**Pre-lab1與Experiment1:**

Pre-lab1:

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 Write the spec (inputs, outputs, and function table) of the design.

1.2 Draw the related block/logic diagram.

1.3 Use a FSM to implement the function of pause/start function. Use one LED to

represent current state.

1.4 Use Verilog to implement 1.3 and verify the design with simulation results.

Experiment1:

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.

**a. Specification :**

Inputs:

pause\_start // 按鈕，按一下開始下數(start)，再按一下會暫停(pause)

reset // 按鈕，按一下會重置計數器

clk\_100mhz // FPGA內建的時脈，頻率為100 M Hz

Outputs:

cathode[7:0] // 控制一個七段顯示器中各個segment的亮暗

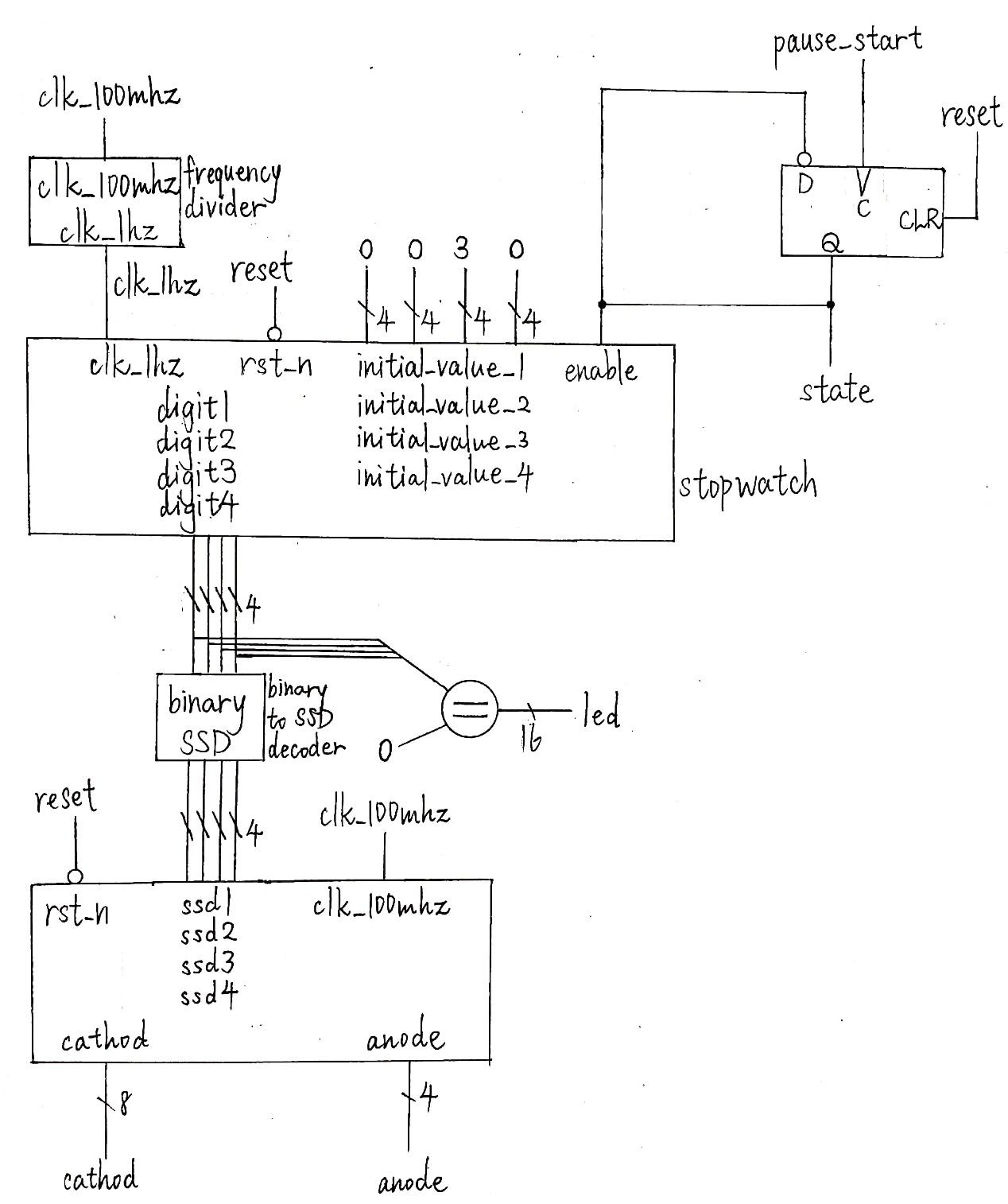
anode[3:0] // 控制要啟用哪一個七段顯示器

led[14:0] // 計數器數到零的時候，亮燈

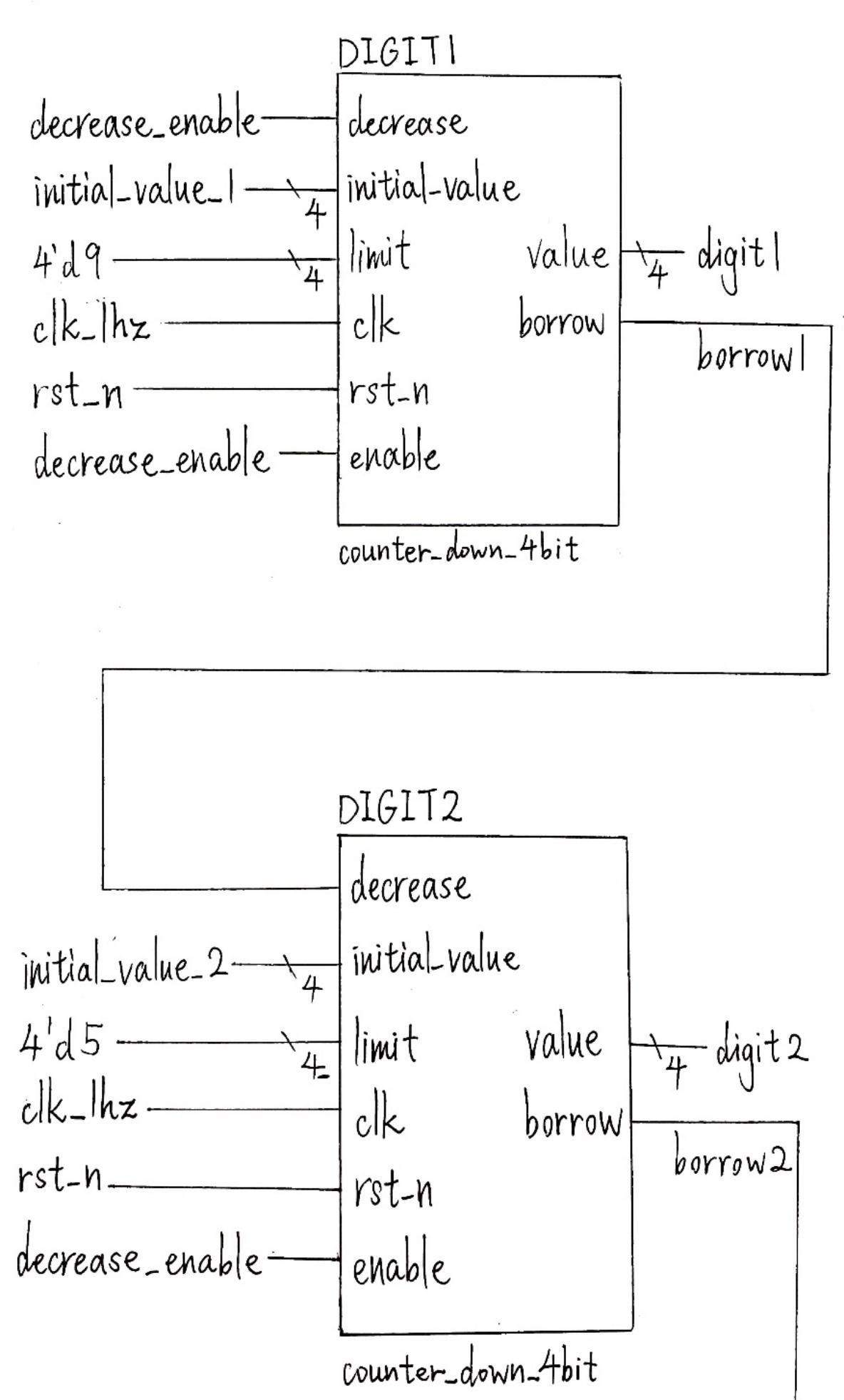
state // 顯示計數器當下的狀態是pause還是start

**b. Block Diagram :**

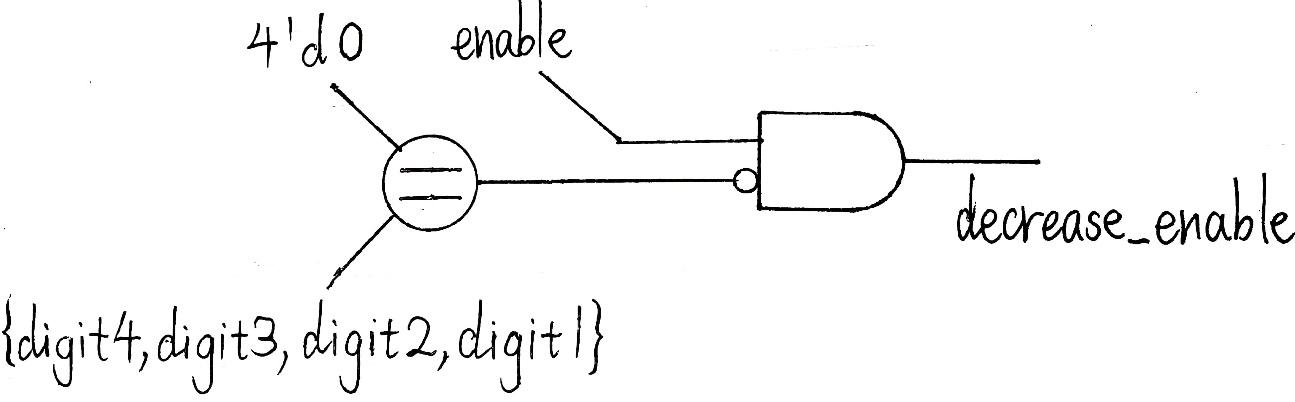
用除頻器製造出一赫茲的時脈。把一赫茲時脈連接到碼表，使碼表的數值每秒減少一。把碼表的二進位制數值以解碼器轉換成七段顯示器的樣式，然後再連接到七段顯示器的控制器。下圖右上角，用一個T Flip-flop來實作有限狀態機，用來控制碼表的狀態，判斷當下碼表應該要下數(start)還是不動(pause)。

