

Lab 5

Counter and Birthdate Display

助教：林冠翰、廖辰綜

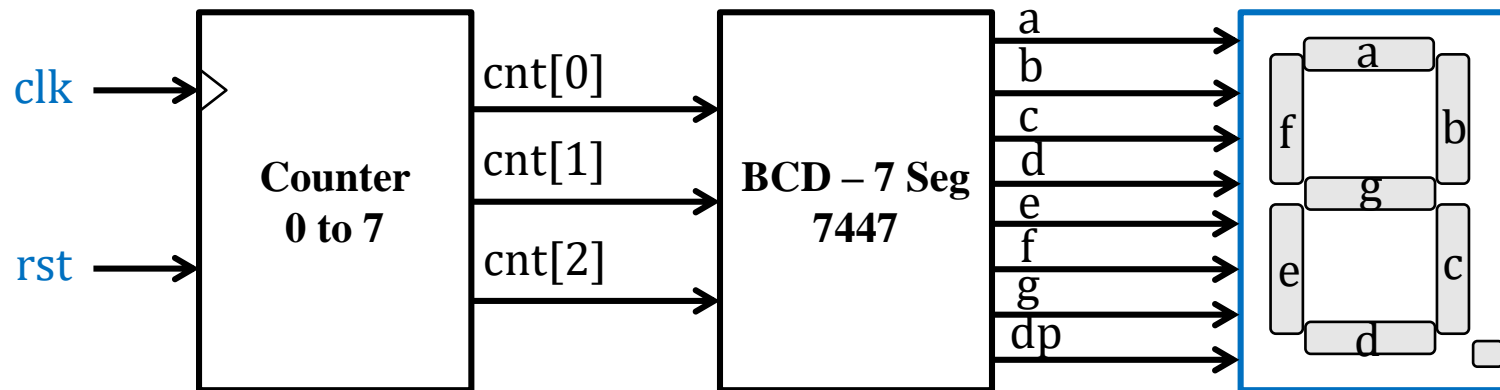
Department of Electrical Engineering and SoC Research Center
National Chung Cheng University



- ❖ 課程目標
- ❖ 循序顯示0~7
- ❖ 生日碼顯示器
- ❖ 驗收內容 & 配分

- ❖ 承上個Lab，大家已經知道如何使用Nexy4 DDR FPGA開發版上的七段顯示器、開關等功能，本次課程將教大家
 - 使用Counter、BCD，在七段顯示器上依序顯示0~7
 - 實作Code Converter組合電路，並控制七段顯示器顯示0~7或生日碼

