邏輯設計實驗 Lab5 結報

105060012 張育菘

1 Construct a 30-second down counter with pause function. When the counter goes to 0, all the LEDs will be lighted up. You can use one push button for reset and one other for pause/start function.

1.1 Implement a periodic 30-second down counter and demo with the FPGA board.

1.2 Implement Prelab

1.3 and demo with the FPGA board. 1.3 Combine 1.2 and 1.3 to finish the experiment.

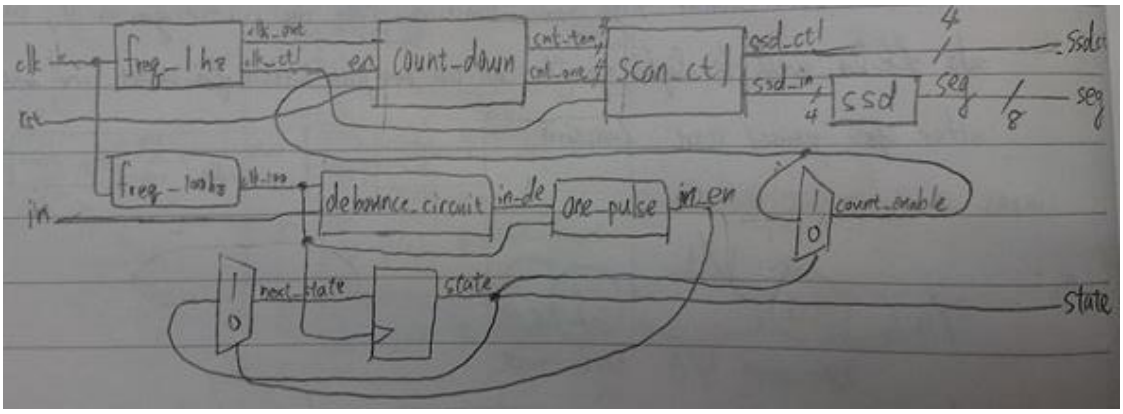
clk	state	rst	in
W5	U16	V17	W19

Design Specification

Input：clk,rst,in;

Output：ssd_ctl[3:0],seg[7:0],state;

block diagram :



Design Implementation

Logic function :

1. display：

   此 display module，專門拿來呼叫其他小 moudule 的。

   還有做為 finite state machine，以接出 count_enable 至 count_down 決定是否能開始執行 count_down 的關鍵。

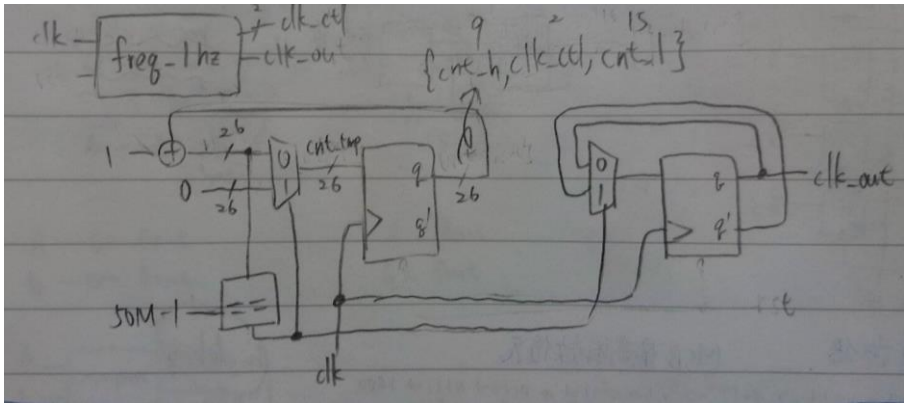
   這裡的 clk 我是接 100hz，因為 clk 在 0→1 時讀 one_pulse(其 clk 為 100hz)處理過的值，因此 display 的 clk 不能太快。

