

第七組

黎佑廷 105030009

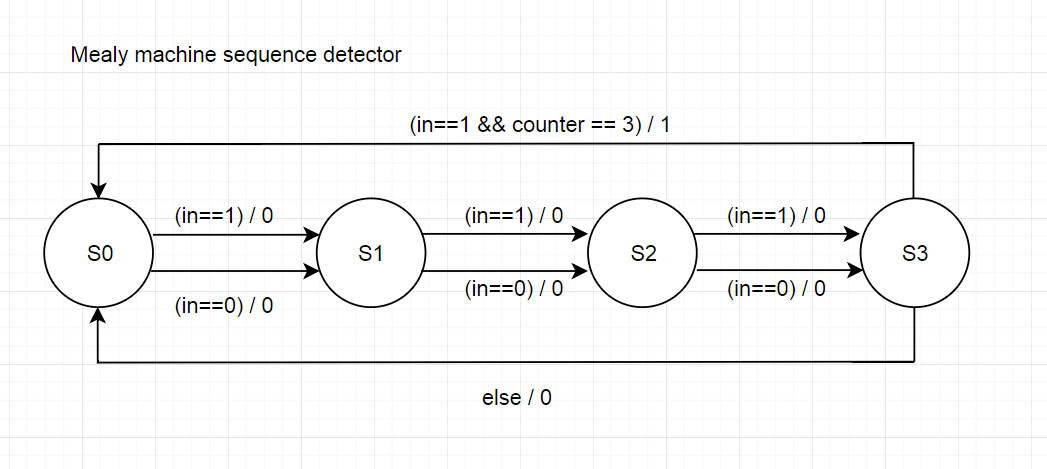
郭家瑋 105030015

lab4

Verilog Question 1:

Mealy machine sequence detector

* State-transition diagram:



當 state == S0時，不管怎樣都是輸出0，而且下一個state一定是S1，只是當in == 1時，我將counter + 1；

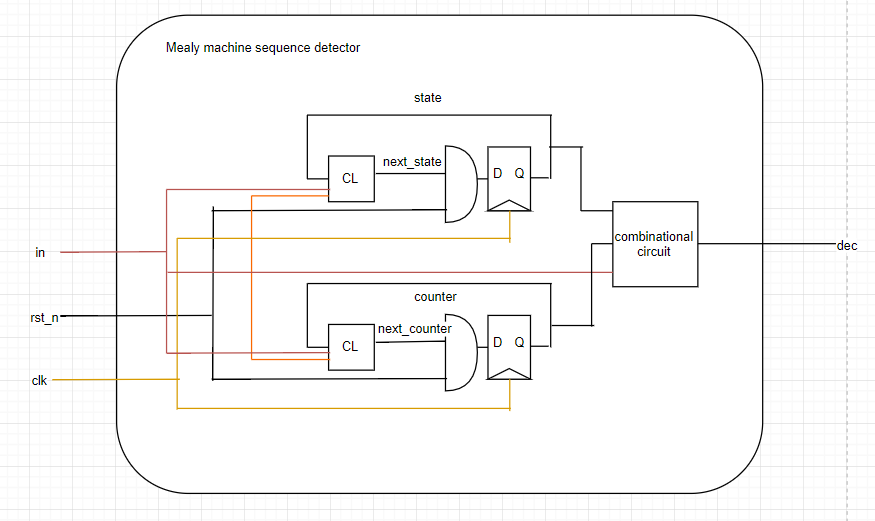
當state == S1 時，也是不管怎樣都輸出0，而且下一個state一定是S2，只是當in == 0時，我將counter + 1 ;

當state == S2 時，也是不管怎樣都輸出0，而且下一個state一定是S3，只是當in == 0時，我將counter + 1 ;

當state == S1 時，如果in == 1而且 counter == 3，代表已經偵測到1001這個pattern，所以輸出1，其他的狀況則是輸出0，下一個state回到S0以繼續偵測接下來的pattern。

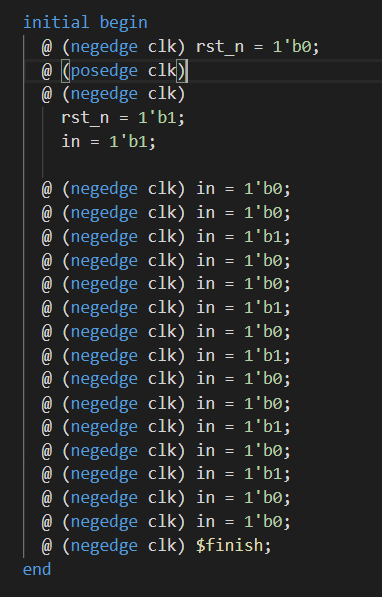
因為無論input為何，state的變換都是S0->S1->S2->S3所以每四個bit都會重新detect一次。