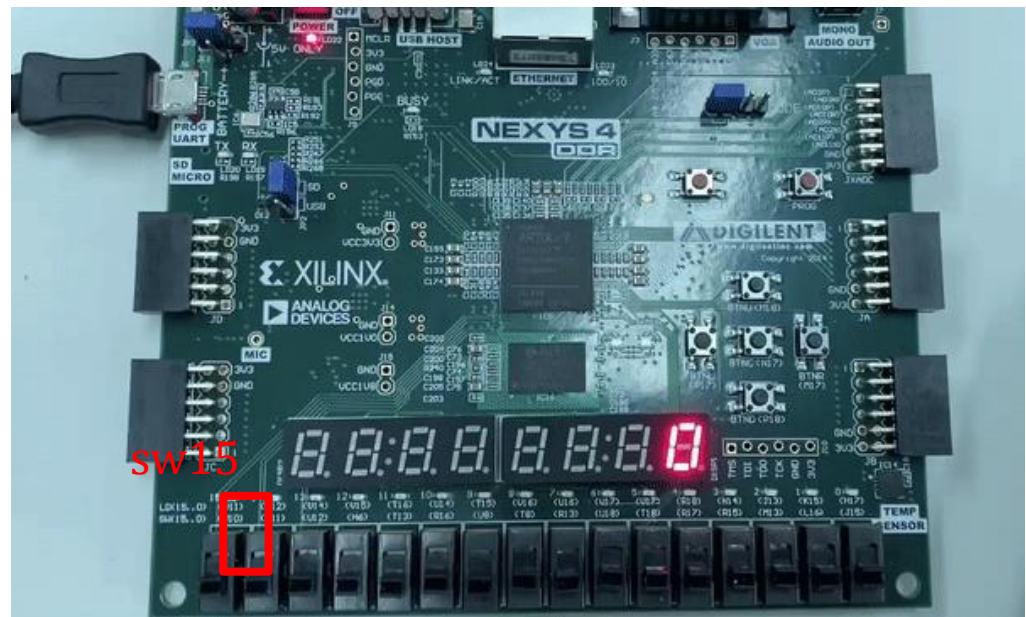
課堂練習_循序0~7



☐ FPGA周邊

❖ sw15 (rst)

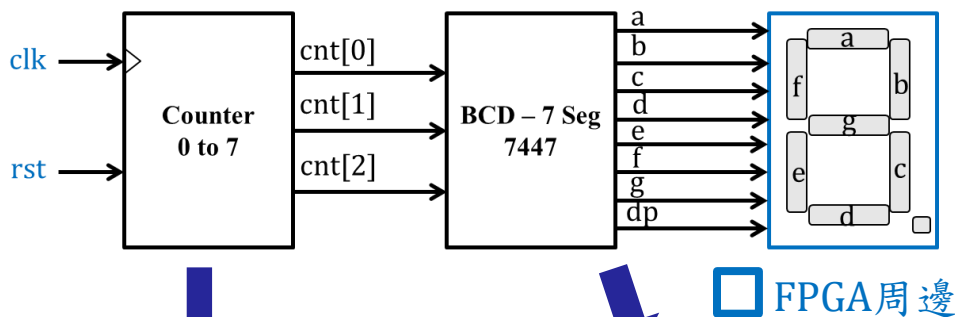
- 0 : 維持在0
- 1 : 循序顯示0~7



展示範例 (需放映投影片)

循序0~7_程式說明

- ❖ 以counter為input，七段顯示器上各段LED為output，利用課堂中所學到的 min SOP (sum of product) 來化簡我們的布林函數，以下為範例程式：



cnt[2:0]	a	b	c	d	e	f	g
0(000)	0	0	0	0	0	0	1
1(001)	1	0	0	1	1	1	1
2(010)	0	0	1	0	0	1	0
3(011)	0	0	0	0	1	1	0
4(100)	1	0	0	1	1	0	0
5(101)	0	1	0	0	1	0	0
6(110)	0	1	0	0	0	0	0
7(111)	0	0	0	1	1	0	1

```
/**COUNTER**/  
always@ (posedge clk_2hz or negedge rst) begin  
    if (~rst)  
        cnt <= 3'b0;  
    else  
        cnt <= cnt + 3'b1;  
end
```

```
/**BCD_to_7seg**/  
always@(*) begin  
    seg_data[0] = (cnt[2] & !cnt[1] & !cnt[0]) | (!cnt[2] & !cnt[1] & cnt[0]); //CA  
    seg_data[1] = (cnt[2] & !cnt[1] & cnt[0]) | (cnt[2] & cnt[1] & !cnt[0]); //CB  
    seg_data[2] = !cnt[2] & cnt[1] & !cnt[0]; //CC  
    seg_data[3] = (cnt[2] & !cnt[1] & !cnt[0]) | (!cnt[2] & !cnt[1] & cnt[0]) | (cnt[2] & cnt[1] & cnt[0]); //CD  
    seg_data[4] = (cnt[2] & !cnt[1]) | (!cnt[2] & cnt[0]) | (cnt[2] & cnt[0]); //CE  
    seg_data[5] = (!cnt[2] & cnt[0]) | (!cnt[2] & cnt[1]); //CF  
    seg_data[6] = (!cnt[2] & !cnt[1]) | (cnt[2] & cnt[1] & cnt[0]); //CG  
end
```