****

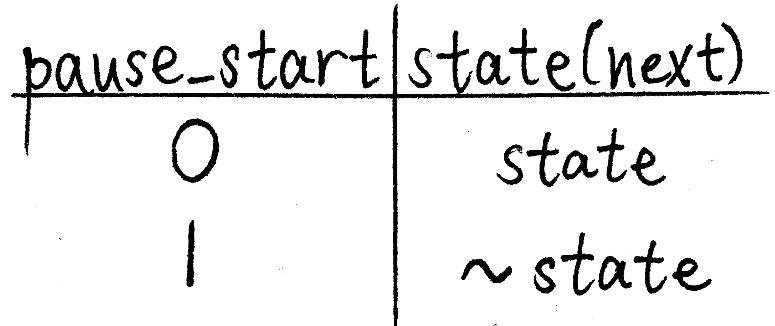
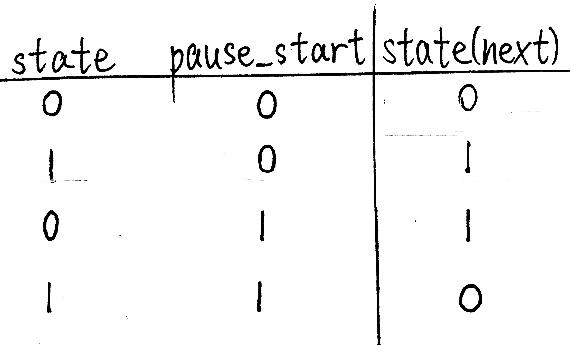
下圖是碼表內部的局部接線。碼表內有四個四位元的計數器，每個四位元計數器負責一個位數。第四、三位數(digit4, digit3)是分鐘的BCD，第二、一位數(digit2, digit1)是秒的BCD。

****

為了使碼表在數到零的時候自動停下來，碼表內的每一個四位元計數器的enable不可以與碼表外部的enable直接連接，而是應該要多判斷一下。如下圖，必須要確定這四個四位元計數器所代表的倒數時間不是零，才繼續下數。

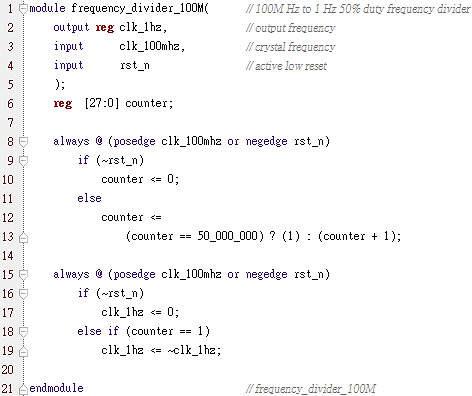
****

**c. FSM :**

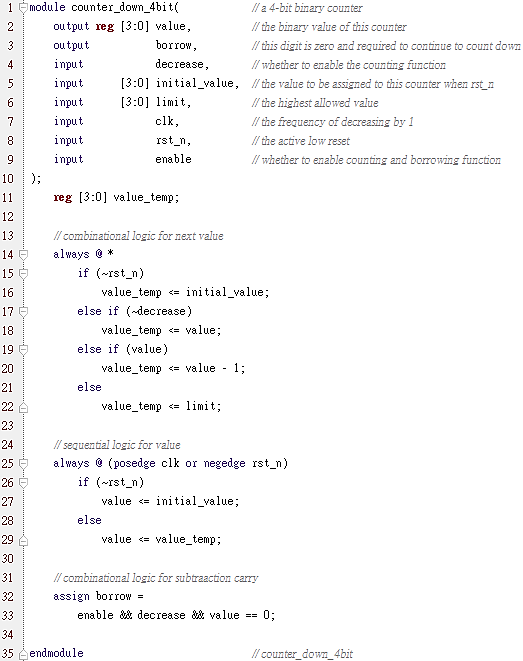
計數器有兩個狀態，S1是正在下數，S2是暫停。可以用一個位元來表示計數器當下的狀態state，當按鈕pause\_start被按下時，會切換計數器的狀態：

**d. Implement :**

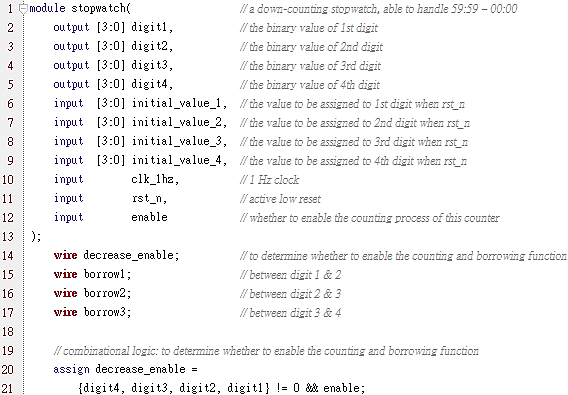
除頻器的作法跟上次的Lab一樣。使用一個計數器，每當數到50M的時候，就觸發T Flip-flop。如下圖，這樣就成功地製造出50% duty的一赫茲時脈了。



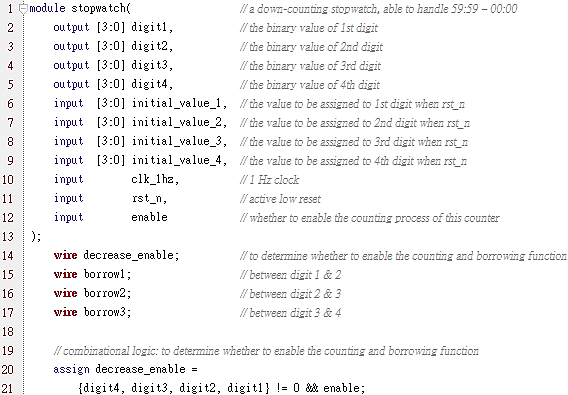
四位元下數計數器的硬體描述語言如下圖，跟上一次LAB的一模一樣。第14到22行判斷下一個狀態的數值：如果要重置，就把數值設定成初始值；否則，如果不要下數，就把保持相同的數值；否則，如果還沒有數到零，就把數值減少一；否則，把數值設定成最大的允許的數值。第25到29行的D Flip-flop用來儲存和更新數值。第32到33行用來判斷是否有需要向上一位數借位：如果要繼續數，但是當下的數值已經數到零了，就需要輸出借位的訊號給上一位數的四位元計數器。



