**1 Implement a stopwatch function (00:00-59:59) with the FPGA board.**

**1.1 Use the four (Seven-Segment Displays, SSDs) as the display. The left two digits represent the minute and the right two digits represent the second.**

**1.2 Use two push buttons to control the function. Use one button to control start/stop and the other to control the lap and reset. When the stopwatch counts, press the ‘lap’ button will freeze the SSDs but the stopwatch continues counting, and when press the ‘lap’ button again, the SSDs will start to show current time.**

**1. Specification**

**內容: 寫下你的電路中的inputs, outputs以及其bit widths，名稱必須跟你的verilog code中相同。**

module LAB6\_1(

output [7:0] cathode, // 控制七段顯示器

output [3:0] anode, // 控制七段顯示器

input push1, // 按一下會開始計時

input push2, // 按一下會凍結或解除凍結

input f\_100mhz, // 內建的時脈

input rst\_n // 可強制關機

);

module upcounter(

output reg [3:0] value, // 現在顯示的數值

output c\_out, // 下一個時脈要進位

input [3:0] limit, // 最大允許的數值

input c\_in, // 下一個時脈要加一

input clk, // 時脈

input rst\_n // 重置

);

module decoder\_BCD\_to\_SSD(

output reg [7:0] ssd, // 七段顯示器的樣式

input [3:0] bcd // 二進位的樣式

);

module stopwatch(

output [7:0] ssd4, // 分鐘的十位數字

output [7:0] ssd3, // 分鐘的個位數字

output [7:0] ssd2, // 秒鐘的十位數字

output [7:0] ssd1, // 秒鐘的個位數字

input en, // 啟用，允許上數

input f\_1hz, // 時脈

input rst\_n // 重置，變成00:00

);

module SSD\_controller(

output reg [7:0] cathode, // 控制七段顯示器

output reg [3:0] anode, // 控制七段顯示器

input [7:0] ssd1, // 秒鐘的個位數字

input [7:0] ssd2, // 秒鐘的十位數字

input [7:0] ssd3, // 分鐘的個位數字

input [7:0] ssd4, // 分鐘的十位數字

input freezed, // 凍結顯示器

input [1:0] clk // 時脈

);

module FSM(

output paused, // 控制碼表是否上數

output freezed, // 控制是否凍結顯示器

input debounced1, // 按一下會開始計時

input debounced2, // 按一下會凍結或解除凍結

input rst\_n // 重設狀態

);

module frequency\_divider(

output reg f\_1hz, // 一赫茲的時脈(50% duty)

output reg [25:0] f\_counter, // 各種不同頻率的時脈

input f\_100mhz, // 內建於FPGA的時脈

input rst\_n // 重置

);

module debounce(

output debounced, // 經穩定後的訊號

input push, // 未經處理的訊號

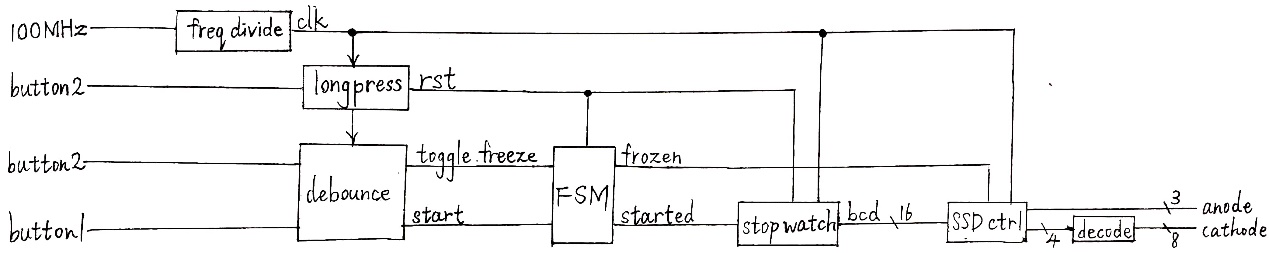
input clk, // 時脈

input rst\_n // 重置

);