   - fsm 不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→fsm 不做事。

2. freq_1hz：

   此為除頻的 module，分別輸出 1hz 的 clk_out 控制 count_down 以及控制 scan_ctl 的 clk_ctl。

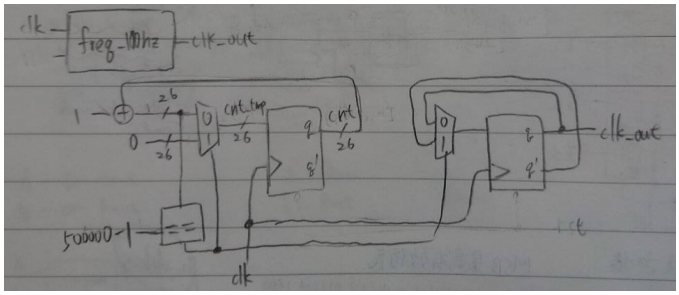
   - 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_1hz 輸出的 clk 持續為 0→freq_1hz 不做事。



3. freq_100hz：

   此為除頻的 module，輸出 100hz 的 clk_out 當作 debounce_circuit、one_pulse 以及 fsm 的 clk(在 display 內)。

   - 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk 持續為 0→freq_100hz 不做事。

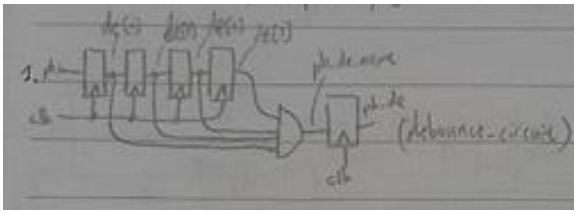


4. debounce_circuit：

   此為**除按鈕雜訊**的 module，原理是，當 debounce_circuit 收到連續 4 個 1 時才產生一個 0→1 的訊號，這樣可以避免收到前段起伏不定的雜訊，達到除雜訊的效果。

   且要接 100hz 的 clk，因為若接此的 clk 頻率太快就無法達到除雜訊的效果，太慢可能會超過按鈕維持穩定值的時間，因此我選擇 100hz 作為 debounce_circuit 的 clk。

   - 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ debounce_circuit 不做事。

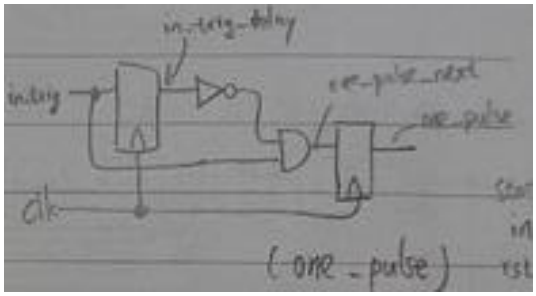


5. one_pulse：

   輸入訊號經過 debounce_circuit 處理後，其週期可能會超過 1 個 clk 的時間，因此需要此 module 使訊號的週期為 1 個 clk 的時間。

   這裡的 clk 我也是接與 debounce_circuit 同的 100hz。

   - 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ one_pulse 不做事。



6. count_down：

   這是同時數兩個變數，個位數 cnt_one 跟十位數 cnt_ten 進行往下數。當 cnt_one=0 時把 cnt_ten 的數值減一以及 cnt_one 的數值變成 9；當 cnt_one 跟 cnt_ten 都為 0 時，讓兩個數都維持 0。而初始化(rst=0)就是把數設為 30(cnt_ten=3 & cnt_one=0)。

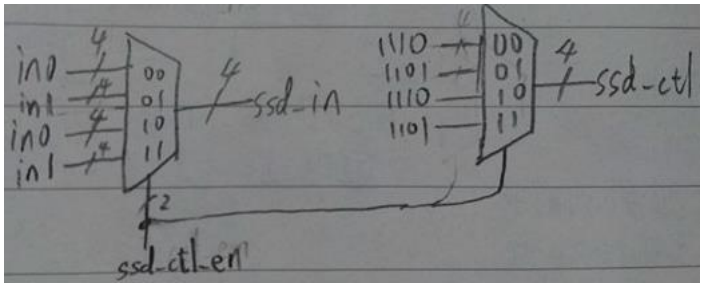
   這裡還接一個 en，作為是否能往下數的判斷值，若 en=1，表示可以往下數；若 en=0，則不能繼續往下數。