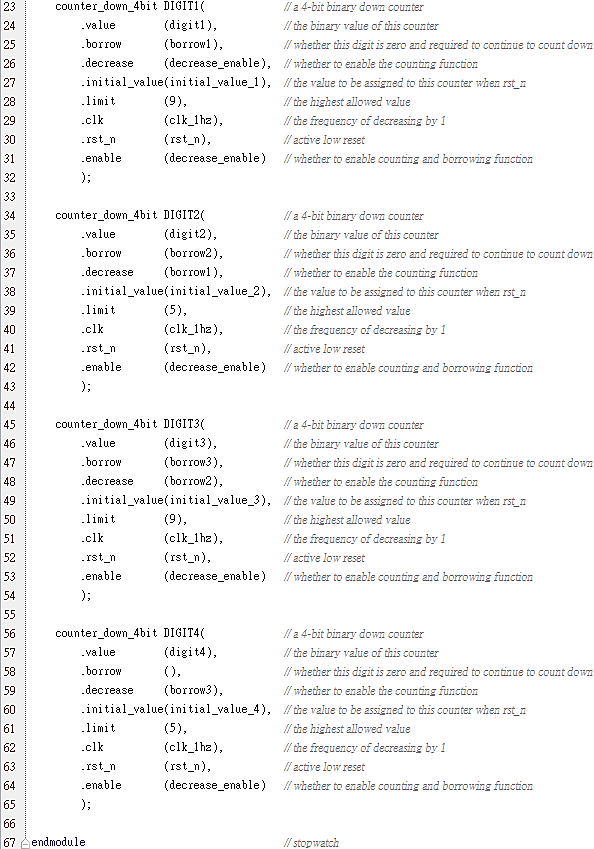
下圖是碼表的輸出、輸入與一些內部的線路名稱。



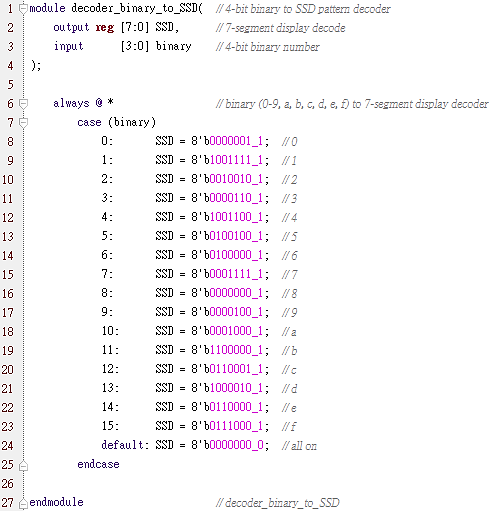
第20到21行用來判斷是否要讓碼表內的那四個四位元計數器繼續下數：如果碼表外部沒有enable訊號，或是這四個計數器所代表的數值已經數到零，就不要讓計數器繼續下數(decrease\_enable)。



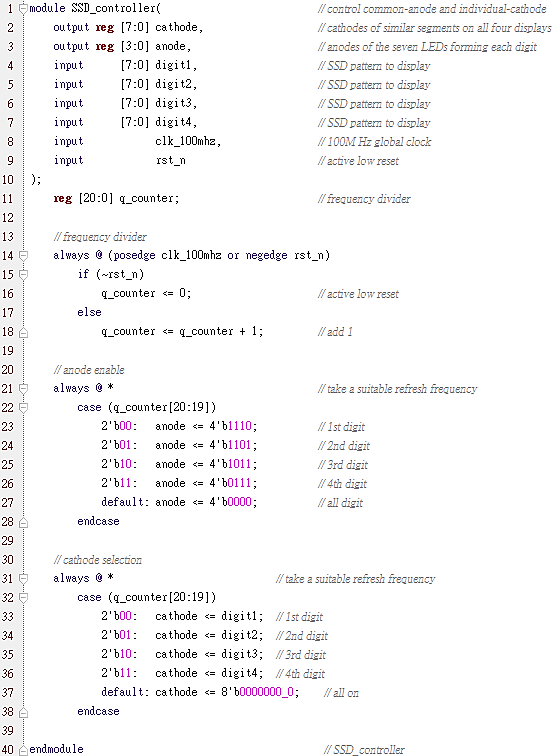
下圖是碼表內那四個四位元計數器的線路連接方法。這四個計數器的enable不是與碼表外部的enable直接連接，而是與判斷過當下數值是否已經下數到零的decrease\_enable連接。



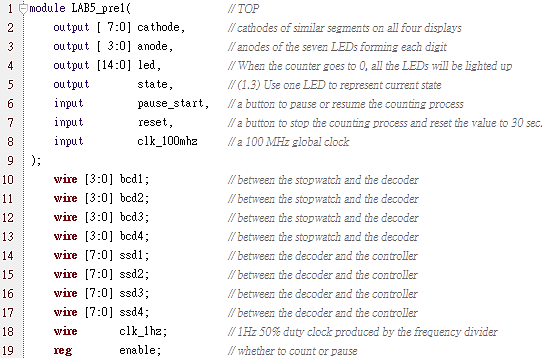
解碼器的硬體描述語言如下圖，跟上一次LAB的一模一樣。



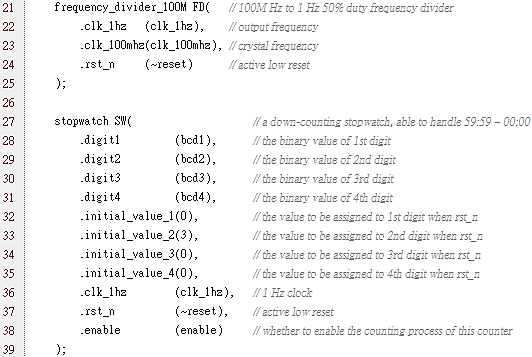
七段顯示器控制器的硬體描述語言如下，跟上一次LAB的一模一樣。第14到18行是除頻器，用來產生適當的頻率。雖然這四個七段顯示器是輪流顯示，並非同時，但是因為人眼的視覺暫留，看起來就像同時顯示了四個不同的數字。第21到28行用來控制要讓四個顯示器中的哪一個顯示器顯示數字，而第31到38行則控制要顯示的樣式。



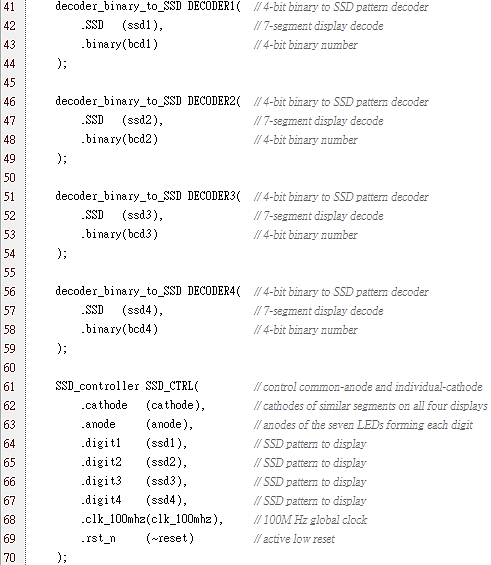
構造出最上層的有限狀態機並規劃好接線之後，用以上五個模組就足以用來完成這個LAB的所有題目。接下來的幾張圖片是依照Pre-lab1的要求所構造出來的硬體描述語言。輸出、輸入與一些內部接線的名稱如下：



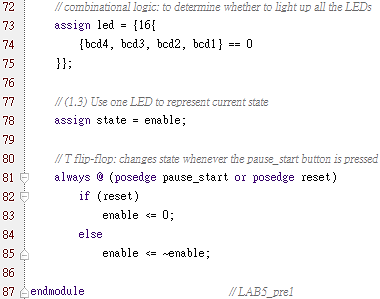
第21到39行，把除頻器所製造出來一赫茲時脈的連接到碼表。



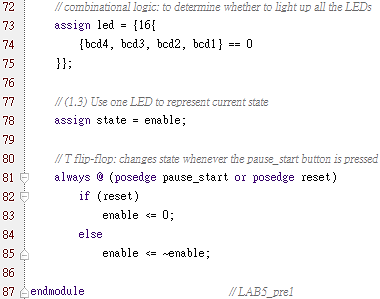
第41到70行，碼表的數值經過解碼器處理後，連接到七段顯示器控制器。



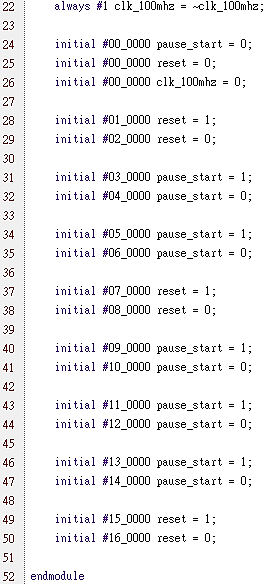
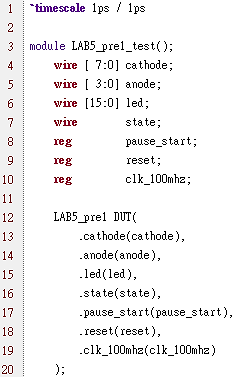
到目前為止，我們已經構造出能開始下數與暫停的30秒倒數馬錶了。應Pre-lab1的題目要求，第72到78行用來控制LED燈的明滅：當碼表數到零時，所有LED燈亮起；並使用一個LED燈來顯示當下碼表的狀態是正在下數還是暫停。