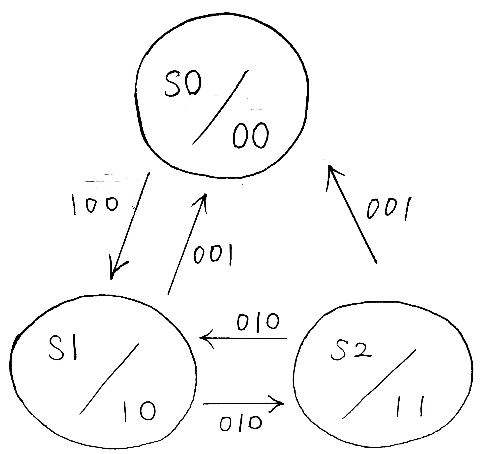
**2. Block Diagram**

**內容: 電路中的Block diagram(可以用手畫拍照或電腦繪圖)。**



**3. Finite state machine**

**內容: 電路中的Finite state machine，若無則寫無。**



state0: reseted

state1: started

state2: frozen

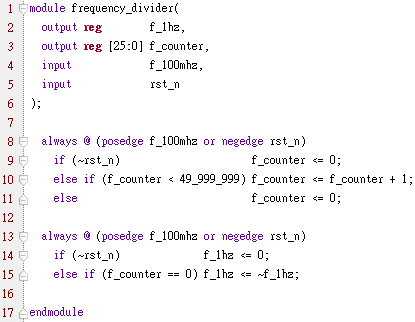
moore machine inputs: {start, toggle\_freeze, reset}

moore machine outputs: {started, frozen}

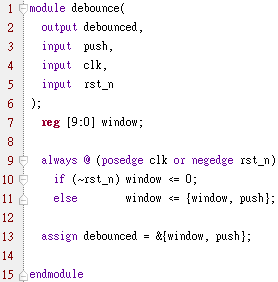
**4. Implement**

**內容: 請列出相關的logic function、詳細用文字解釋電路的運作方法、結果等等，可以貼code解釋或拍FPGA輔助解釋(但不能只貼code跟FPGA結果)。**

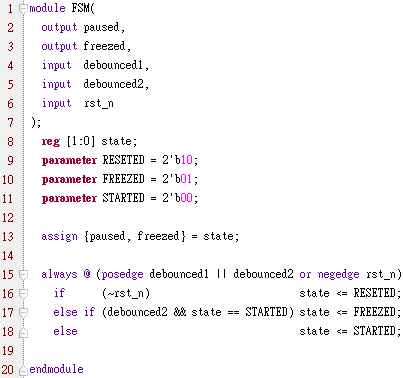
下圖是除頻器，跟上次Lab的差不多。第8到11行是計數器。第13到15行用來製造50% duty的一赫茲時脈。將除頻器的counter當作output，讓top的其他模組可以參照各種頻率。



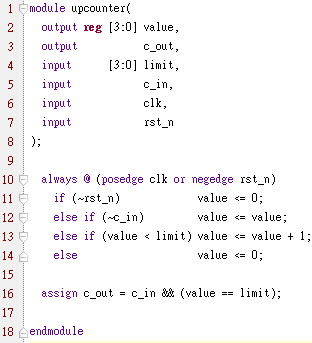
因為按鈕按下的瞬間會有擾動，有時候雖然只按一下，但卻會形成數個震盪。所以可以先把未處理的按鈕訊號連接到一個去除雜訊的模組。下圖是用來排除雜訊的，第9行到第11行是shifter。第13行的AND可以用來判斷按鈕是否處於按下的狀態超過數個時脈。如果偵測到連續按下，那就代表按鈕已經穩定。



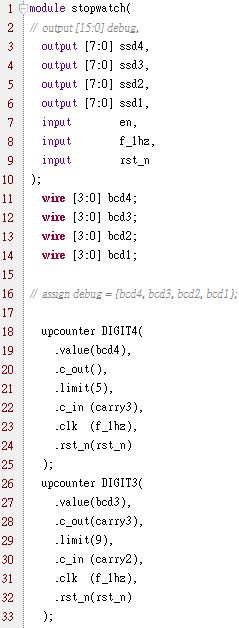
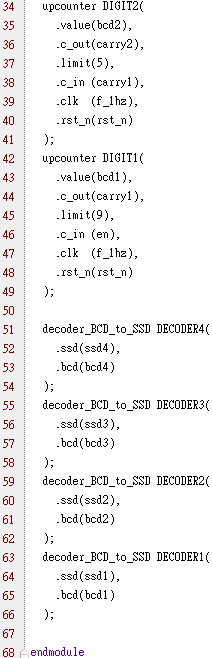
下圖是有限狀態機，只有三個狀態。因為output只與當下的狀態有關，所以是moore machine。而且，這三個狀態的output兩兩相異，所以可以直接用output的組合來表示當下的狀態，如第13行。可以用parameter來建立狀態與output之間的關係，如第9到11行。狀態是RESETED時，碼表不要上數，所以是2’b10；狀態是FREEZED時，碼表要持續下數，而顯示器要凍結，所以是2’b01；狀態是STARTED時，碼表要下數且顯示器不凍結，所以是2’b00。第15到第18行用來判斷下一個狀態。只有當按鈕被按下，或是要重置的時候，才需要改變狀態，如第15行。如果要重置，就把狀態變成RESETED，如第16行。如果當下的狀態是STARTED，而且第二個按鈕(toggle\_frozen)被按下了，則把狀態變成frozen (FREEZED)，如第17行。其餘的狀況，例如第一個按鈕(start)被按下，會使狀態變成STARTED，如第18行。