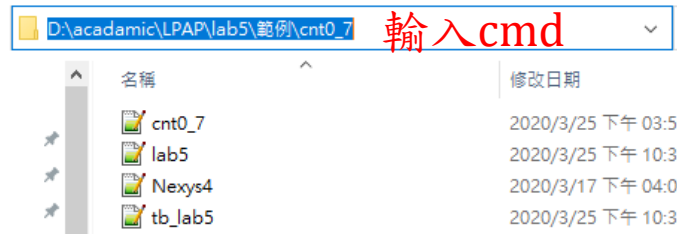
➤ 因為實驗目的是顯示0~7，因此3bit剛好可以完全表示我們讓 counter 每一個 clock cycle就加一，以達到循序的效果。

➤ 因為 cnt經過每一個 clock cycle都會加一，也就代表每一個cycle的輸出會不同。使用BCD - 7seg (7447)的真值表並利用 min sop所化簡的布林函數可讓我們在七段顯示器上依序顯示0~7。

- ❖ 同學在寫demo作業時先利用testbench確認自己撰寫的Code功能正確後，再上板子進行驗證
 - 程式模擬 (於命令提示字元輸出，觀察結果是否與目標一致)
 - 周邊整合 (將verilog中的input、output與FPGA上的腳位連接)
 - 上板驗證 (觀察七段顯示器)

循序顯示0~7_程式模擬

- ❖ 開啟資料夾 lab5/範例/cnt0_7 並於路徑處輸入cmd開啟命令提示字元



- ❖ 輸入「iverilog -o test tb_lab5.v」

➤ 輸入檔案的第一個字母後按「tab」即會出現第一字母相同的檔案

- ❖ 輸入「vvp test」

- ❖ 即可觀察到 output根據cnt的不同在輸出

```
D:\academic\LPAP\lab5\範例\cnt0_7>vvp test
VCD info: dumpfile lab5.fsdb opened for output.
250005 cnt = 1, output = 1111001
500005 cnt = 2, output = 0100100
750005 cnt = 3, output = 0110000
1000005 cnt = 4, output = 0011001
1250005 cnt = 5, output = 0010010
1500005 cnt = 6, output = 0000010
1750005 cnt = 7, output = 1011000
2000005 cnt = 0, output = 1000000
2250005 cnt = 1, output = 1111001
2500005 cnt = 2, output = 0100100
2750005 cnt = 3, output = 0110000
3000005 cnt = 4, output = 0011001
```

循序顯示0~7_周邊整合

❖ UCF (User Constraint File)

- 參考Lab4第七頁，因本次實驗之須用到最右邊的七段顯示器，所以我們只需另AN0為0即可
- 七段顯示器輸出的數字則由BCD-7seg的輸出「seg_data」來決定
- rst設定在sw15

```
assign {AN7,AN6,AN5,AN4,AN3,AN2,AN1,AN0} = 8'b1111_1110; //turn on the 8th 7seg LED
assign {CG,CF,CE,CD,CC,CB,CA} = seg_data;
```

##7 segment display

Pin	Signal	Verilog Output
T10	IOSTANDARD LVCMOS33 [get_ports	CA]; #IO_L24N_T3_A00_D16_14 Sch=ca
R10	IOSTANDARD LVCMOS33 [get_ports	CB]; #IO_25_14 Sch=cb
K16	IOSTANDARD LVCMOS33 [get_ports	CC]; #IO_25_15 Sch=cc
K13	IOSTANDARD LVCMOS33 [get_ports	CD]; #IO_L17P_T2_A26_15 Sch=cd
P15	IOSTANDARD LVCMOS33 [get_ports	CE]; #IO_L13P_T2_MRCC_14 Sch=ce
T11	IOSTANDARD LVCMOS33 [get_ports	CF]; #IO_L19P_T3_A10_D26_14 Sch=cf
L18	IOSTANDARD LVCMOS33 [get_ports	CG]; #IO_L4P_T0_D04_14 Sch=cg
J17	IOSTANDARD LVCMOS33 [get_ports	AN0]; #IO_L23P_T3_F0E_B_15 Sch=an[0]
J18	IOSTANDARD LVCMOS33 [get_ports	AN1]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
T9	IOSTANDARD LVCMOS33 [get_ports	AN2]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
J14	IOSTANDARD LVCMOS33 [get_ports	AN3]; #IO_L19P_T3_A22_15 Sch=an[3]
P14	IOSTANDARD LVCMOS33 [get_ports	AN4]; #IO_L8N_T1_D12_14 Sch=an[4]
T14	IOSTANDARD LVCMOS33 [get_ports	AN5]; #IO_L14P_T2_SRCC_14 Sch=an[5]
K2	IOSTANDARD LVCMOS33 [get_ports	AN6]; #IO_L23P_T3_35 Sch=an[6]
U13	IOSTANDARD LVCMOS33 [get_ports	AN7]; #IO_L23N_T3_A02_D18_14 Sch=an[7]

