Lab 4

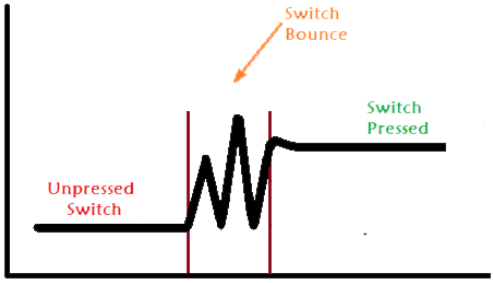
學號: 108071003 姓名: 李彥璋

1. 實作過程

- **debounce**

需要 debounce 的原因：

操作按鈕時，理想狀況下應該是【按下 = 1 && 放開 = 0】，但因為每個按鈕都有金屬彈簧，所以在訊號穩定前會產生我們不想要的 0、1 震盪 (如圖)。



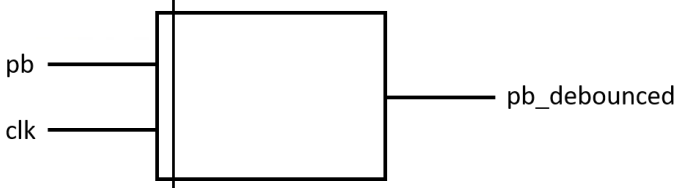
debounce 實作：

- pb -> debounce 前的 input 訊號。
- clk -> input clock。
- pb_debounced -> debounce 後的 output 訊號。

以一個 4bits 的 shift_reg 當作過濾器，每次往左 shift 一位並加入 pb 訊號。當 shift_reg 的每個 bit 都等於 1 時，代表訊號已經穩定，輸出 pb_debounced 為 1；否則為 0。

```
module debounce (pb_debounced, pb, clk);

  output pb_debounced;
  input  pb;
  input  clk;

  reg [3:0] shift_reg;

  always @(posedge clk)
    begin
      shift_reg[3:1] <= shift_reg[2:0];
      shift_reg[0] <= pb;
    end

  assign pb_debounced = ((shift_reg == 4'b1111) ? 1'b1 : 1'b0);

endmodule
```

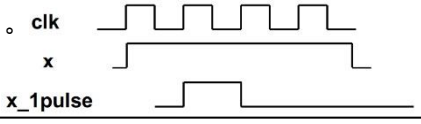
debounce block diagram



- **onepulse**

需要 onepulse 的原因：

人為按按鈕，每次按的時長可能不盡相同，我們希望不論按得多久，都能產生一個 one cycle pulse (也就是一個 cycle 的 1)，如圖中的 x_1pulse。



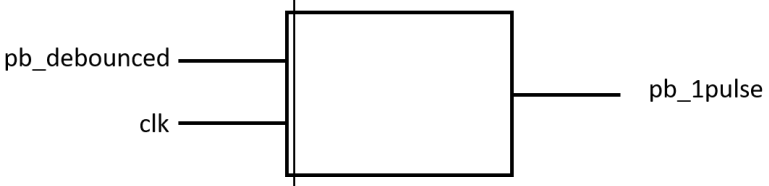
onepulse 實作：

- ✧ clk -> input clock。
- ✧ pb_debounced -> debounce 後的 input 訊號。
- ✧ pb_1pulse -> onepulse 後的 output 訊號。

以一個 pb_debounced_delay 將 pb_debounced 延遲一個 cycle，那個 cycle 正好會是 pb_debounced = 1 ＆＆ pb_debounced_delay = 0 的時候，我們將此設定為 1，輸出 pb_1pulse 訊號；否則為 0。

```
module onepulse (pb_debounced, clk, pb_1pulse);
  input  pb_debounced;
  input  clk;
  output  pb_1pulse;
  reg  pb_1pulse;
  reg  pb_debounced_delay;

  always @(posedge clk) begin
    if (pb_debounced == 1'b1 & pb_debounced_delay == 1'b0)
      pb_1pulse <= 1'b1;
    else
      pb_1pulse <= 1'b0;

    pb_debounced_delay <= pb_debounced;
  end
endmodule
```

onepulse block diagram



◆ **lab4_1**

題目描述：

lab4_1 要實作一個 2 bits BCD up/down counter，包含暫停、加速、減速等功能。

題目整理：

- ✧ 按下 reset，回到初始狀態，【BCD1、BCD0】顯示【0、0】並為暫停狀態。
- ✧ 按下 en，控制 counting 的暫停＆繼續。
- ✧ 長按 dir，counting down；否則 counting up。若為暫停狀態，dir 無效。【BCD2】顯示 arrow down 或 arrow up。
- ✧ 按下 speed_up 或 speed_down，counting 加速或減速。0 – 2 的速度等級（彼此差 2 倍）。【BCD3】顯示速度等級。
- ✧ 數到 99，led0 亮燈；數到 00，led1 亮燈。