7. scan_ctl：

   此 module 是讓七段顯示器同時顯示不同位數的 module。

   ssd_ctl_en(from freq_1hz 的 clk_ctl[2:0])的用意為顯示兩個不同的數，因為七

段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到 1hz 的 ssd_ctl_en 產生同時顯示兩個數的錯覺。



8. ssd：

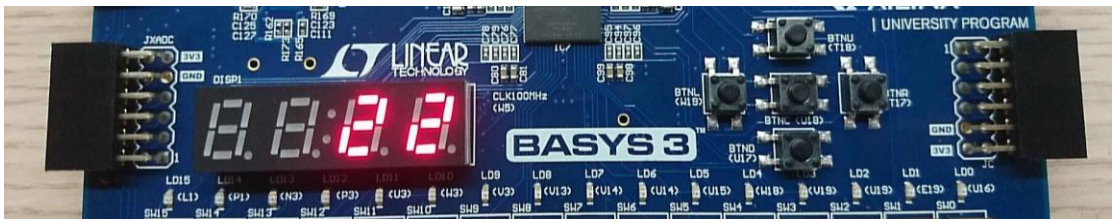
當 bcd 為 4'd0: segs = 8'b00000011; //七段顯示器顯示 0
當 bcd 為 4'd1: segs = 8'b10011111; //七段顯示器顯示 1
當 bcd 為 4'd2: segs = 8'b00100101; //七段顯示器顯示 2
當 bcd 為 4'd3: segs = 8'b00001101; //七段顯示器顯示 3
當 bcd 為 4'd4: segs = 8'b10011001; //七段顯示器顯示 4
當 bcd 為 4'd5: segs = 8'b01001001; //七段顯示器顯示 5
當 bcd 為 4'd6: segs = 8'b01000001; //七段顯示器顯示 6
當 bcd 為 4'd7: segs = 8'b00011111; //七段顯示器顯示 7
當 bcd 為 4'd8: segs = 8'b00000001; //七段顯示器顯示 8
當 bcd 為 4'd9: segs = 8'b00001001; //七段顯示器顯示 9
default: segs = 8'b00000000; //七段顯示器顯示 8.

Result

1. 圖一：當 U16 亮，表示 state=1→en=1→開始往下數。
2. 圖二：當 U16 暗，表示 state=0→en=0→停止往下數。
3. 圖三：當按下 V17，表示 rst=1→數字重新回至 30，且停止往下數。

圖一

圖二



圖三

Discussion

1. 這個實驗最重要的要注意 rst 接的 module，因為這裡的 rst 是希望 count_down 重新開始數。
2. count_down 判斷 rst 要在 rst 從 0→1 時判斷，因為 rst 是一個按鈕，按下的過程是 0→1，且要在 rst==1 時做出重置，這是跟以往實驗較不同之處。
3. 處理按鈕雜訊要使用 debounce_circuit，原理在上述已有做說明了。
4. 若要使多個週期的訊號簡化成週期為 1 個 clk 則要使用 one_pulse 去做處理。

2.The same function as Exp. 1. Instead of using two push buttons for reset/pause/start, try to use just one push button to finish the design. (Hint: You can press the push button longer to represent the reset)

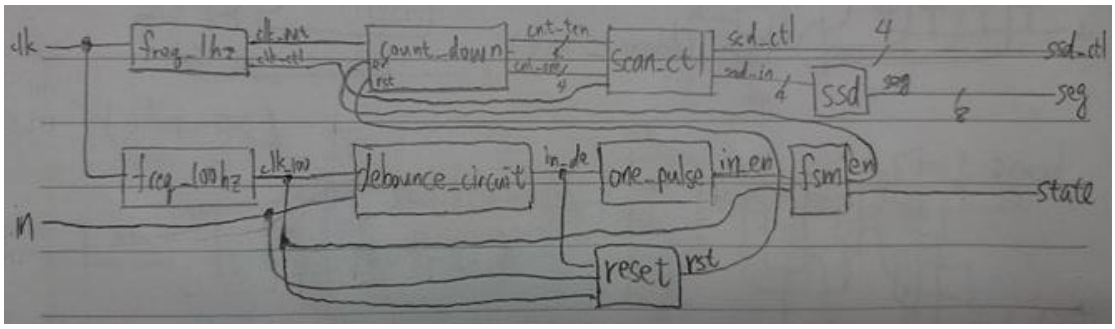
clk	state	in
W5	U16	W19

Design Specification

Input：clk,in;