七段顯示器驅動腳位

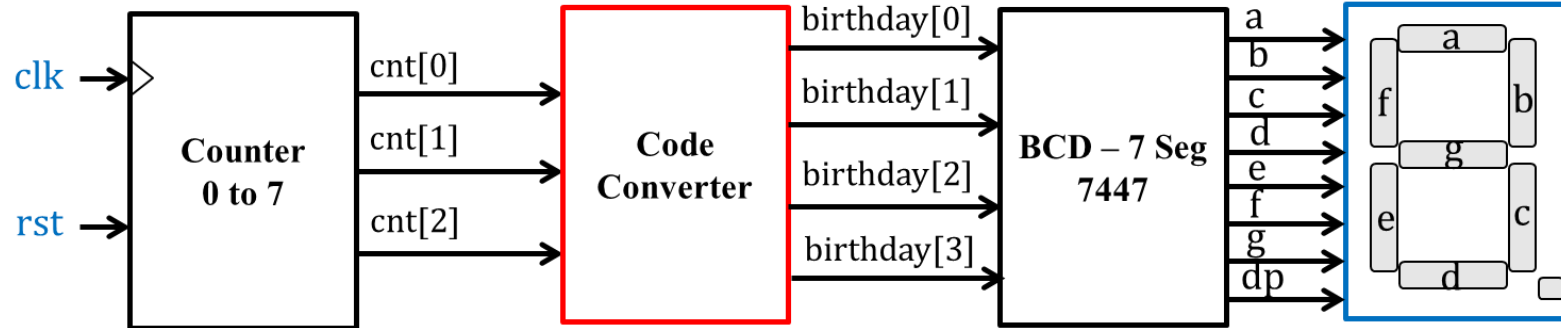
```
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { rst }]; #IO_L21P_T3_DQS_14 Sch=sw[15]
```

sw15在板上的代碼

verilog中的rst

- ❖ 前面解說完 Counter和 7447之間的關係之後，接下來我們要介紹 Code Converter 使我們可以延伸應用前面的 design，在七段顯示器上依序顯示生日。