* Block Diagram



這題我所使用的驗證方法是給予跟老師投影片一樣的input，看看出來的waveform有沒有跟投影片上的一樣。

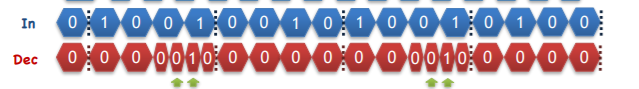
Code的部分如下(給予1001\_0010\_1001\_0100的input):



* Waveform



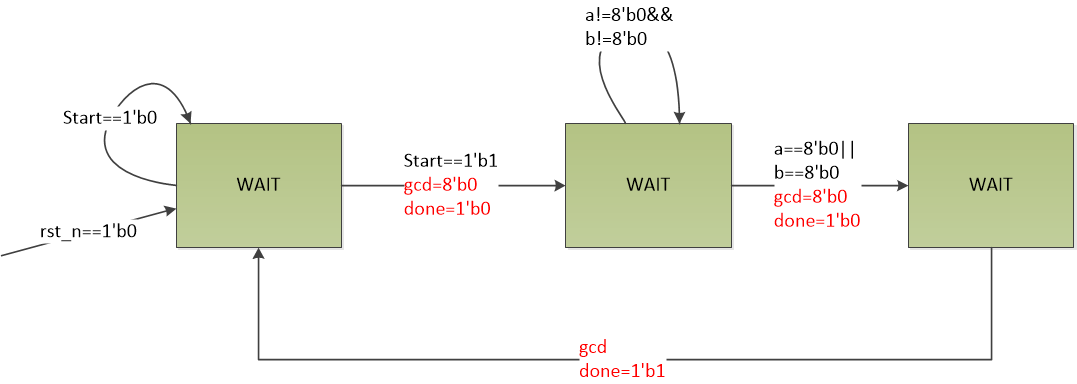
可以發現這個waveform跟老師投影片的一模一樣



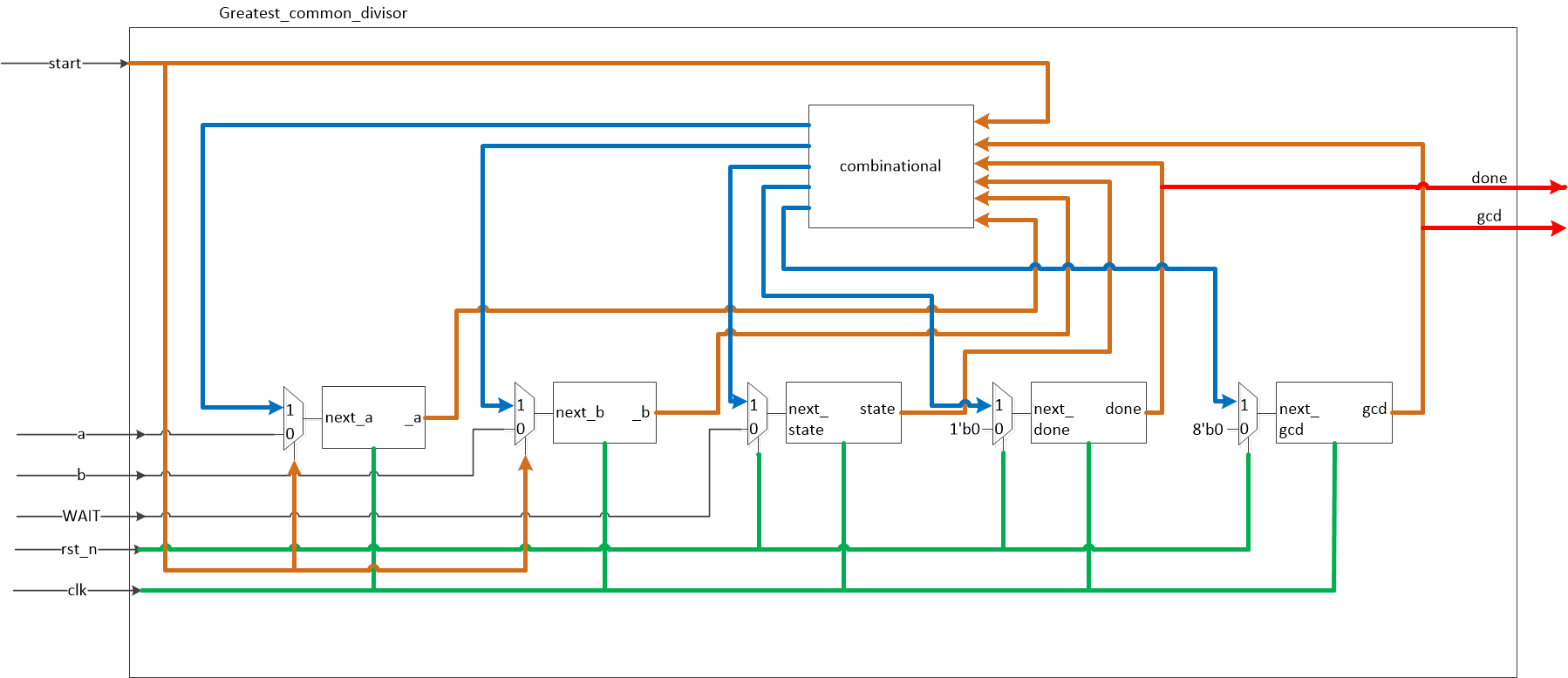
Verilog Question 2

64 x 8 memory array MEM

* State transition



* Block diagram



在state transition diagram中。首先由rst=0使系統進入WAIT state，因此rst\_n也是初始所有的DFF的信號。在rst=0時，會初始state為WAIT、done=0、gcd=0。若start等於0就會一直在WAIT state中，當start=1，會由combinational電路馬上讀進a、b，存入next\_a、next\_b。