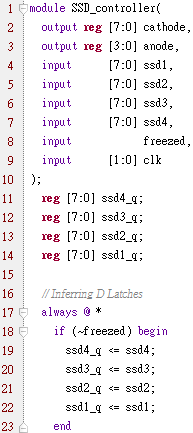
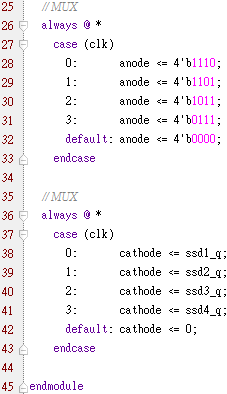
碼表有四位數，每一個位數由一個四位元上數計數器組成。下圖是上數計數器。第10到14行用來判斷下一個時脈過後要變成什麼數值。如果沒有carry in (c\_in)，就不動。否則如果當下的數值小於最大允許數值，就加一。其餘的狀況就重置為零。第16行判斷要產生進位carry out (c\_out)的條件，條件為carry in (c\_in) (代表要加一)且當下數值已經到達最大允許數值。



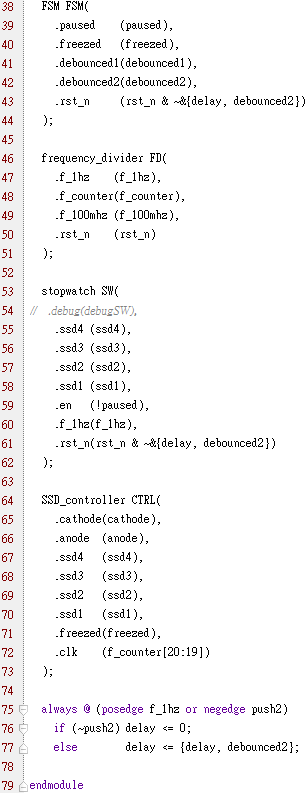
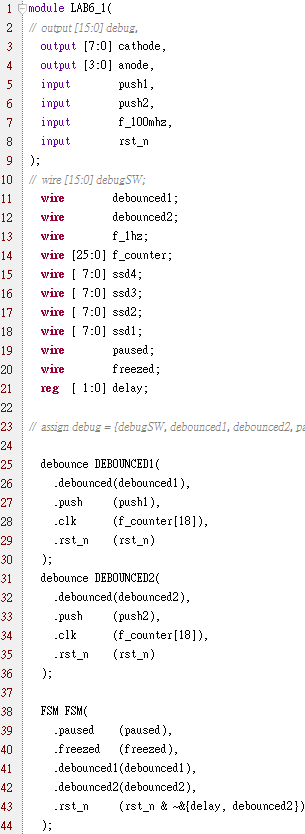
下圖是碼表stopwatch的接線方法。只要把四個上數計數器和四個解碼器接好即可。如下，把計數器的carry in (c\_in)和carry out (c\_out)兩兩相接。

用來控制七段顯示器的模組如下，跟前幾次Lab的差不多。在此，將freezed作為input，可以在第17到23行使用一些D latch來凍結顯示器。

下圖是top，把剛剛建構好的那些模組接起來即可。第75到77行用來判斷按鈕長按，如果連續按下超過數個時脈，就發出重設訊號。除了stopwatch以一赫茲為時脈之外，其餘模組運作的時脈以除頻器所產生的的f\_counter[18]為主，頻率為10^9/2^19，約1907赫茲。



**5. Conclusion**

**內容: 可以寫下你的這個lab的想法、遇到的問題、解決方法、心得等等，請自由發揮。**

這一題跟LAB4很類似，所以建構的過程很順利，沒有遇到困難。然而，我在建構這題的模組時，有許多flip flop的clk是async的，這使得我難以將本題的模組直接沿用在下一題。即使成功套用了，也會出一些莫名其妙的bug，很難debug。

**2 Implement a timer (can support as long as 23:59) with the following functions.**

**2.1 Use one DIP switch as the ‘setting’ control. When the ‘setting’ is ON, you can use two buttons to set the minute and second.**

**2.2 Use other two buttons to control the timer operation. One button for start/stop and the other button for pause/resume.**