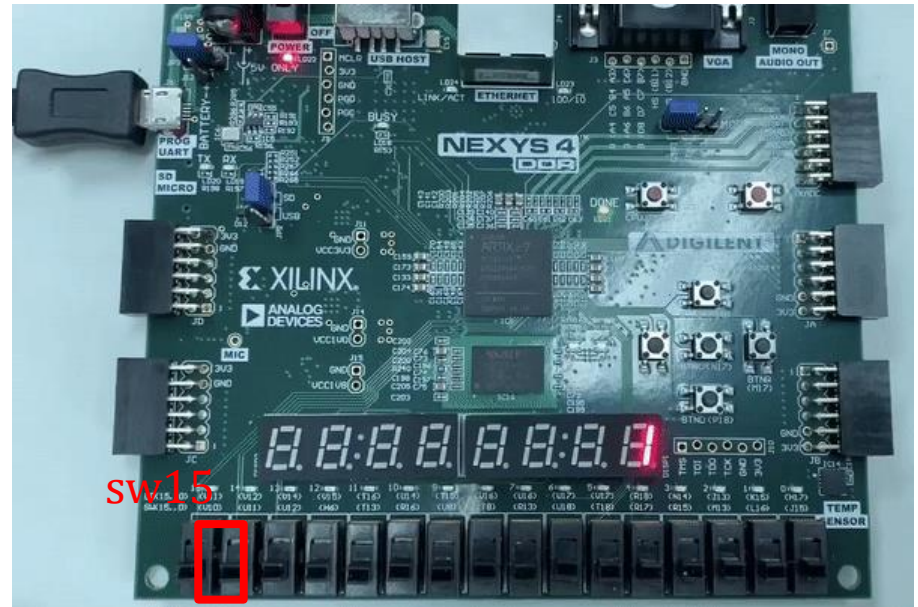
課堂練習_生日碼顯示器



☐ FPGA周邊

❖ sw15 (rst)

- 0：維持在1 (生日碼第一個數字)
- 1：循序顯示生日碼

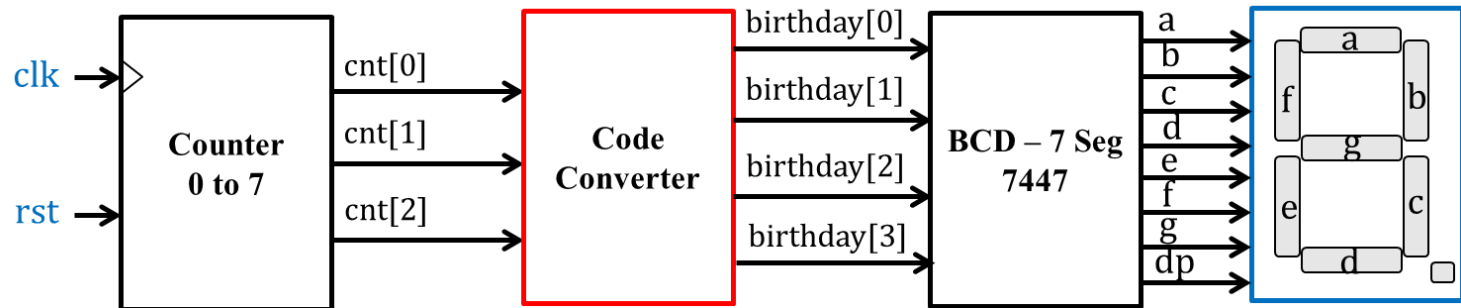


展示範例 (需放映投影片)

Code Converter 實作 (1/2)

❖ min SOP (Sum of Product)

- 生日碼顯示器可以沿用前一個範例的想法，一樣以 cnt 為輸入，只是這次在 Code Converter 的 output 會依序顯示出預先設計的生日，再利用 7447 將生日輸出至七段顯示器



□ FPGA周邊

```
/**birthday_display**/
always@(*)begin
    birth_num[3] = (!cnt[2] & !cnt[1] & cnt[0]) | (cnt[2] & cnt[1] & cnt[0]);
    birth_num[2] = !cnt[2] & cnt[1];
    birth_num[1] = (!cnt[2] & cnt[1]) | (cnt[2] & !cnt[1] & cnt[0]);
    birth_num[0] = (!cnt[2] & !cnt[1]) | (cnt[2] & cnt[0]) | (cnt[1] & !cnt[0]);
end
```