指定next\_state=CAL state，此時的gcd和done都是output=0。在CAL state中，若a!=0&&b!=0，就會一直在CAL state中互相相減。直到a或b等於0，就會將不等於零的一方設為next\_gcd，並且next\_done=1，next\_state=FINISH。會需要next\_gcd和next\_done的原因在於，由於spec的CAL state到FINISH state的輸出皆為0，而在FINISH state才output gcd與done，因此要放在DFF中。最後將next state設為WAIT state。

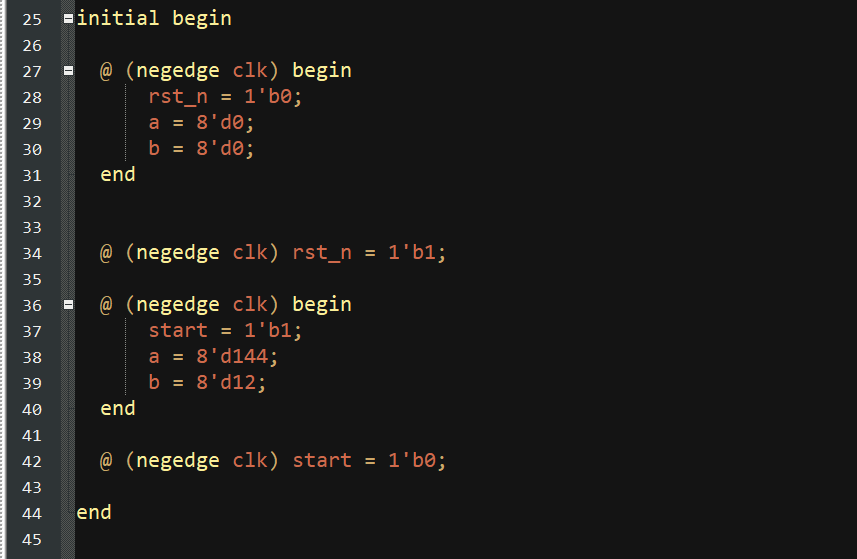
有想過要不要打default。照理說，宣告兩個bit的state，卻只有三個state應該要打default。不過，我在default中，將next\_state=WAIT會使的就算還沒reset\_n，一開始state就會是WAIT，這樣會跟spec不符，所以還是刪去了。

測試的方法為在negedge clk給出rst的初始信號，將state固定在WAIT，給出a與b，此時的a與b不是真正的a與b，因為start!=1，只是為了測試在start=1時能不能拿到對的a與b。