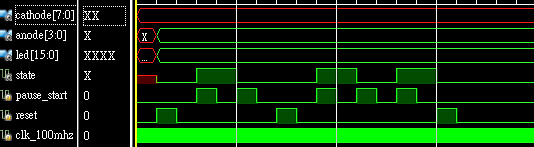
第80到85行是一個有限狀態機，用來控制碼表應該要下數還是暫停。



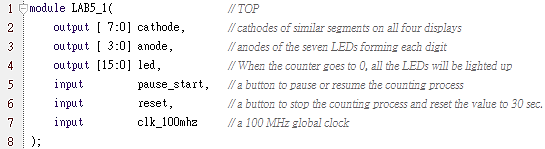
下圖是用來測試Pre-lab1的testbench。



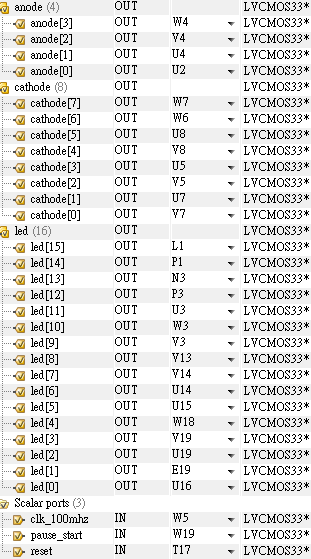
波型圖如下。cathode要等到一個一赫茲的時脈產生之後才會有值，然而，整個stimulus的過程只有數秒鐘，不足以使除頻器裡的計數器從一數到五千萬。因此，在整個stimulus的數秒鐘內，cathode都是紅的。特別留意當pause\_start或reset被按下的時候，state是如何改變的。



至於Experiment1，只要微調module LAB\_pre1的輸出，如下圖，並刪掉原本用來顯示狀態的第77與78行即可。



下圖是用來demo Pre-lab1和Experiment1的。



**e. Conclusion :**

這一題最讓我感到困難的是碼表的enable。一開始我直接把碼表內計數器的enable直接與碼表外部的enable連接，但是發現會有問題。因為當碼表內的計數器數到零秒的時候，所有計數器的enable都還是開啟的狀態。這會使得計數器輸出borrow的訊號，造成下一個時脈的數值變成9999。為了讓這四個計數器能在數到零秒的時候自動停下來，我先試著另外使用一個and gate，先判斷碼表外部有enable訊號，而且當下四個計數器的數值都不為零，才輸出訊號到第一個計數器的decrease。

**Experiment2:**

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)

**a. Specification :**

Inputs:

button // 短按切換狀態(下數/暫停)，長按重置

clk\_100mhz // FPGA內建的時脈，頻率為100 M Hz

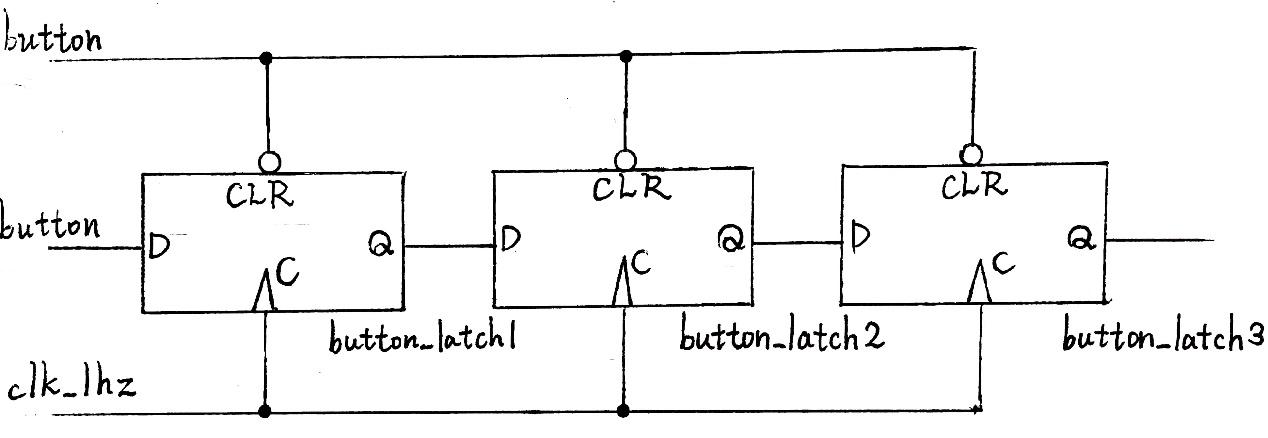
Outputs:

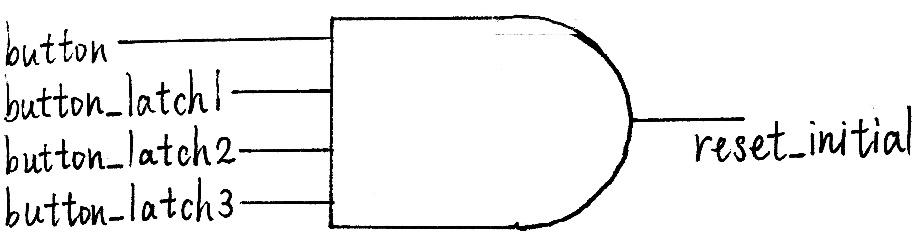
cathode[7:0] // 控制一個七段顯示器中各個segment的亮暗

anode[3:0] // 控制要啟用哪一個七段顯示器

led[15:0] // 計數器數到零的時候，亮燈

**b. Block Diagram :**

原本Experiment1中，所有reset訊號由獨立的一個按鈕控制。為了將reset功能按鈕與pause\_start合併成單一按鈕button，我們可以按久一點來表示reset信號。以Experiment1為基礎，只要再多使用三個D Flip-flop就可以偵測按鈕被長按。如下圖，如果三個D Flip-flop都是1，且button被按下，就輸出重置信號(reset\_initial)給其他模組。