- Input為 counter，利用min sop所化簡的方程式來輸出預先設計的生日碼

- 跟上一個範例不同，這次需要輸出到9，所以input會用到4bit

```
/**BCD_to_7SEG**/
always@(*) begin
    seg_data[6] = (!birth_num[3] & !birth_num[2] & !birth_num[1]) | (!birth_num[3] & birth_num[2] & birth_num[1] & birth_num[0]);
    seg_data[5] = (!birth_num[3] & !birth_num[2] & birth_num[0]) | (!birth_num[3] & !birth_num[2] & birth_num[1]);
    seg_data[4] = (!birth_num[3] & birth_num[2] & !birth_num[1]) | (!birth_num[2] & !birth_num[1] & birth_num[0]) | (!birth_num[3] & birth_num[2] & birth_num[0]) | (!birth_num[3] & birth_num[1] & birth_num[0]);
    seg_data[3] = (!birth_num[3] & birth_num[2] & !birth_num[1] & !birth_num[0]) | (!birth_num[3] & !birth_num[2] & !birth_num[1] & birth_num[0]) | (!birth_num[3] & birth_num[2] & birth_num[1] & birth_num[0]);
    seg_data[2] = (!birth_num[3] & !birth_num[2] & birth_num[1] & !birth_num[0]);
    seg_data[1] = (!birth_num[3] & birth_num[2] & !birth_num[1] & birth_num[0]) | (!birth_num[3] & birth_num[2] & birth_num[1] & !birth_num[0]);
    seg_data[0] = (!birth_num[3] & birth_num[2] & !birth_num[1] & !birth_num[0]) | (!birth_num[3] & !birth_num[2] & !birth_num[1] & birth_num[0]);
end
```

Code Converter 實作 (2/2)

❖ High-level Verilog description

- 每次都要找 min SOP 其實很費力，在 verilog 語法中有一個方法為「case」，可以使 input 快速轉換成我們需要的輸出，這邊就以生日碼顯示器作為範例。

❖ Code Converter

根據括弧中的 input 會有相對應的 output，
括弧中可為定值或變數，也可放運算子

```
case(cnt)
  3'b000:
    seg_number = 4'd1;
  3'b001:
    seg_number = 4'd9;
  3'b010:
    seg_number = 4'd7;
  3'b011:
    seg_number = 4'd6;
  3'b100:
    seg_number = 4'd0;
  3'b101:
    seg_number = 4'd3;
  3'b110:
    seg_number = 4'd1;
  3'b111:
    seg_number = 4'd9;
endcase
```

❖ BCD to 7seg (7447)

```
always@(*) begin
  case(seg_number)
    4'd0: seg_data = 7'b100_0000;
    4'd1: seg_data = 7'b111_1001;
    4'd2: seg_data = 7'b010_0100;
    4'd3: seg_data = 7'b011_0000;
    4'd4: seg_data = 7'b001_1001;
    4'd5: seg_data = 7'b001_0010;
    4'd6: seg_data = 7'b000_0010;
    4'd7: seg_data = 7'b101_1000;
    4'd8: seg_data = 7'b000_0000;
    4'd9: seg_data = 7'b001_0000;
    default: seg_number = seg_number;
  endcase
end
```

生日碼顯示器_周邊整合

- ❖ 因生日碼顯示器的input、output皆與循序顯示0~7的範例相同，故UFC皆與p. 8相同，在此不再贅述

```
assign {AN7,AN6,AN5,AN4,AN3,AN2,AN1,AN0} = 8'b1111_1110; //turn on the 8th 7seg LED
assign {CG,CF,CE,CD,CC,CB,CA} = seg_data;
```

##7 segment display

Pin	IO	Standard	Level	Port	Verilog Output
T10	IOSTANDARD	LVC	MOS33	[get_ports CA]	CA
R10	IOSTANDARD	LVC	MOS33	[get_ports CB]	CB
K16	IOSTANDARD	LVC	MOS33	[get_ports CC]	CC
K13	IOSTANDARD	LVC	MOS33	[get_ports CD]	CD
P15	IOSTANDARD	LVC	MOS33	[get_ports CE]	CE
T11	IOSTANDARD	LVC	MOS33	[get_ports CF]	CF
L18	IOSTANDARD	LVC	MOS33	[get_ports CG]	CG
J17	IOSTANDARD	LVC	MOS33	[get_ports AN0]	AN0
J18	IOSTANDARD	LVC	MOS33	[get_ports AN1]	AN1
T9	IOSTANDARD	LVC	MOS33	[get_ports AN2]	AN2
J14	IOSTANDARD	LVC	MOS33	[get_ports AN3]	AN3
P14	IOSTANDARD	LVC	MOS33	[get_ports AN4]	AN4
T14	IOSTANDARD	LVC	MOS33	[get_ports AN5]	AN5
K2	IOSTANDARD	LVC	MOS33	[get_ports AN6]	AN6
U13	IOSTANDARD	LVC	MOS33	[get_ports AN7]	AN7