**2.3 When the time goes to 0, light up all the LEDs.**

**1. Specification**

**內容: 寫下你的電路中的inputs, outputs以及其bit widths，名稱必須跟你的verilog code中相同。**

module LAB6\_2\_top(

output [ 7:0] cathode, // 控制顯示器

output [ 3:0] anode, // 控制顯示器

output [15:0] led, // 控制LED的明滅

input button1, // 按一下會開始倒數或增加一小時

input button2, // 按一下會暫停/繼續倒數或增加一分鐘

input setting, // 切換模式

input rst\_n, // 強制重置

input clk\_100mhz // 內建於FPGA的時脈

);

module frequency\_divider(

output reg [26:0] clk\_g, // 提供各種頻率給其餘模組使用

input clk\_100mhz, // 內建於FPGA的時脈

input rst\_n // 強制重置

);

module debounce(

output reg debounced, // 經處理後的訊號

input push, // 未經處理的訊號

input clk, // 時脈

input rst\_n // 重置

);

module onepulse(

output reg onepulse, // 經處理後的脈衝

input debounced, // 去除了繞動後的訊號

input clk, // 時脈

input rst\_n // 重置

);

module debounce\_onepulse(

output onepulse, // 經處理後的脈衝

output debounced, // 經處理後的訊號

input push, // 未經處理的、帶有雜訊的訊號

input clk, // 時脈

input rst\_n // 重置

);

module fsm(

output started, // 要開始數

input onepulse\_button1, // 要開始數或加一小時

input onepulse\_button2, // 要暫停/繼續或加一分鐘

input onepulse\_button2\_longpressed, // 要重置

input debounced\_setting, // 用來判斷現在是否處於設定模式

input rst\_n // 強制重置

);

module up\_counter(

output reg [3:0] value, // 顯示的數值

output carryout, // 需要進位

input carryin, // 需要加一

input [3:0] init, // 初始數值

input [3:0] limit, // 最大允許數值

input clk, // 時脈

input enable, // 允許加一

input rst\_n // 重設或初始化

);

module down\_counter(

output reg [3:0] value, // 顯示的數值

output carryout, // 需要借位

input carryin, // 需要減一

input [3:0] init, // 初始數值

input [3:0] limit, // 最大允許數值

input clk, // 時脈

input enable, // 允許減一

input rst\_n // 重設或初始化

);

module minute\_up\_counter(

output [3:0] bcd0, // 分鐘的個位數的數值

output [3:0] bcd1, // 分鐘的十位數的數值

output carryout, // 要進位一小時

input carryin, // 要加一分鐘

input [3:0] init0, // 分鐘的個位數的初始數值

input [3:0] init1, // 分鐘的十位數的初始數值

input clk, // 時脈

input enable, // 允許加一分鐘

input rst\_n // 重置或初始化

);

module hour\_up\_counter(

output [3:0] bcd0, // 小時的個位數的數值

output [3:0] bcd1, // 小時的十位數的數值

output carryout, // 要進位一天

input carryin, // 要加一小時

input [3:0] init0, // 小時的個位數的初始數值

input [3:0] init1, // 小時的十位數的初始數值

input clk, // 時脈

input enable, // 允許加一小時

input rst\_n // 重置或初始化

);

module minute\_down\_counter(

output [3:0] bcd0, // 分鐘的個位數的數值

output [3:0] bcd1, // 分鐘的十位數的數值

output carryout, // 要借位一小時

input carryin, // 要減一分鐘

input [3:0] init0, // 分鐘的個位數的初始數值

input [3:0] init1, // 分鐘的十位數的初始數值

input clk, // 時脈

input enable, // 允許減一分鐘

input rst\_n // 重置或初始化

);

module hour\_down\_counter(

output [3:0] bcd0, // 小時的個位數的數值

output [3:0] bcd1, // 小時的十位數的數值

output carryout, // 要借位一天

input carryin, // 要減一小時

input [3:0] init0, // 小時的個位數的初始數值

input [3:0] init1, // 小時的十位數的初始數值

input clk, // 時脈

input enable, // 允許減一小時

input rst\_n // 重置或初始化

);