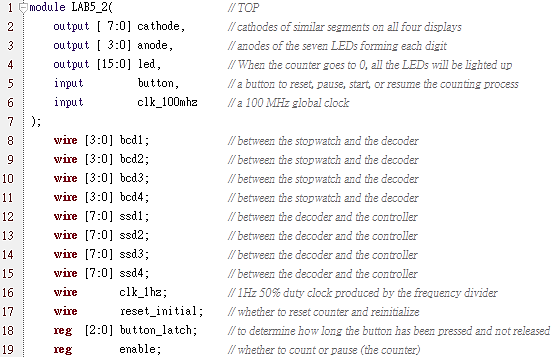


**c. FSM :**

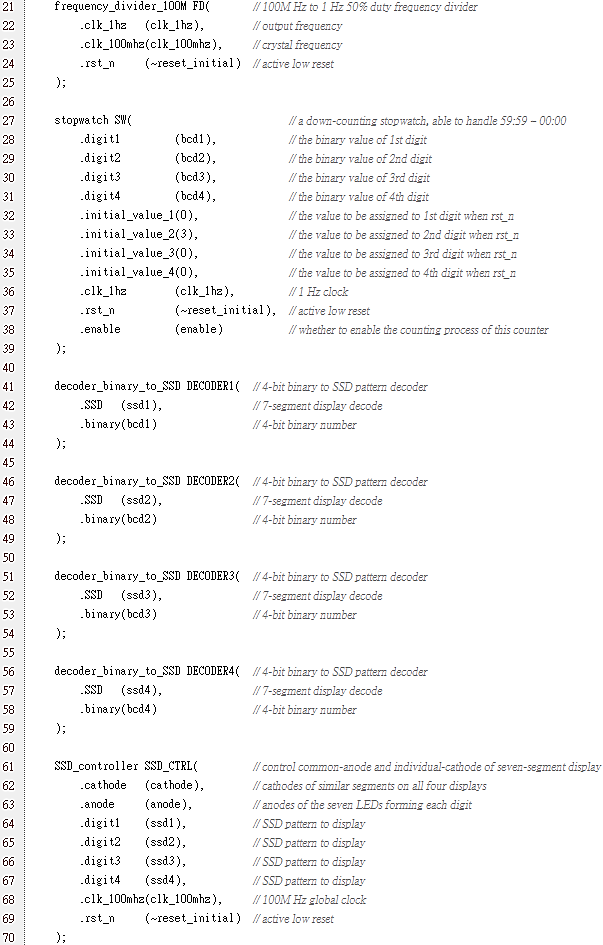
跟Experiment1一模一樣，用一個T Flip-flop來切換模式。

**d. Implement :**

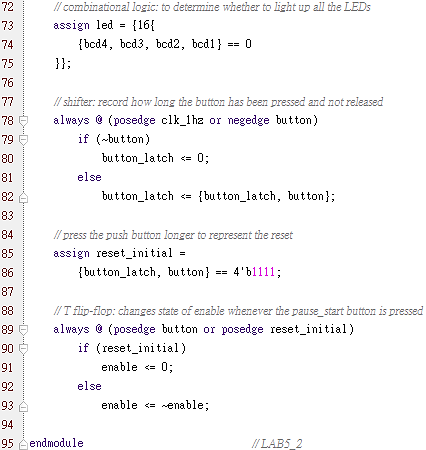
直接沿用Experiment1的module，僅需微調LAB5\_1即可。接下來的幾張圖片是微調後的LAB5\_2。把Experiment1的input pause\_start和input reset合併成單一按鈕input button；新增了一些接線：wire reset\_initial用來代替Experimemt1中原本的input reset，且多了reg button\_latch用來偵測按鈕長按，如下圖。



與其他模組的接線沒有太多更動，把原本的reset都改成reset\_initial即可。

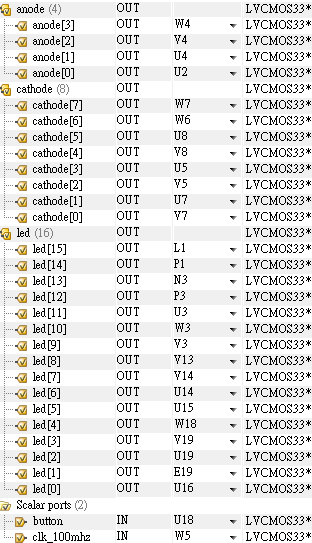


相較於Experiment1，Experiment2多了第77到86行：第77到82行是三個D Flip-flop，將clear與反相的button連接，用來記錄按鈕已被連續按下多久。第85到86行用來判斷是否已經被連續長按，如果按鈕被連續長按，就產生重置訊號(reset\_initial)。



這樣就構造完畢了，其他未提及的模組都跟上一題Experiment1的一模一樣。

下圖是用來demo Experiment2的。



**e. Conclusion :**

建議往後在著手建構Verilog前，先完整的瀏覽過所有題目，掌握該次LAB要實作的所有內容。畢竟這些LAB都是經過精心安排的，性質相近的題目通常會放在一起。如果一開始就把所有功能都考慮進來，就不會發生某個module在前幾題可堪使用，但不適用後面的題目的情況。這樣一來，就可以達到一個模組重複利用的優勢，節省下重複開發類似功能模組的時間與心力。

**Experiment3:**

(Bonus) Use two push buttons to control a multi-function stop timer (mode selection, reset, start, stop). The stop timer has two modes: 30-second/1-minute countdown. When being reset, the seven-segment display shows the digits 30/1:00. When the timer counts to 0, it will stop.

3.1 List the specification of the detector.

3.2 Design the FSM used in this design.

3.3 Draw the block diagram/logic schematic.

3.4 Implement the stop timer with FPGA demo board.

**a. Specification :**

Inputs:

button // 短按切換狀態(下數/暫停)，長按重置

mode\_select // 按一下會切換倒數30秒或一分鐘

clk\_100mhz // FPGA內建的時脈，頻率為100 M Hz

Outputs:

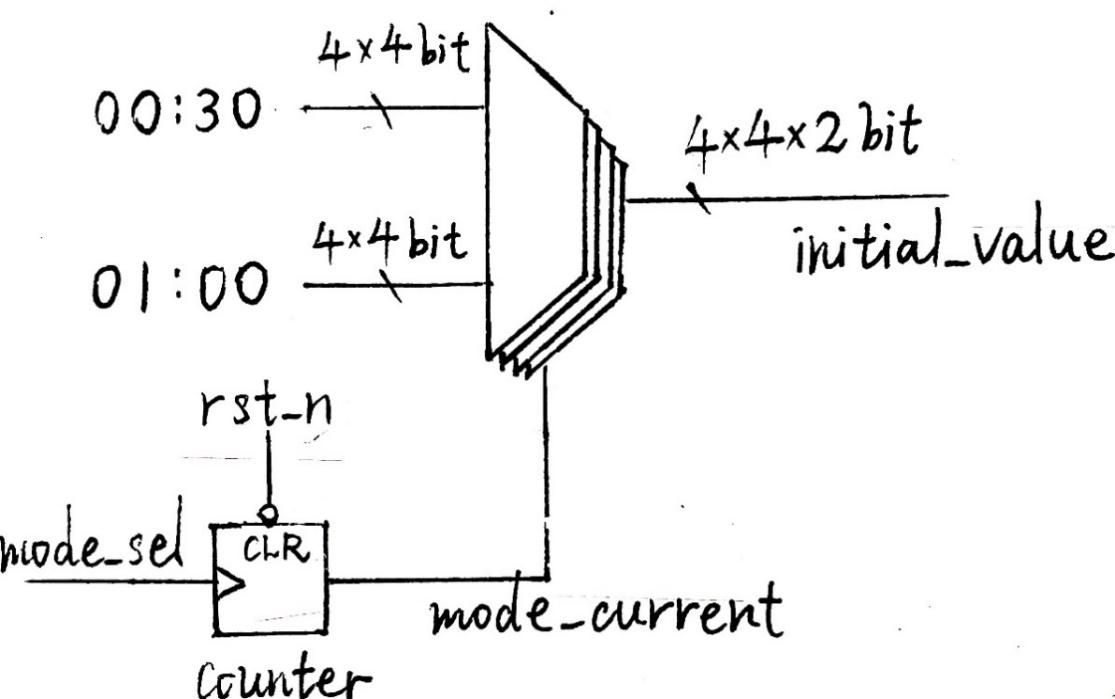
cathode[7:0] // 控制一個七段顯示器中各個segment的亮暗

anode[3:0] // 控制要啟用哪一個七段顯示器

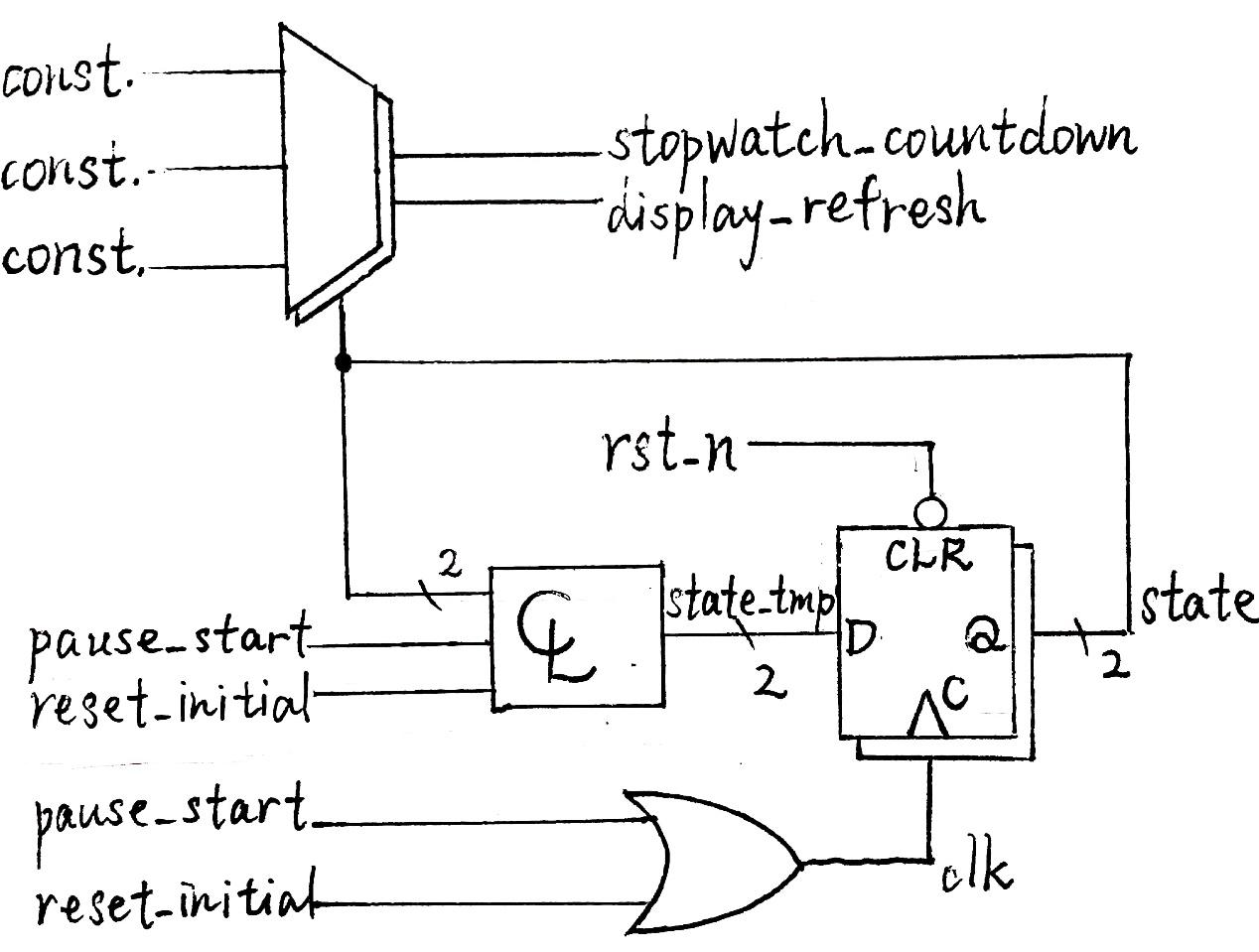
led[15:0] // 計數器數到零的時候，亮燈

**b. Block Diagram :**

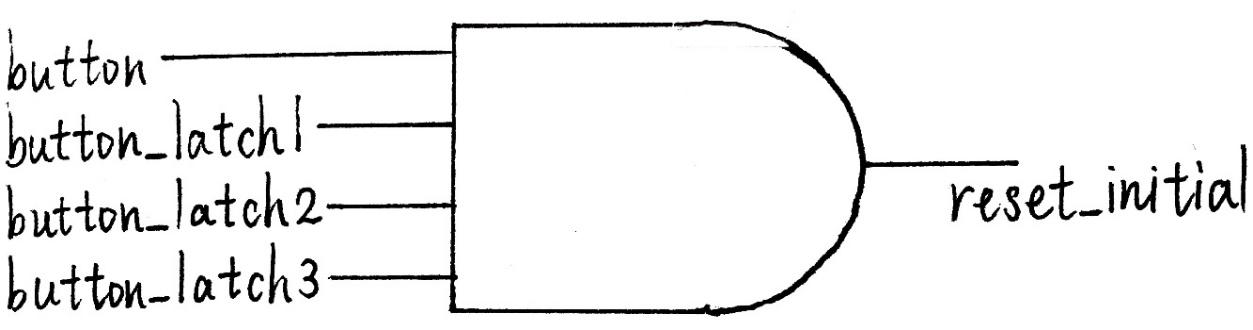
較Experiment2，多了mode\_selector和FSM，其餘一模一樣。mode\_selector由兩部分組成，先用一個計數器來儲存目前的模式(mode\_current)，再連接到一個MUX來產生該模式對應的初始數值(initial\_value)。如果未來有需要新增更多mode，只要增加計數器的位元數和MUX的選項即可。但其實因為這一題只有兩個mode，所以可以用一個T Flip-flop來代替計數器。

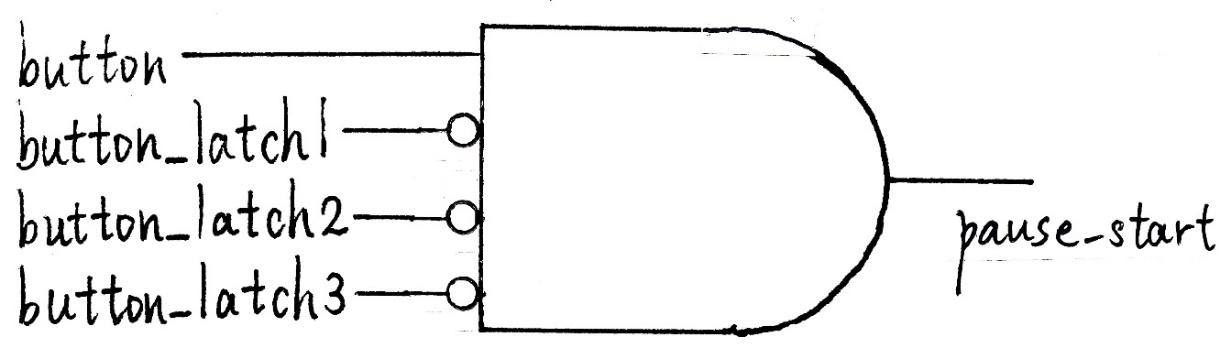


這題的FSM輸出只由當前的狀態所確定，也就是Moore machine。共三個狀態：正在倒數、凍結顯示器、重置，所以需要兩個D Flip-flop來儲存當下的狀態。當FSM收到切換pause或start的訊號，或是收到reset的信號，就更新狀態。所以D Flip-flop的C連接到pause\_start | reset\_initial。

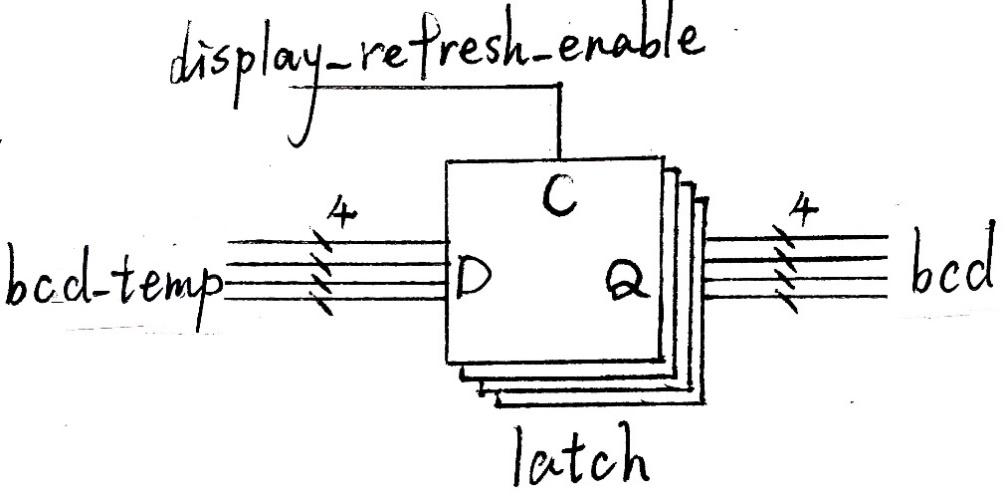


reset的訊號跟Experiment2一樣，條件是連續按下按鈕達三個時脈以上；而切換pause或start的訊號則是短按一下按鈕，如下圖。





為了使七段顯示器能凍結，可以在把碼表的數值傳送到解碼器之前就先檢查當下的狀態是否為凍結。如下圖，如果要凍結，則不更新latch的數值。



其他未提及的部分跟Experiment2一樣，直接沿用即可。

**c. FSM :**

state:

S0: RESETED

S1: COUNTING

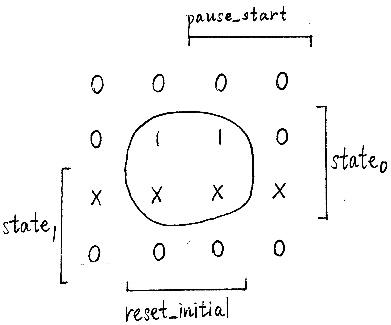
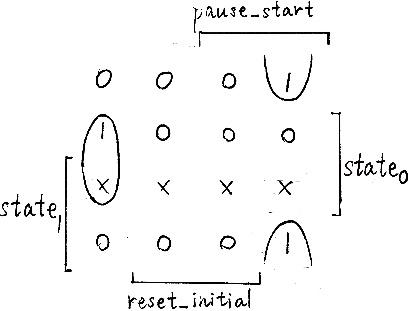
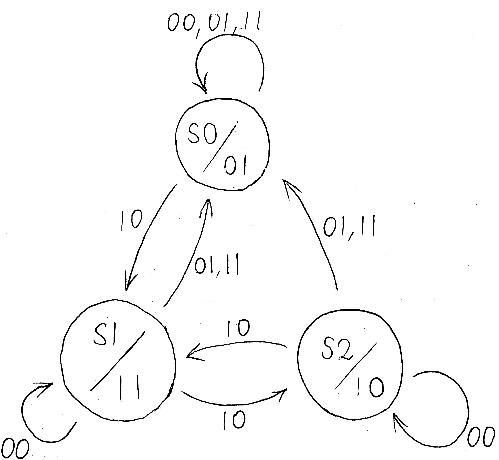
S2: FREEZED

input:

pause\_start/reset\_initial

output:

stopwatch\_countdown\_enable/display\_refresh\_enable



Next state equations/Excitation equations:

state0(t+1) = D0(state1, state0, pause\_start, reset\_initial)

= Σ(2, 4, 10)

= state0 × pause\_start’ × reset\_initial’ +

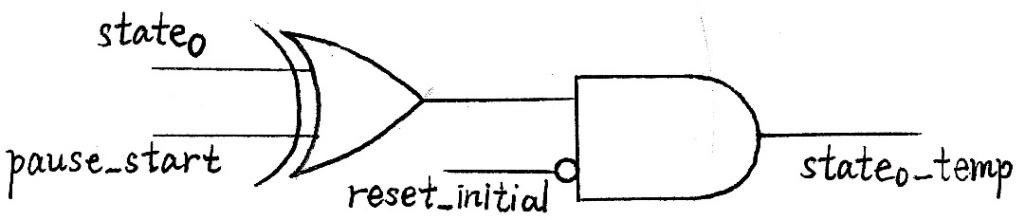
= state0’ × pause\_start × reset\_initial’

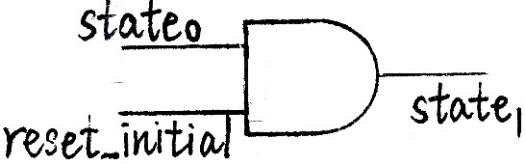
= (state0 ⊕ pause\_start) × reset\_initial’

state1(t+1) = D1(state1, state0, pause\_start, reset\_initial)

= Σ(6, 8)

= state0 × reset\_initial

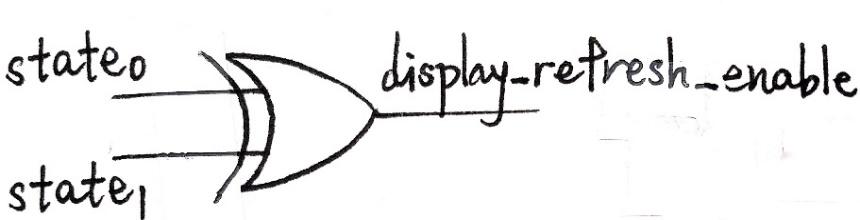




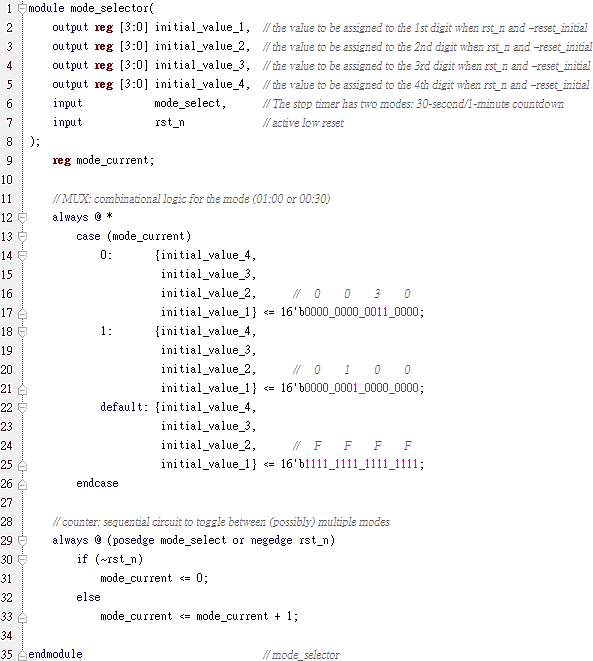
Output equations:

stopwatch\_countdown\_enable = state1’

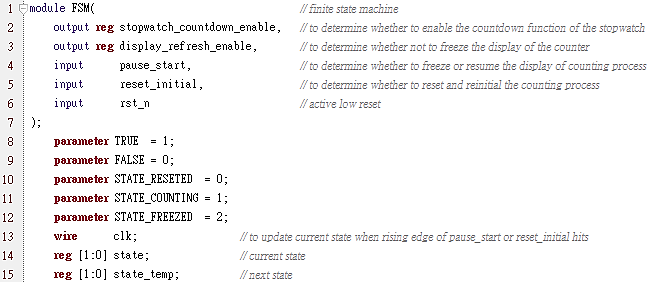
display\_refresh\_enable = state0 ⊕ state1



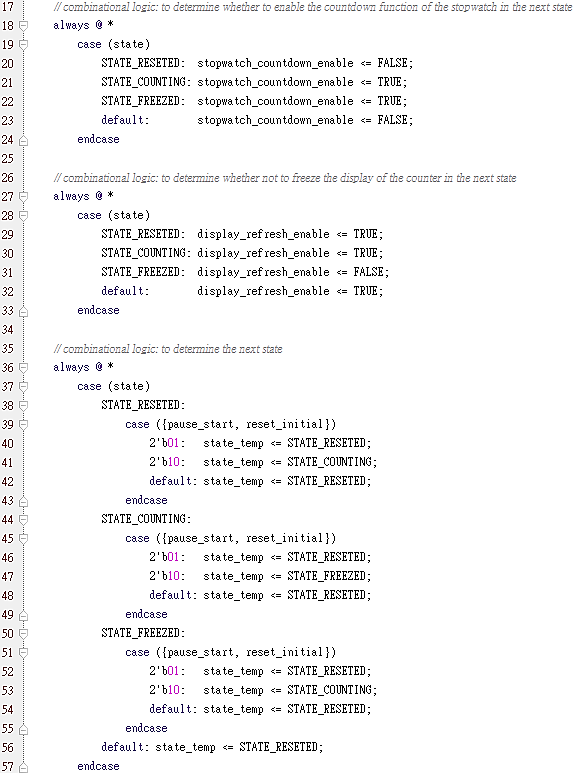
**d. Implement :**

這題只要多建構兩個模組mode\_selector、FSM，並稍加修改Experiment1的LAB\_2就好，其餘的可以直接沿用。mode\_selector在不同的模式下，要將不同的初始數值送到碼表內。可以用MUX，如下圖。 

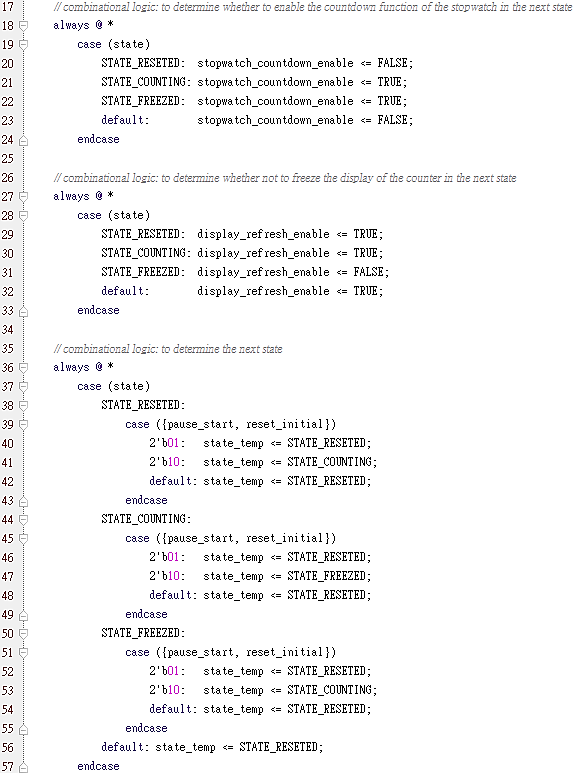
接下來的幾張圖片是模組FSM的硬體描述語言。FSM的輸出、輸入與一些內部線路的名稱如下。