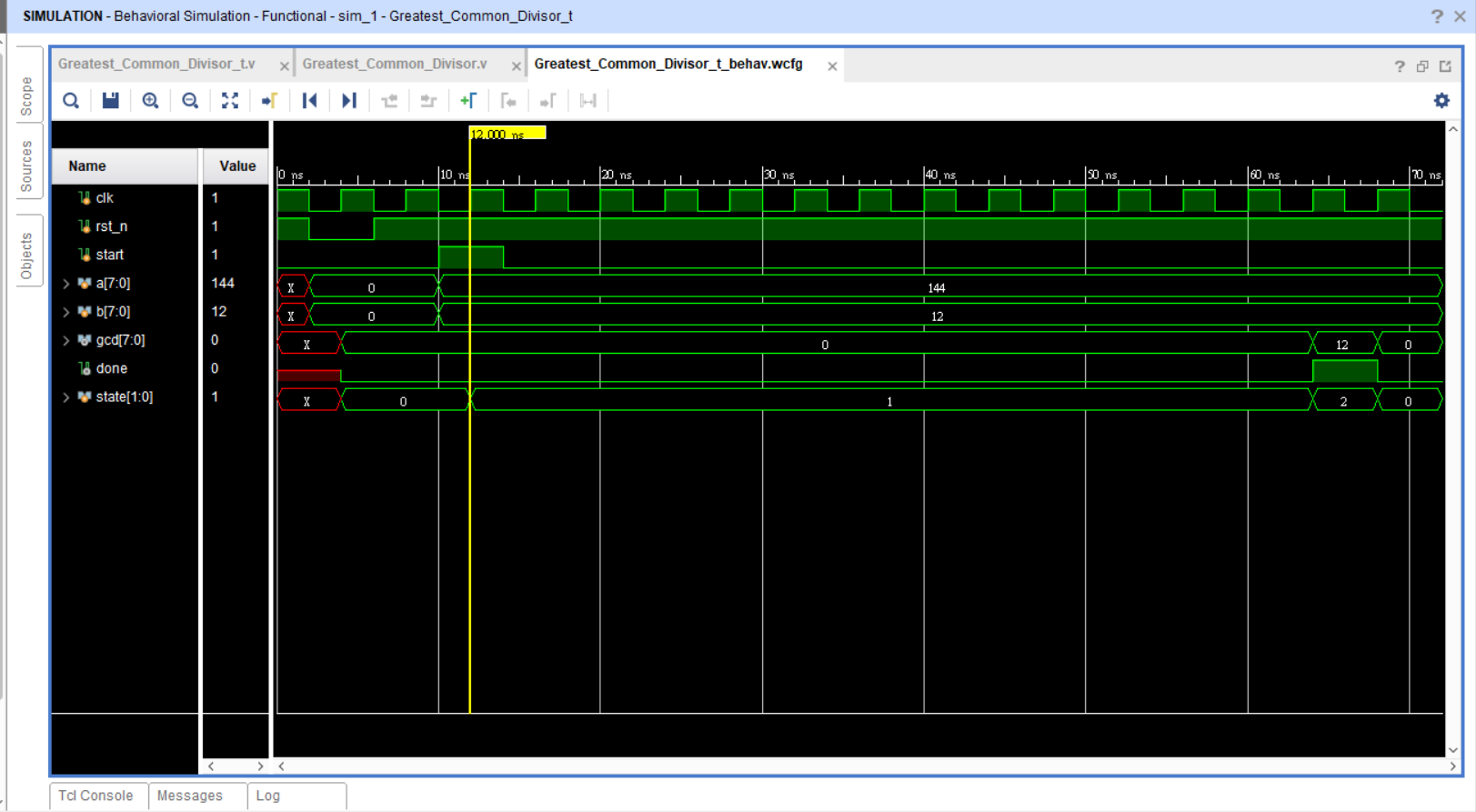
過一個cycle的negedge clk，將rst=1。再過一個cycle的negedge clk，讓start=1，使state能進入CAL，此時給出將要計算的a、b。