module ssd\_control(

output [7:0] cathode, // 控制顯示器

output reg [3:0] anode, // 控制顯示器

input [3:0] bcd0, // 分鐘的個位數的數值

input [3:0] bcd1, // 分鐘的十位數的數值

input [3:0] bcd2, // 小時的個位數的數值

input [3:0] bcd3, // 小時的十位數的數值

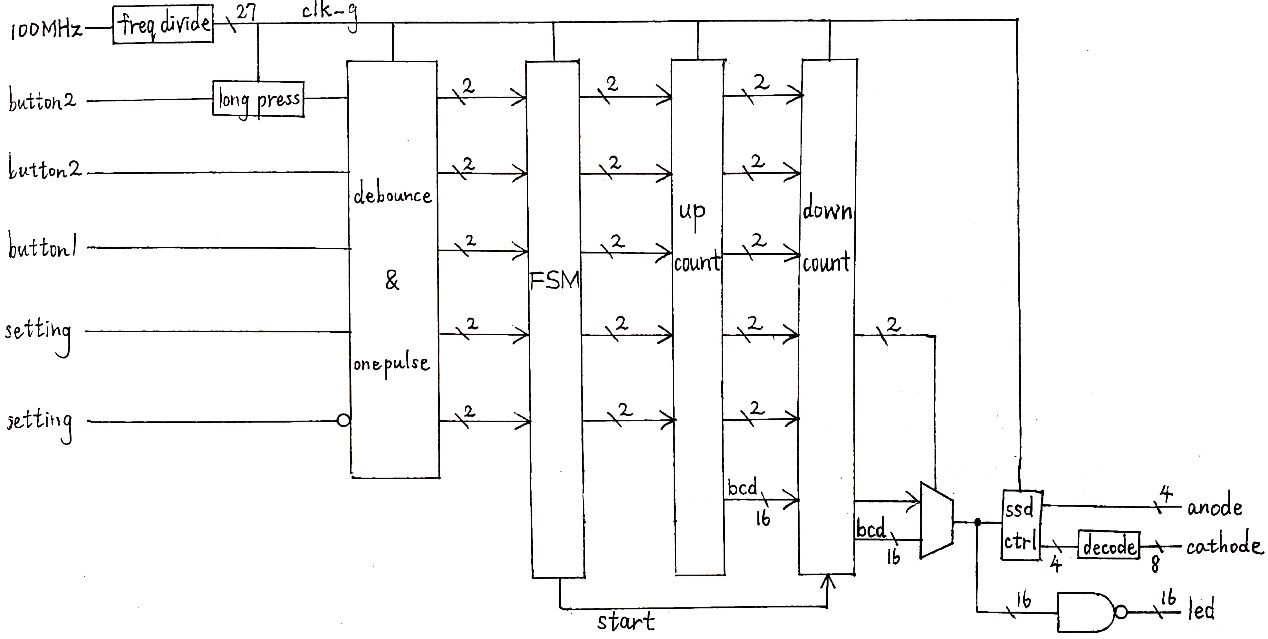
input freezed, // 凍結顯示器 (本題沒用到)

input [1:0] clk // 時脈

);

**2. Block Diagram**

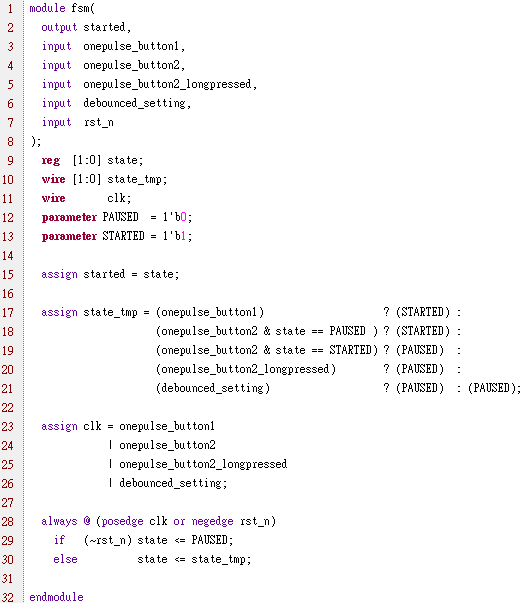
**內容: 電路中的Block diagram(可以用手畫拍照或電腦繪圖)。**



**3. Finite state machine**

**內容: 電路中的Finite state machine，若無則寫無。**

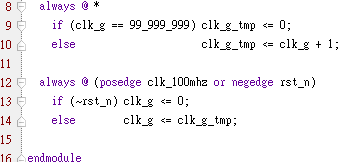
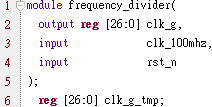
只有兩個狀態，正在下數或不動。有四個輸入，分別代表「第一個按鈕被按下」、「第二個按鈕被按下」、「第二個按鈕被持續按下超過數個時脈」、「正處於設定模式」。此FSM有一個輸出，用來控制下數計數器是否要以一赫茲為頻率減一分鐘。由於這是moore machine，而且狀態跟輸出之間的關係很單純，所以可以把狀態直接與輸出連接起來，如第15行。當狀態是不動(PAUSED)時，計數器不要下數，所以是1’b0，如第12行。當狀態是正在下數(STARTED)時，計數器要下數，所以是1’b1，如第13行。只有當輸入有改變時，才需要更新FSM當前的狀態，所以將時脈與輸入連接，如第23到26行所示。而第17到21行會依照當下的輸入與狀態來判定下一個狀態。若第一個按鈕被按下，則下一個狀態為STARTED。如果第二個按鈕被按下，且當下狀態是PAUSED，則下一個狀態也要變成STARTED。其餘情況都會使下一個狀態變成PAUSED。