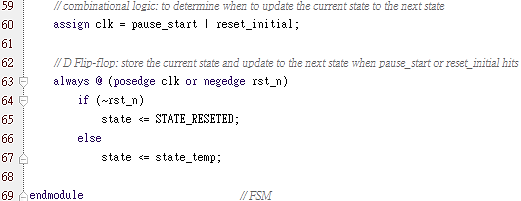
因為是moore machine，所以輸出不受輸入影響，可由狀態直接決定。如下圖，可以用MUX來依照當下的狀態來產生對應的輸出。



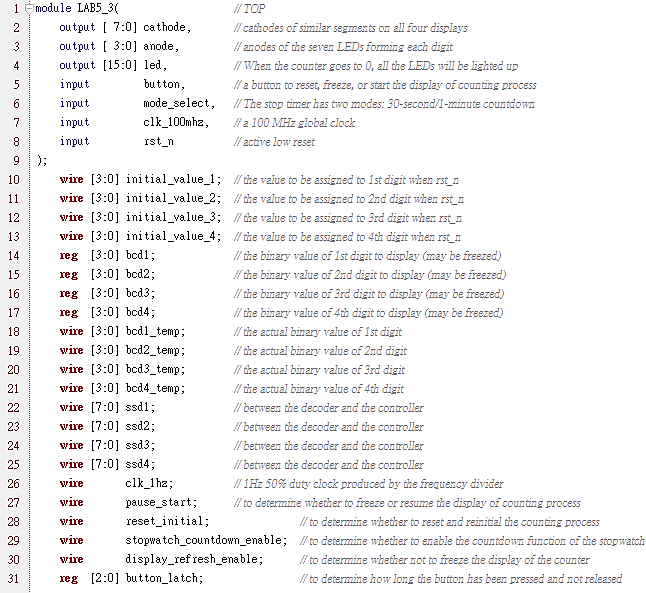
下圖是下一個狀態的combinational logic。在正常情況下，pause\_start與reset\_initial不會同時為1，而且第60到67行的D Flip-flop只有在pause\_start或reset\_initial為1時才更新狀態。因此，只需要明確列出01和10的case即可。



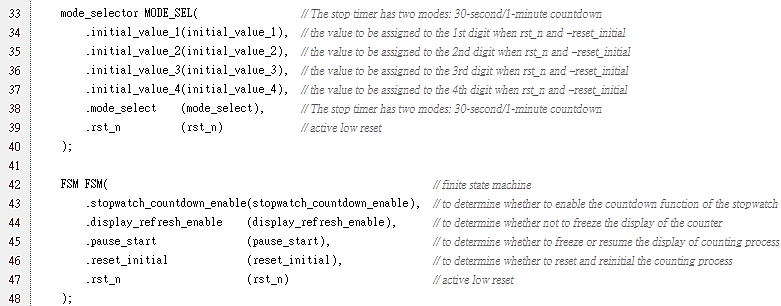
第60行的邏輯閘使得第60到67行的D Flip-flop僅在pause\_start或reset\_initial的rising edge時更新FSM的狀態。



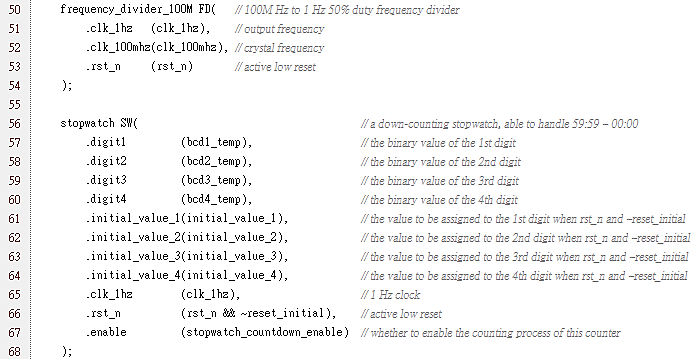
至此已萬事具備，剩下的只要微調LAB5\_2就好。接下來的幾張圖片是修改後的LAB5\_3。輸出、輸入和一些內部接線如下。



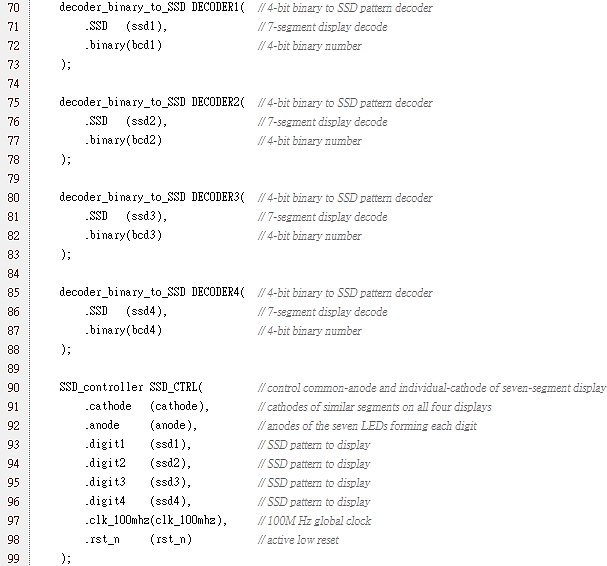
新加入兩個模組，mode\_selector和FSM。mode\_selector決定當下的倒數計時模式，在一分鐘和三十秒之間切換，且輸出對應的初始數值給碼表作重置用。而FSM則決定當下碼表的狀態，會依照輸入的不同，決定按按鈕之後會變成哪一個狀態。



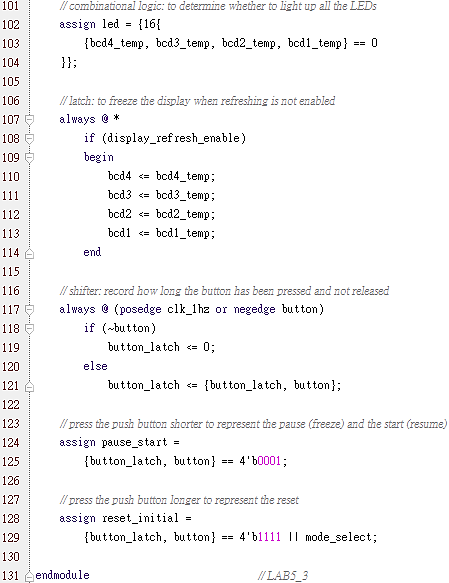
須留意碼表的數值輸出後須先經過第107到114行的latch判斷當下狀態是否為凍結顯示器，再連接到解碼器。



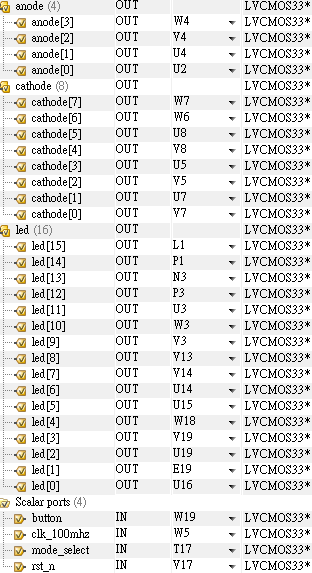
其餘模組的接線保持不變。



第102到104行有微調。原本是拿bcd4、bcd3、bcd2、bcd1來判斷是否已經數到零了，然而由於有可能因為持續保持凍結顯示器的狀態，有可能會出現明明已經過了30秒或一分鐘，燈仍然不會亮起。因此，應改為以latch前面的數值為判斷依據，如此一來，縱使顯示器的時間被凍結了，一旦碼表數到零，燈還是會亮起。第107到114是用來凍結顯示器的latch。另須留意，第129行的判定相較Experiment2之下，還需要考慮mode\_select是否有被按下。這樣無論碼表是否在計時，無論顯示器是否被凍結，只要按下mode\_select按鈕，就會立刻重置，並顯示切換後該模式的計時秒數。其餘未提及的部分與Experiment2一模一樣，直接沿用即可。



下圖是用來demo Experiment3的。



**e. Conclusion :**

其實這一題的FSM可以像老師的ppt一樣按照推導出來的table直接使用case和if…else…列出來就好，不需要像剛剛提到的一樣分成兩個output的MUX和一個next state的MUX來寫。但因為這題是moore machine，output不受input影響，可以單由state就決定，所以即使把硬體描述展開來寫，也不會感到過度雜亂。一開始在建構FSM的時候，我將D Flip-flop的C連接到TOP的一赫茲時脈，造成按鈕被按下後，還要等到下一個時脈過了才會更新到下一個狀態。後來，我將input直接連接到D Flip-flop的C，即達成「按鈕按下的當下就有反應」的效果。