實作：

- ✧ **debounce and onepulse –**

  將各個按鈕進行 debounce and onepulse，方法已於上方說明。

  值得注意的一點是，因為 dir 這個按鈕是「長按」發生作用，所以並不需要 onepulse 的步驟。

✧ **7 segment 顯示 –**

以 frequency 的頻率輪流切換 4 個 7 segment 的 value，frequency 需要有一定速度，以達到視覺暫留的效果（看起來同時）。

以另一個 always block 定義當 value 為多少時，7 segment 該顯示什麼在畫面上。

✧ **按鈕狀態 always block –**

```
always@(posedge clk or posedge rst) begin
  if(rst == 1'b1) begin
    resume <= 1'b0;
    speed <= 2'd0;
  end else begin
    if(pulse_en == 1'b1) resume <= ~resume;
    else resume <= resume;

    if(pulse_speed_up == 1'b1) begin
      if(speed == 2'd2) speed <= 2'd2;
      else speed <= speed + 2'd1;
    end else if(pulse_speed_down == 1'b1) begin
      if(speed == 2'd0) speed <= 2'd0;
      else speed <= speed - 2'd1;
    end
  end
end
```

這個 always block 主要以其他變數紀錄按鈕按下的狀況。

(1). 按下 rst，各變數回到初始狀態。

(2). 按下 en，resume 變數 negate（暫停&繼續切換；否則維持原值。

(3). 按下 speed_up 或 down，speed 進行速度的控制，注意當達到最大/小值時，不能再加/減速。

✧ **決定 BCD 是多少 always block –**

pseudo code：

```
if rst :
    BCD0、BCD1、min、max = 0
else :
    if resume = 0 (means pause mode) :
        BCD0、BCD1 remain the same
    else :
        if up count 數到 99 或 down count 數到 00 :
            BCD0、BCD1 remain the same
        else :
            if 沒有長按 dir (means up counting) :
                if BCD0 != 9 :
                    BCD0 = BCD0 + 1
                    BCD1 remains the same
                else if BCD1 != 9 :
                    BCD0 = 0
                    BCD1 = BCD1 + 1
            else (means down counting) :
                同理。
```

設定 max、min：

分別為數到 98、01 時 = 1(這邊主要是為了讓顯示 99、00 時，led 能夠「同時」亮燈)。

這個 always block 值得注意的地方是，它的 clk 是根據 speed 決定的。因為 speed0、1、2 的速度都差 2 倍，所以我利用 speed 判斷 clock 應該選/2^23 or /2^24 or /2^25。

◆ **lab4_2**

題目描述：

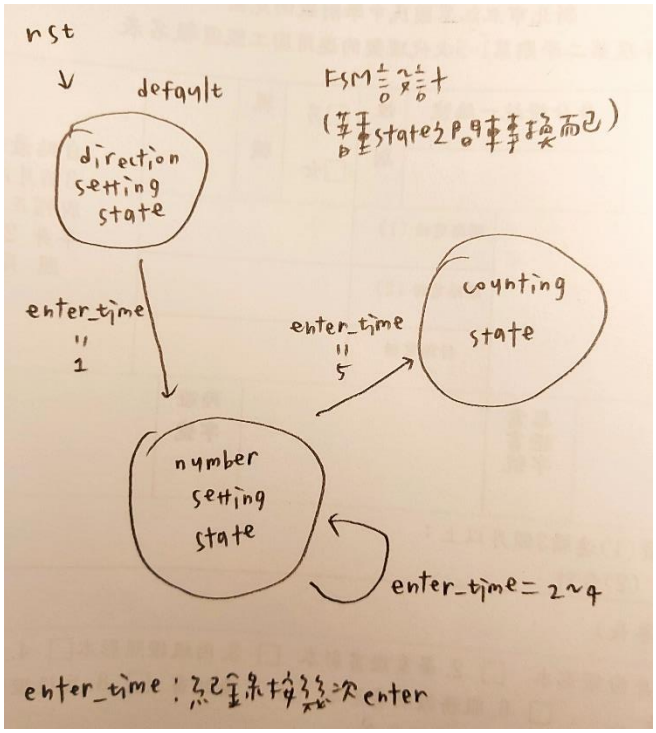
lab4_2 要實作一個正數、倒數計時器，包含暫停功能。

題目整理：

隨時按下 rst，回到 direction setting state。

1. **Direction setting state**
   - ✧ 按下 count_down 可控制計時器正數或倒數。
   - ✧ 按下 enter，換到 number setting state。
   - ✧ 7 segment 顯示【- - - -】。
   - ✧ Led0 亮燈表示倒數；否則暗燈。
2. **Number setting state**
   - ✧ 按下 input_number 可以設定每一位的數字（最大 1 5 9 9，超過歸 0）。
   - ✧ 按下 enter 可以換到設定下一位 或 切換到 counting state（最後一次 enter）。
   - ✧ 7 segment 初始顯示【0 0 0 0】，後來顯示使用者訂定的數字。
3. **Counting state**
   - ✧ en，可控制暫停＆繼續 counting。
   - ✧ 7 segment 顯示 counting 過程的數字。
   - ✧ 數到 goal or 0，停止。