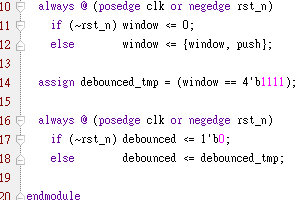
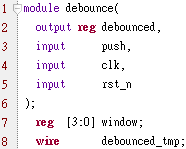
**4. Implement**

**內容: 請列出相關的logic function、詳細用文字解釋電路的運作方法、結果等等，可以貼code解釋或拍FPGA輔助解釋(但不能只貼code跟FPGA結果)。**

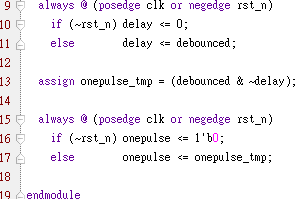
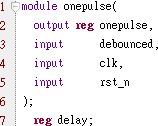
下圖是除頻器。把FPGA內建的100M Hz連接到一個27位元計數器，從零開始數，最高數到100M - 1，並直接把計數器的數值當作輸出。如果有其他模組需要一赫茲的時脈，只要直接接入clk\_g[26]即可。其他位元就當作各種不同頻率的時脈，讓其餘有特殊需求的模組各取所需。



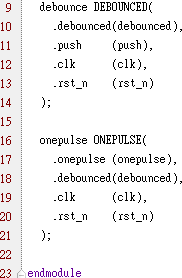
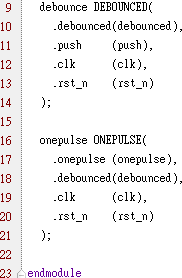
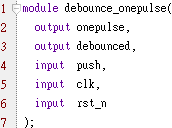
下圖可以穩定訊號，用來消除按鈕的震動。原理跟上一題一樣。



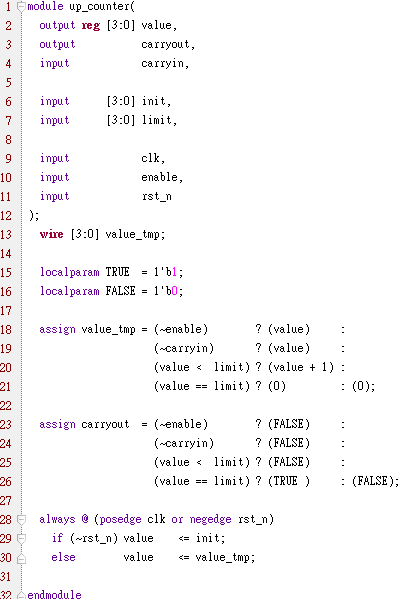
下圖是用來產生脈衝的。如此一來，模組產生訊號給有限狀態機後，有限狀態機比較能夠輕鬆而不易出錯地更新到下一個狀態。其主要目的是為了讓有限狀態機內運作的頻率跟外部其他模組盡可能地同步。



下圖只是把剛剛那兩個模組包裝在一起而已。只要把一個訊號連接到這個模組，這個模組就會產生去除按鈕跳動等雜訊後的乾淨訊號，以及時長為一個時脈的脈衝。

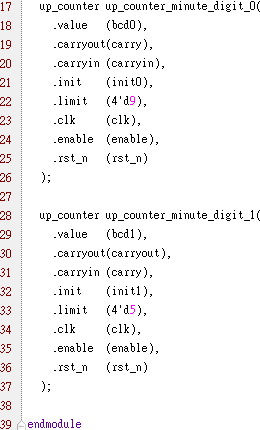
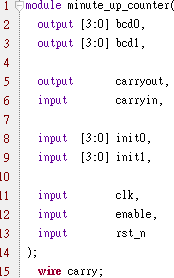


以下兩張圖片分別是四位元上數計數器和四位元下數計數器，結構相同。第18到21行判斷下一個時脈過後的數值應變為多少，然後第28到第30行在時脈來的時候更新數值。而第23到26行則是判斷使否要輸出進位的訊號。

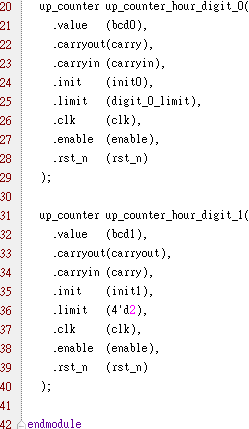
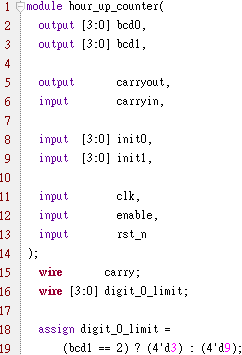