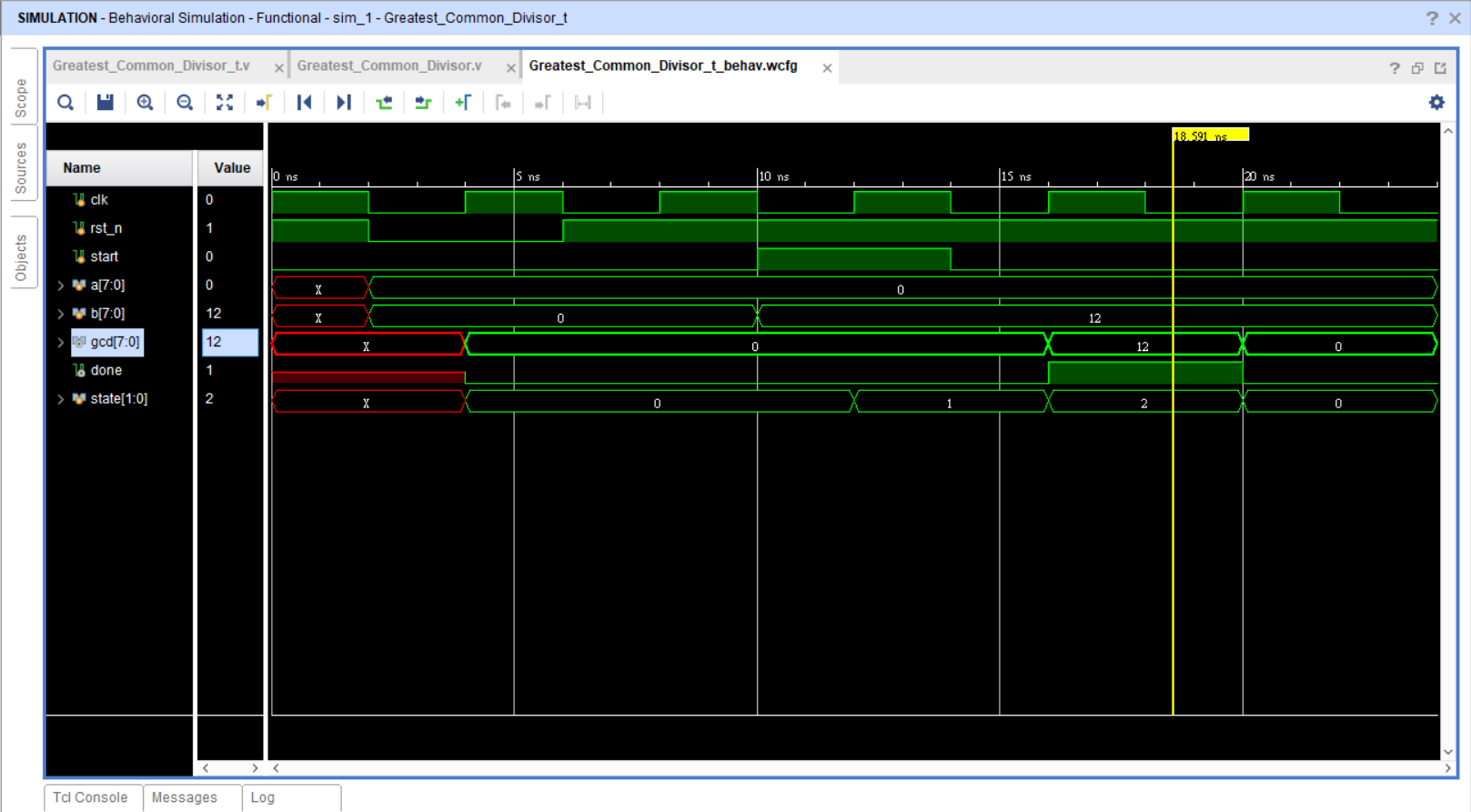
再過一個cycle讓start=0。最後，因為結果易於檢查，僅用人工檢查結果。

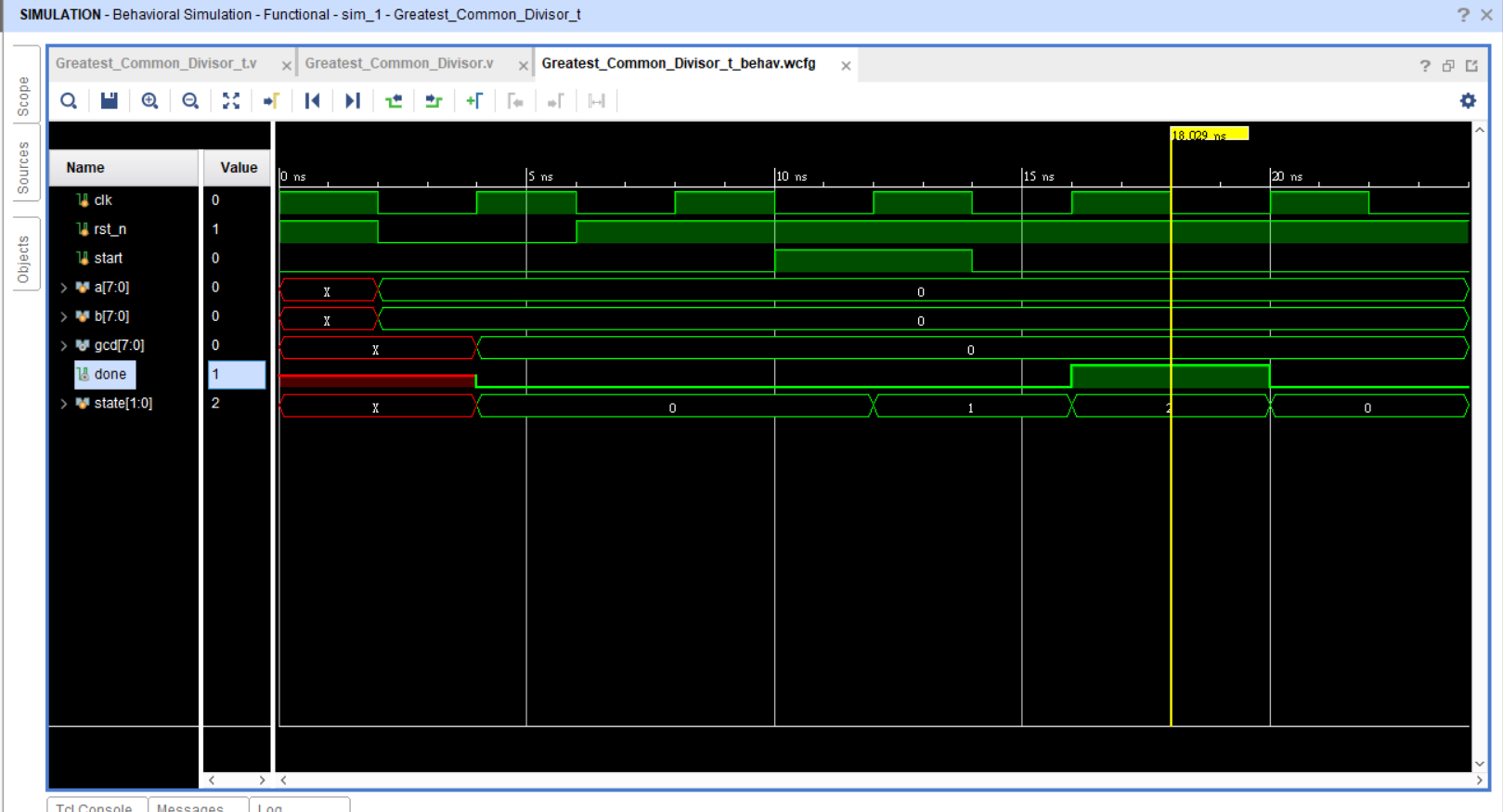
如圖：



* Waveform





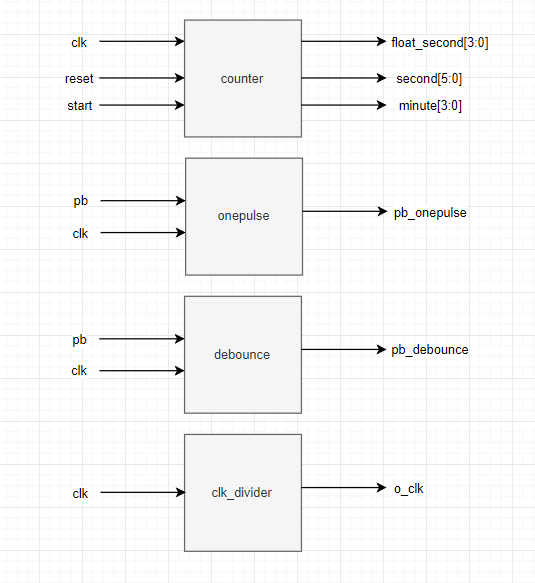


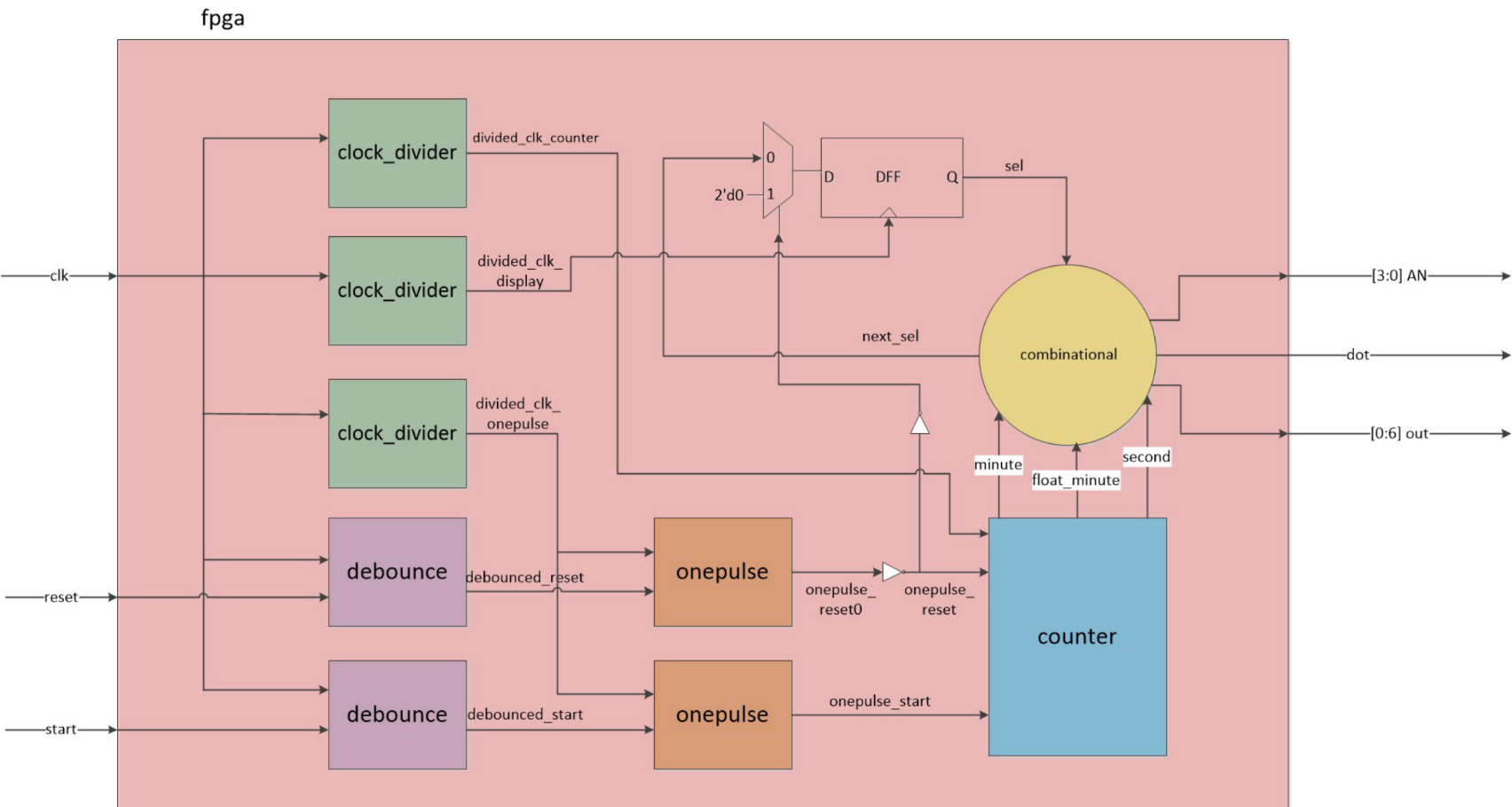
Verilog Question 3

FPGA

* Block diagram

在top module：fpga中包含了以下小module，為了簡化整體的block diagram，先放上小module本身的port，在整體的block diagram就省略小module的port。





* 說明

關於clock，因為這次的精準度是0.1sec所以要先將板子我提供的頻率(100MHZ)除以107這樣頻率就會是10HZ，週期就會是0.1秒，由於不同情況下會需要不一樣的clock來進行運算，所以我用clock\_divider做出了三個不一樣的clock:

divided\_clk\_counter : 用於顯示數字的快慢，因為精準度是0.1sec所以就除以107，用於counter這個module。

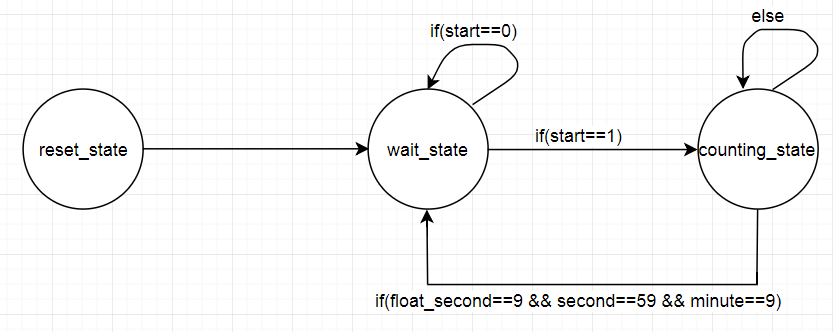
divided\_clk\_onepulse : 用於onepulse這個module上，它的頻率必須跟divided\_clk\_counter一樣，這樣才不會發生在counter在遇到posedge clk前onepulse的訊號就變回0的尷尬情形。

divided\_clk\_display : 用於讓四個七段顯示器分別通電的時間週期。我將內建clock乘以215。若四個七段顯示器通電時間太短則來不及通電，無法顯示正確的數字，若太長則無法造成明顯的視覺暫留。

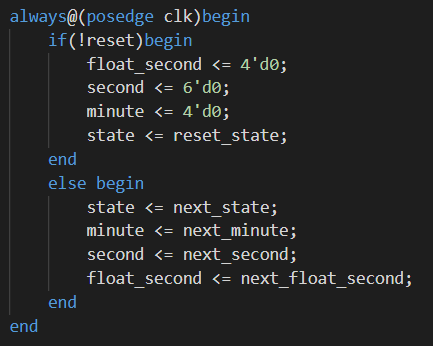