Output：ssd_ctl[3:0],seg[7:0],state;

block diagram :



Design Implementation

Logic function :

1. display：

   此 display module，專門拿來呼叫其他小 moudule 的。

2. fsm：

   此 module 做為 finite state machine，以接出 count_enable 至 count_down 決定是否能開始執行 count_down 的關鍵。

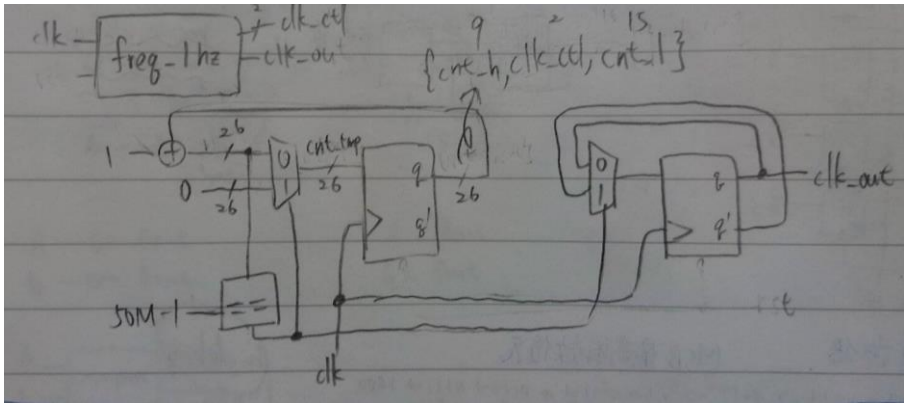
這裡的 clk 我是接 100hz，因為 clk 在 0→1 時讀 one_pulse(其 clk 為 100hz)處理過的值，因此 display 的 clk 不能太快。

- fsm 不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→fsm 不做事。

3. freq_1hz：

此為除頻的 module，分別輸出 1hz 的 clk_out 控制 count_down 以及控制 scan_ctl 的 clk_ctl。

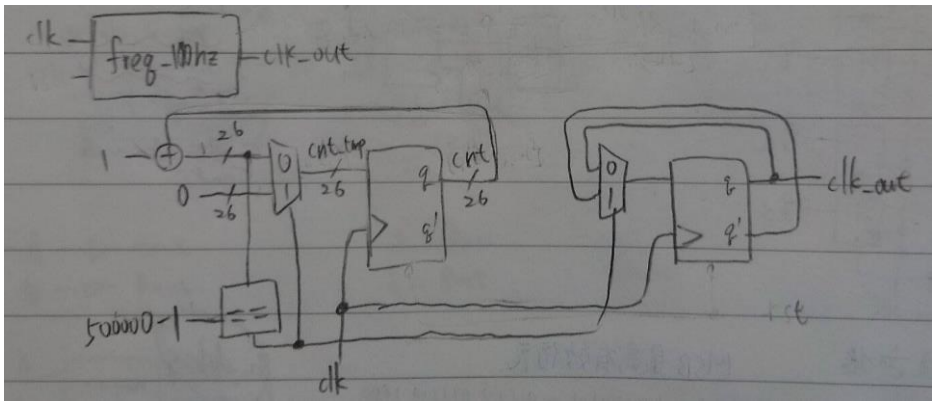
- 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_1hz 輸出的 clk 持續為 0→freq_1hz 不做事。



4. freq_100hz：

此為除頻的 module，輸出 100hz 的 clk_out 當作 debounce_circuit、one_pulse 以及 fsm 的 clk(在 display 內)。