四位元上數計數器和四位元下數計數器只有一點點不同，只要使用文字編輯器將所有圓括號內的「limit」與「0」互換，再將「+」改成「-」，即為四位元下數計數器。

下圖是兩位數的上數計數器，負責控制分鐘的個位數字和分鐘的十位數字。分鐘的個位數字最高到9，如第22行。分鐘的十位數字最高到5，如第33行。把個位數和十位數之間的進位訊號接好，如第19、31行。其他接線直接與input或output連接即可。

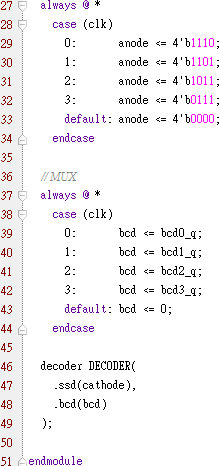
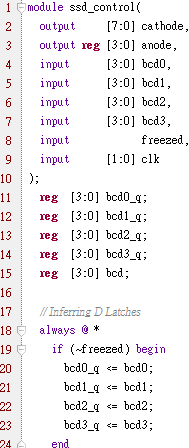


下圖也是兩位數的上數計數器，但負責控制的是小時的個位數字和小時的十位數字。小時的個位數字的最大允許數值由小時的十位數字決定，如第18、19、25行。如果十位數是2，那個位數的最大允許數值就是3，否則為9。而小時的十位數字則最高到2，如第36行。把個位數和十位數之間的進位訊號接好，如第22、34行。其他接線直接與input或output連接即可。

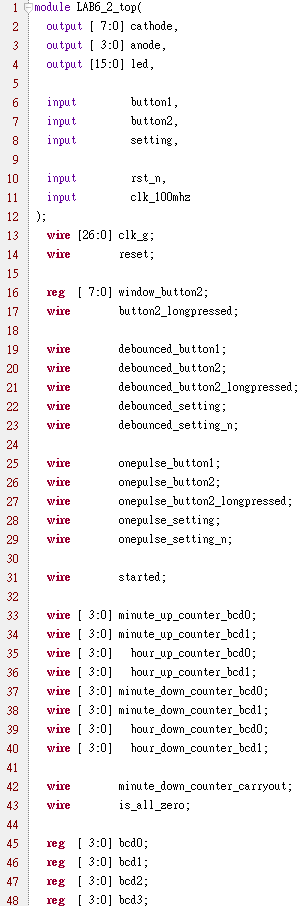


另外兩個下數計數器跟剛剛提到的兩個上數計數器只有一點點差別。用文字編輯器把所有「up」取代為「down」，即為另外兩個下數計數器。

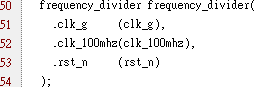
下圖是控制七段顯示器的模組，跟之前的原理一樣。因為這題不用凍結顯示器，所以實際上只需要那兩個MUX就好，不用第18到24行的D Latch。其中的decoder功能跟上一題一樣，可直接沿用。