clk設定好後，處裡按鈕的去除雜訊，按鈕包括rst與flip。debounce module包含四個DFF相連，DFF的clk都使用BASYS內建clk，即100MHz。當四個DFF output都是1代表準確的按下，輸出debounced\_reset、debounced\_start。

關於counter這個module，它的主要任務就是計算時間，並將output(float\_second、second、minute)傳給top module，詳細解釋如下:

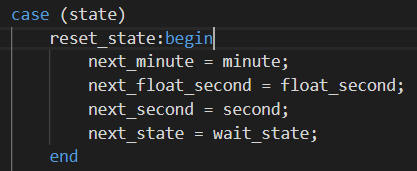
* State transition diagram



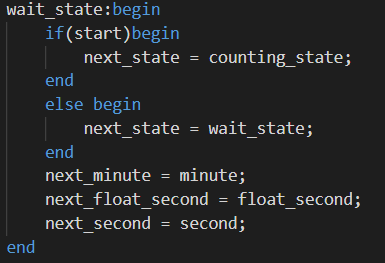
在偵測到reset訊號時，我將output都reset成0、state設給reset\_state，如果沒有偵測到reset訊號，那state、float\_second、second、minute就會分別等於conbinational block計算出來的next\_state、next\_float\_second、next\_second、next\_minute，code如下圖所示:



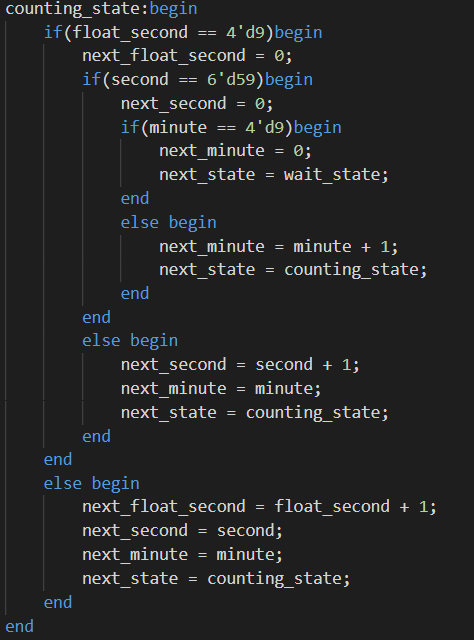
在reset\_state時，下一個state設給wait\_state以等待start訊號，有關時間的變數則是保持不動，code如下圖所示:



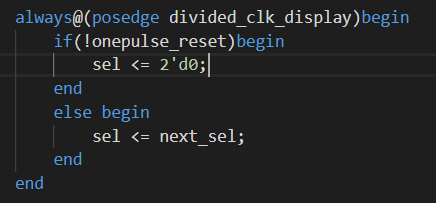
在wait\_state時，主要的任務就是等待start訊號，如果start訊號來了，那下一個state就是counting\_state，如果沒來就是繼續在wait\_state等待，有關時間的變數則是保持不動，code如下圖所示:



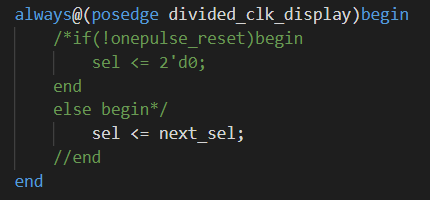
counting\_state的主要任務就是進行時間的運算，如果小數第一位等於0.9時(0.9秒)我們必須將它歸零，並把second加1，如果second等於59秒，那也一樣必須將它歸零，並把minute加1，如果minute跟float\_second都等於9，而且second也等於9就代表已經算到底了，接下來要歸零，並進入wait\_state等待下一個start訊號再開始計算，code如下圖所示:



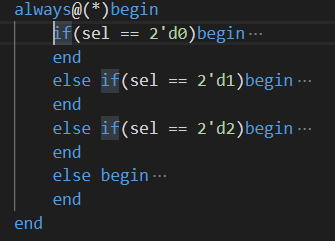
這個部分主要是處理如何將float\_second、second以及minute正確的顯示在七段顯示器上，因為七段顯示器有四個，需要分別通電而不能同時，因此我利用sel依序去跑，來決定現在哪個七段顯示器要通電，用來初始的訊號是onepulse\_reset，而這裡的DFF則是用clk\_display，code如下圖所示:



但後來在demo的時候助教跟我說控制四個七段顯示器的變數並不用吃reset訊號，這樣reset時才不會發生問題，所以我後來把它改掉了，code如下:



至於conbinational的架構大概長這樣(如下圖所示)，其中sel = 0是用來處理最右邊的七段顯示器，sel = 1是右邊數來第二個，sel = 2是右邊數來第三個，sel = 3是最左邊的。



* 心得(BY郭家瑋)

這次的lab感覺比較輕鬆。比較有釐清上問題的是，我本來搞不太清楚reset跟start的差異，後來想清楚應該是，reset是初始的信號，而start是把state固定在WAIT或是CAL開始計算的信號。這樣也比較符合reset與start的字詞意義。Reset是在系統發生奇怪錯誤時，可以按下，讓系統不管是現在在何state，都能回到最初的狀態，也是WAIT，而start是一種從IDLE(WAIT)開始的信號。

* 心得(BY黎佑廷)

我覺得這次lab的第一題不難，因為之前數位邏輯設計好像也寫過差不多的題目，所以寫起來比較沒那麼陌生，至於fpga的部分，由於上次的lab已經讓我熟悉了如何讓七段顯示器四個都顯示，以及clk等相關問題，所以這次寫起來並沒有上次費力，但也很充實，讓我對按鈕、七段顯示器的運作更加熟悉。

**成員:**

黎佑廷: 負責第1題以及fpga

郭家瑋: 負責第2題以及fpga