- 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk 持續為 0→freq_100hz 不做事。

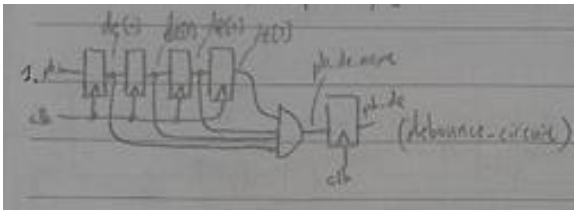


5. debounce_circuit：

此為**除按鈕雜訊**的 module，且要接 100hz 的 clk，因為若接此的 clk 頻率太快就無法達到除雜訊的效果，太慢可能會超過按鈕維持穩定值的時間，因此我

選擇 100hz 作為 debounce_circuit 的 clk。

- 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ debounce_circuit 不做事。

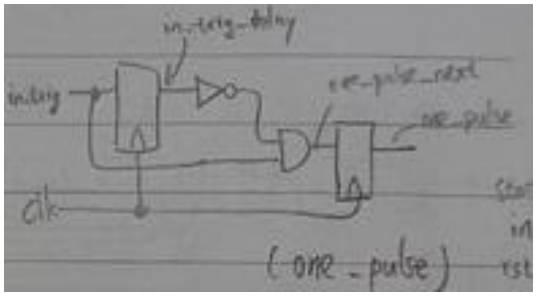


6. one_pulse：

   輸入訊號經過 debounce_circuit 處理後，其週期可能會超過 1 個 clk 的時間，因此需要此 module 使訊號的週期為 1 個 clk 的時間。

   這裡的 clk 我也是接與 debounce_circuit 同的 100hz。

   - 這裡不能接 display 的 input rst，因為那裡的 rst 是希望 count_down 重新開始數。當 rst=0 時，freq_100hz 輸出的 clk_100 持續為 0→ one_pulse 不做事。



7. count_down：

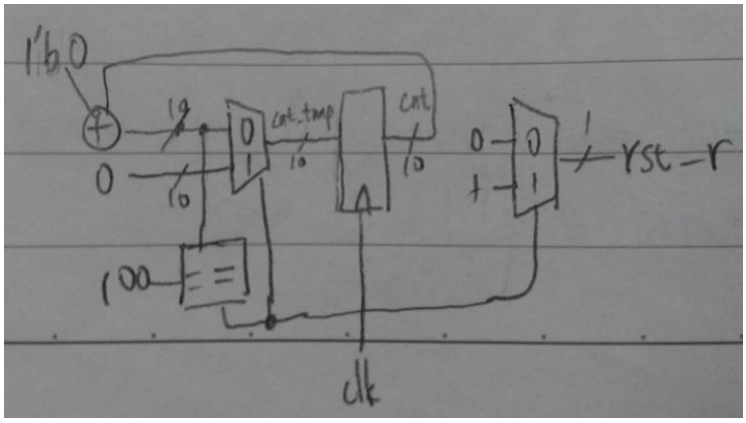
   這是同時數兩個變數，個位數 cnt_one 跟十位數 cnt_ten 進行往下數。當 cnt_one=0 時把 cnt_ten 的數值減一以及 cnt_one 的數值變成 9；當 cnt_one 跟 cnt_ten 都為 0 時，讓兩個數都維持 0。而初始化(rst=0)就是把數設為 30(cnt_ten=3 & cnt_one=0)。

   這裡還接一個 en，作為是否能往下數的判斷值，若 en=1，表示可以往下數；若 en=0，則不能繼續往下數。

8. reset：

   此 module 功能是為產生 rst 訊號。

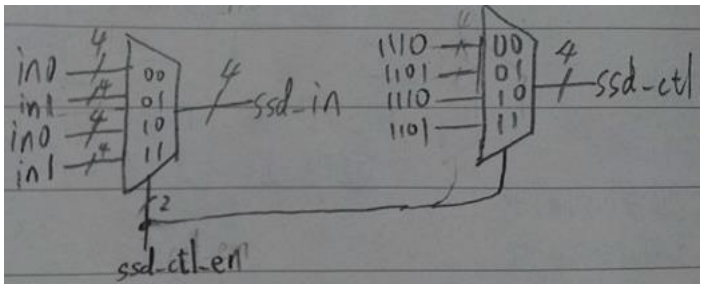
   而此 input 訊號 rst 來自只經過 debounce_circuit 處理過的輸入訊號 in_de，因為此週期可能為多個週期的 clk_100，因此可以拿來作為判斷長按的依據。