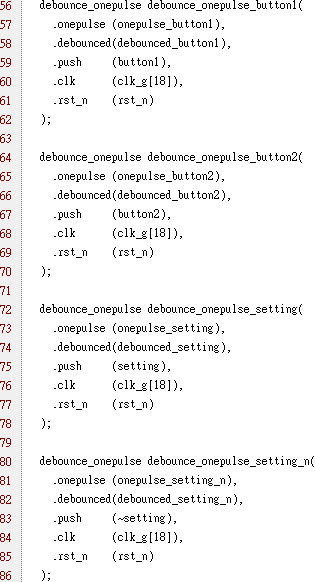
接下來只要把上述模組連接起來就好。LAB6\_2\_top的線路名稱如下圖。



先用除頻器製造出所有模組可能會用到的時脈。

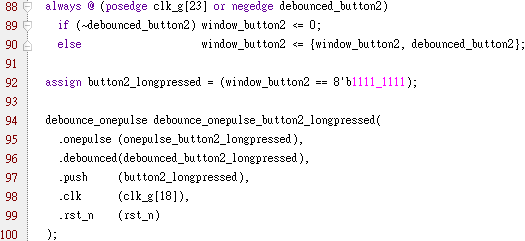


處理所有input，把彈跳過濾掉，並產生脈衝訊號。這些模組的時脈仿照上一題，使用clk\_g[18]，頻率為10^9/2^19，大約1907赫茲。

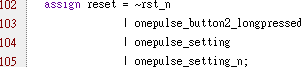


為了偵測按鈕長按，使用一些shifter來判定按鈕是否被按下超過連續數個時脈

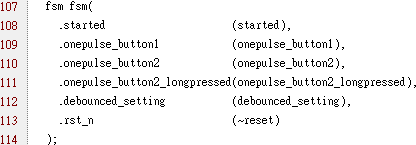
。並且也將長按的訊號處理成脈衝訊號。



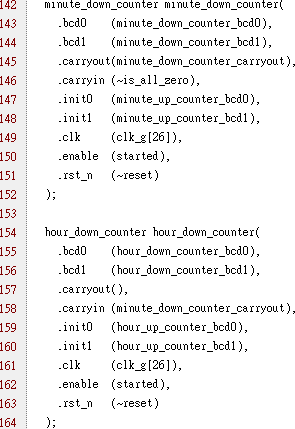
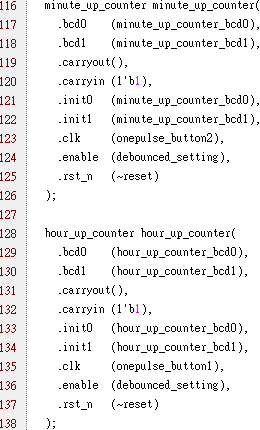
要reset的情形包含：強制重置、第二個按鈕被按下過久、或切換設定模式時。



把經過處理的input連接到有限狀態機。輸出started用來控制計數器是否下數。



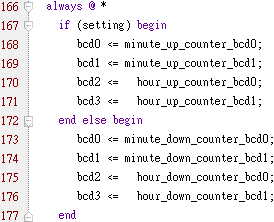
把所有計數器的接好。下數計數器的分鐘要向小時傳送借位訊號，如第145、158行。第一個按鈕被按下時，上數計數器的小時數值會加一，如第135行。第二個按鈕被按下時，上數計數器的分鐘數值會加一，如第123行。而因為clk\_g[26]的頻率為一赫茲，故適合作為下數計數器的時脈，如第149、161行。其中，上數計數器只有在設定模式下允許數值加一，如第124、136行。下數計數器則只有在有限狀態輸出started時才允許下數，如第150、162行。且上數計數器的數值連接到下數計數器的初始數值，如此一來，當下數計數器收到重置訊號時，就會把數值設定成上數計數器的數值，如第147、148、159、160行。



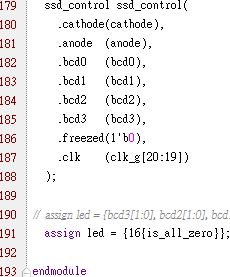
另外，為了避免下數計數器數到零之後還繼續下數，必須考慮數值是否已經全部為零，如果還沒數到零，才允許繼續下數。因此，將以下的判斷條件連接到下數計數器的carry\_in，如第146行。



至此，計數器都已經配置完成。接下來只要建構好顯示器的部分即可。將剛剛那些計數器的數值連接到MUX，當處於設定狀態下時，把上數計數器的數值連接到顯示器控制器，反之，把下數計數器的數值連接到顯示器控制器。



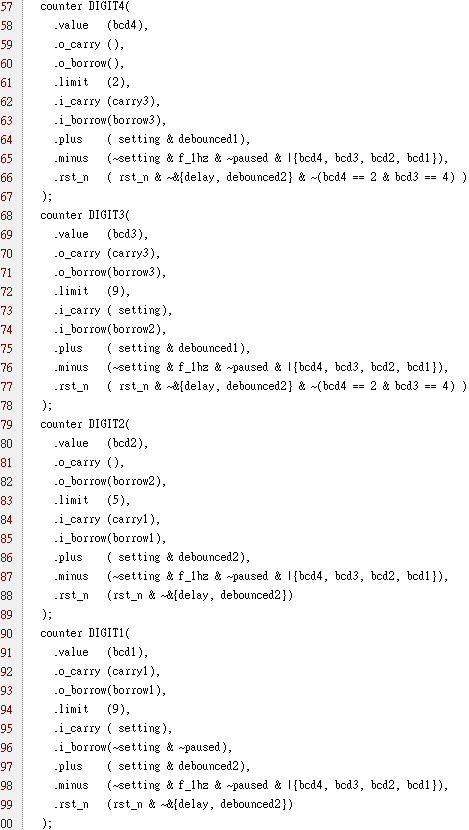
最後連接顯示器控制器，大功告成。



**5. Conclusion**

**內容: 可以寫下你的這個lab的想法、遇到的問題、解決方法、心得等等，請自由發揮。**

這一題花了我較多的時間，我捨棄上一題的作法，並盡量讓所有訊號都與clk\_g[18]同步，以避免各個計數器抓不到進位或借位時機的問題。我一開始試著沿用上一題的模組，但後來發現上一題的模組的性質不夠好，即使我把重寫部分的counter，整合上數和下數的功能，在我硬著頭皮把所有top的接線連接起來之後，仍然造成了如下窘境：



我花了一個下午試著添加更多判斷條件在計數器的接口上（當然，這樣是行不通的），最後還是無法成功debug，反而還讓我的bug更難找了。