七段顯示器驅動腳位

```
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { rst }]; #IO_L21P_T3_DQS_14 Sch=sw[15]
```

sw15在板上的代碼

verilog中的rst

驗收內容&配分

❖ 利用 testbench 於命令提示字元中顯示生日 (20%)

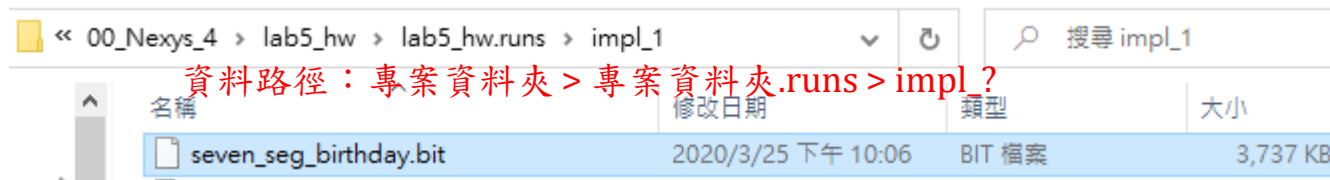
```
D:\academic\LPAP\DD_lab5\cnt0_7>vvp test
VCD info: dumpfile Lab5.fsdb opened for output.
2500005 birth = 9
5000005 birth = 7
7500005 birth = 6
10000005 birth = 0
12500005 birth = 3
15000005 birth = 1
17500005 birth = 9
20000005 birth = 1
22500005 birth = 9
25000005 birth = 7
27500005 birth = 6
30000005 birth = 0
32500005 birth = 3
35000005 birth = 1
37500005 birth = 9
40000005 birth = 1
42500005 birth = 9
45000005 birth = 7
47500005 birth = 6
50000005 birth = 0
```

❖ 將循序輸出 0~7 與生日碼顯示器結合

➤ 以 sw14 為開關，為 0 時最右邊的七段顯示器會依序顯示 0~7，為 1 時則會依序顯示出自己的生日 (60%)

★ Demo 時需準備可證明自己生日之證件！

★ 當日 demo 請攜帶 usb，儲存你在 vivado 生成的 .bit 檔，現場直接燒錄即可，請不要在現場 generate bitstream



❖ 於回饋單上描述比較兩種方法在開發版上合成的差異 (20%)

附錄

- ❖ 卡諾圖是一種以圖形化的方式來進行布林代數化簡的表示圖。
- ❖ 由於卡諾圖本身亦記錄布林函數的數值，因此我們也可以視其為一種特別版本的真值表

❖ 範例

a/b	b	b'
a	1	1
a'	1	0

紅色圈選處所代表的函數為 a
(因為 b 的數值不會影響結果)

綠色圈選處所代表的函數為 b
(因為 a 的數值不會影響結果)

故化簡後布林函數為 $F = a + b$

Code Converter

❖ Code Converter Table Example

count[2:0]	birthday[3:0]
0(000)	1
1(001)	9
2(010)	7
3(011)	6
4(100)	0
5(101)	3
6(110)	1
7(111)	9

Birthday [3:0]	a	b	c	d	e	f	g
0 (0000)	0	0	0	0	0	0	1
1 (0001)	1	0	0	1	1	1	1
2 (0010)	0	0	1	0	0	1	0
3 (0011)	0	0	0	0	1	1	0
4 (0100)	1	0	0	1	1	0	0
5 (0101)	0	1	0	0	1	0	0
6 (0110)	0	1	0	0	0	0	0
7 (0111)	0	0	0	1	1	0	1
8 (1000)	0	0	0	0	0	0	0
9 (1001)	0	0	0	0	1	0	0