9. scan_ctl：

   此 module 是讓七段顯示器同時顯示不同位數的 module。

   ssd_ctl_en(from freq_1hz 的 clk_ctl[2:0])的用意為顯示兩個不同的數，因為七段顯示器一次只能顯示一種數次，因此需要一個頻率介於內建時間到 1hz 的 ssd_ctl_en 產生同時顯示兩個數的錯覺。



10. ssd：

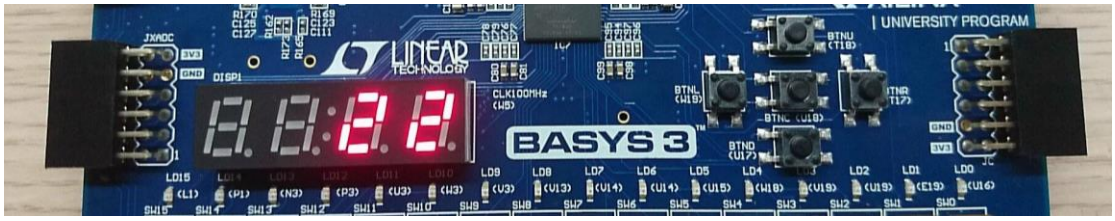
```
當 bcd 為 4'd0: segs = 8'b00000011;   //七段顯示器顯示 0
當 bcd 為 4'd1: segs = 8'b10011111;   //七段顯示器顯示 1
當 bcd 為 4'd2: segs = 8'b00100101;   //七段顯示器顯示 2
當 bcd 為 4'd3: segs = 8'b00001101;   //七段顯示器顯示 3
當 bcd 為 4'd4: segs = 8'b10011001;   //七段顯示器顯示 4
當 bcd 為 4'd5: segs = 8'b01001001;   //七段顯示器顯示 5
當 bcd 為 4'd6: segs = 8'b01000001;   //七段顯示器顯示 6
當 bcd 為 4'd7: segs = 8'b00011111;   //七段顯示器顯示 7
當 bcd 為 4'd8: segs = 8'b00000001;   //七段顯示器顯示 8
當 bcd 為 4'd9: segs = 8'b00001001;   //七段顯示器顯示 9
default: segs = 8'b00000000;          //七段顯示器顯示 8.
```

Result

1. 圖一：當 U16 亮，表示 state=1→en=1→開始往下數。
2. 圖二：當 U16 暗，表示 state=0→en=0→停止往下數。
3. 圖三：當長按(1 秒)W19，表示 rst=1→數字重新回至 30，且停止往下數。

圖一

圖二



圖三

Discussion

1. 這個實驗最重要的要注意 rst 接的 module，因為這裡的 rst 是希望 count_down 重新開始數。
2. count_down 判斷 rst 要在 rst 從 0→1 時判斷，因為 rst 是一個按鈕，按下的過程是 0→1，且要在 rst==1 時做出重置，這是跟以往實驗較不同之處。
3. 處理按鈕雜訊要使用 debounce_circuit，原理在上述已有做說明了。
4. 若要使多個週期的訊號簡化成週期為 1 個 clk 則要使用 one_pulse 去做處理。
5. 此多用 reset module 產生 rst 訊號，是用加法器的概念去達成長按能產生 rst 效果。

Conclusion：

這次的 lab 更深入帶我們了解到如何使用少許的按鈕達成多種功能，以及清楚了解使用 rst 的時機，雖然過程中挫折不斷，但是打完真的是成就感滿滿，真的很開心。