FSM 設計：



實作：

- ✧ **debounce and onepulse** – 同 lab4_1。(lab4_2 沒有長按按鍵)
- ✧ **7 segment 顯示** – 同 lab4_1。
- ✧ **按鈕狀態 always block** –

  基本上這個部分也跟 lab4_1 雷同，都只是記錄變數上的差異而已。主要不一樣的地方是，因為 lab4_2 是有 state 狀態的，所以每次都會令 state <= state_next。

- ✧ **決定 BCD 是多少 always block (state block)** –

  1. **Clock 設定：**

     因為我有做 lab4_3，將在底下 lab4_3 時說明。

  2. **各個 state：**

     Direction setting ：

     如果 enter_time = 1，state_next = number setting state、BCD0~3 = 0。

     此外 state_next = direction setting state、BCD0~3 = dash。

     Number setting ：

     舉第一位為例，如果 enter_time 為 1 (第一位)，如果有按下 input_number，BCD3 就會往上加 (最高到 1，超過歸 0)，否則維持 BCD3。

     BCD0~2 則維持 0。

     State_next = number setting state。

     記得要用一個 record3 紀錄 BCD3 的值，因為我們需要以此當作 up counting 的 goal 或 down counting 的 start point。

     二、三、四位同理。

     Counting ：

     state_next 總是 = counting。(rst 的狀況寫在另一個 always block)。

     如果不是 pause mode，

     如果是 count up，

     如果到達 goal，停止。如果還沒到 goal，就照著 lab4_1 BCD counter 的方式設定 BCD0~3。

     Counter down 同理。

     如果是 pause mode，BCD0~3 = BCD0~3。

◆ **lab4_3**

題目描述：

Lab4_2 的計數器跳一次是 0.1second。

實作：

將 lab4_2 **決定 BCD 是多少 always block (state block) 的 clock 設定成 0.1 second**。

1 / 0.1 second = 10，所以我們希望得到 10 HZ 的 clock。又 FPGA 板子預設是 100M HZ，所以我們要除以 10000000，才能得到 10 HZ。

```
module clock_divider(
  input clk,
  output wire clk_div
  );

  reg [23:0] cnt;
  parameter n = 24;

  always @(posedge clk) begin
    if(cnt < 10000000-1'b1) begin
      cnt <= cnt + 1'b1;
    end else begin
      cnt <= 24'd0;
    end
  end
  assign clk_div = cnt[n-1];
endmodule
```

因為 10000000 是 24bits，所以設定一個 24bits 的變數從 0 數到 9999999，而後歸零。Output clock 便是最前面的那個位數，因為一開始的一半為 0；後面的一半為 1。

2. 學到的東西與遇到的困難

(1)．Lab4_2 時無法產生 bit 檔案，我不知道有什麼問題。所以一直重開 project，然後一直不行，後來發現是 constrain 沒有寫好。

(2)．Lab4_1 要設定三種速度差 2 倍，我一開始以為可以用一個 global 變數紀錄，譬如說數 2、4、8 來當成速度的依據。後來發現好像不行，所以想到用不同 clock 的方法。

(3)．這次的作業有感覺比之前難很多（差點要放棄了 TT），我覺得一開始寫 Lab4_1 的時候，因為還有點不清楚這次的程式架構是怎樣（因為加了很多新的元素），所以花了非常多時間（超多），成功弄明白後，lab4_2 就比較順利了。

(4)．學到蠻重要的東西是 Lab4_1 的長按按鍵 -> 不用 onepulse，以及 Lab4_2 的 input_number -> 這個按鈕 onepulse 的 clock 要設定成跟 BCD change 的 clock 是相同的，否則可能會有按一下跳很多數字的情況。

3. 想對老師或助教說的話

(1)．謝謝老師＆助教！

(2)．講一些廢話：那天看過老師給的 report 範例，覺得自己之前寫的 report 真的是隨(雜)心(亂)所(無)欲(章)，所以這次就很認真寫 A_A，雖然到後面也是亂亂的= =。