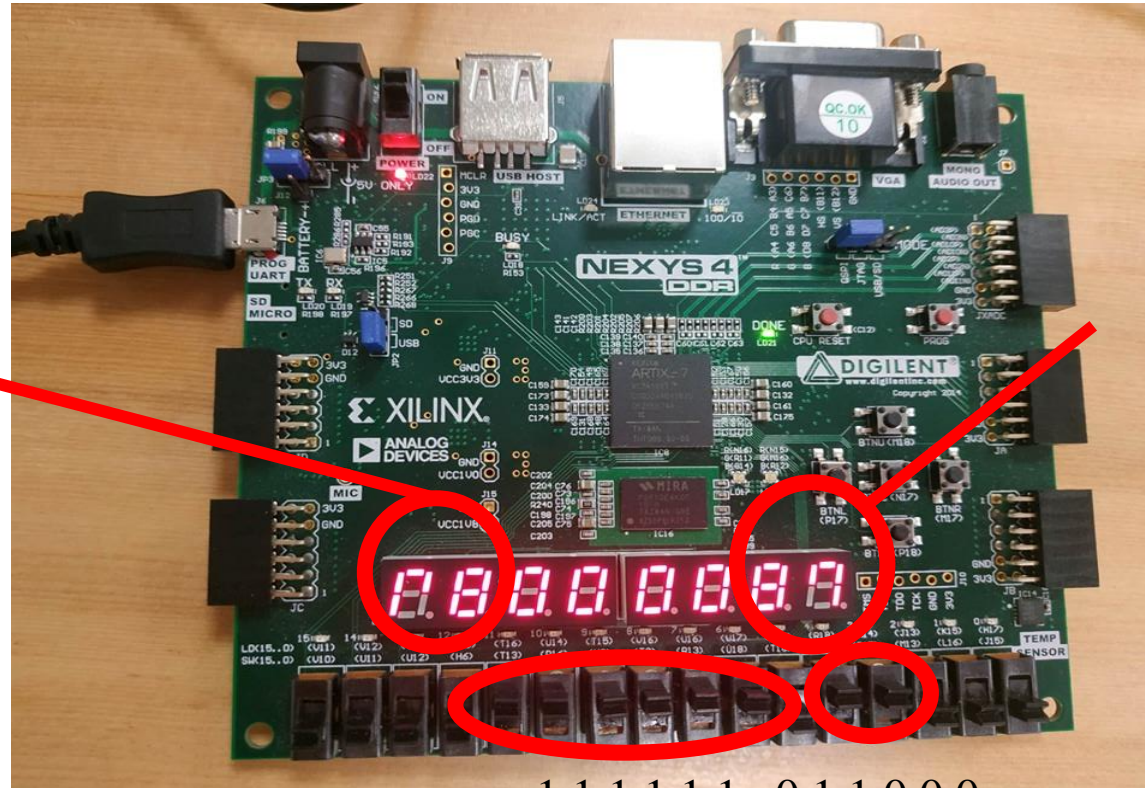
10 0 0 0 0 , 1 0 0 0 0 1  
(32, 33)



# Lab作業

- 用 12 顆 switches (sw0~sw11) 分別控制兩筆輸入，將兩筆輸入相加結果顯示在最右邊兩個七段顯示器，並在最左邊兩個七段顯示器顯示對應的鏡像(如下圖)

顯示為相加結果87的鏡像



顯示為相加結果87

第11~6,4,3個switch打開，相加。

111111, 011000

(63,24)

# 課程評分

---

■ Demo時間:依E-Course公布為準

■ Demo梯次:依E-Course公布為準

■ Demo地點:依E-Course公布為準

■ 評分方式

1.執行課程教材20%

2.完成課程練習20%

2.完成Lab作業60%

記得填寫意見回饋表，否則不予以計分