|  |
| --- |
| **EECS 2070 02 Digital Design Labs 2019**  **Lab 4** |
| **學號：107000115 姓名：林珈卉** |

1. 實作過程
   1. lab4\_1.v
      1. 設計概念
         1. 這個程式的功能為 BCD Counter
         2. 使用 Switch 去控制數值的大小
         3. 每四個 Switch 為一組，一組控制一個 digit 的 7-segment LED 顯示器
         4. 每一組的 Switches 由左到右為 、、、
         5. 若數值為 10 至 15 之間，則顯示器顯示 9。
      2. 實作方法
         1. 宣告四個 wire [3:0] 變數 BCD0、BCD1、BCD2、BCD3，分別代表四個 digit 的數值
         2. reset 為 1 時，四個變數皆 assign 成 0；reset 為 0 時，則 assign 為 switch 計算出的值
         3. 將 clk 經過 module clock\_divider，得到週期為 倍的 clk\_div
         4. 設置一個 always block，由 posedge clk\_div trigger，用以控制要顯示及改變哪一個 digit 的顯示器，輸出的變數為 reg [3:0] DIGIT
         5. 另一個 always block 將要顯示的數值轉換成可以控制 7-segment ，7 個 bit 的 reg 變數 DISPLAY
   2. lab4\_2.v
      1. 設計概念
         1. 設計一個可以往上／往下數的 2-digit counter，由顯示器的右兩個 digit 顯示，範圍由 0 至 99。
         2. 透過按鈕可以控制 counter 暫停／繼續，及用左邊兩個 digit 記錄當下的數值，如同馬錶的功能。
         3. 當 counter 往上數至 99 或是往下數至 0，counter 會停下，並各有對應的 LED 燈會亮起。
      2. 實作方法
         1. 設置 reg [1:0] state 去紀錄目前的狀態
         2. 共有三種狀態，為「初始狀態」、「計算中」、「暫停」，state 三個數值各別對應到這三種狀態
         3. 使用 clock\_divider 得到不同週期的三種 clock，clk\_13、clk\_23 及 clk\_25（、、）
         4. 將由按鈕獲得的訊號 en、dir、record 分別接上 debounce、onepulse 及 delay\_signal
            1. delay\_signal 是用來將 onepulse signal 延長
            2. debounce 使用 clk\_13，onepulse 使用 clk\_23，delay\_signal 使用 clk\_23
            3. 在初始狀態無法使用這三個控制訊號
            4. 經過三個 module 後的訊號稱為 xxx\_d
         5. 宣告 reg 變數 reg\_dir，用來記錄目前的方向
         6. 宣告 reg 變數 [6:0] count，用來記錄目前計算的數值
         7. 宣告 reg 變數 [3:0] BCD2, BCD3，用來記錄碼表紀錄的數值
         8. 宣告 wire 變數 [3:0] BCD0, BCD1，分別表示個位數及十位數的數值，assign 成 count%10 及 count/10
         9. 設計兩個 always block
            1. 一個用 clk\_25 trigger

可以 reset（state == 「初始狀態」時）

其他兩個狀態時，將 next\_xxx 訊號送給 reg\_dir、count、BCD2、BCD3

* + - * 1. 另一個為 always@\*

最上面先設置 default，將各個訊號送給 next\_xxx

若 state 為「初始狀態」，則只有 en\_d 訊號可以控制，將 next\_state 設為「計算中」

若 state 為其他兩種狀態

訊號 dir、reset、record 可以使用

若 dir\_d 為 1，next\_dir = ~reg\_dir

若 reset 為 1，next\_state = 「初始狀態」

若 record\_d 為 1，next\_BCD2 = BCD0、next\_BCD3 = BCD1

state 為「計算中」

根據 reg\_dir 及判斷 count 是否在範圍內

若 reg\_dir == 1 && count < 99，next\_count 設為 count + 1

若 reg\_dir == 0 && count > 0，next\_count 設為 count - 1

若 en\_d 為 1，next\_state =「暫停」

state 為「暫停」

若 en\_d 為 1，next\_state =「計算中」

* + - 1. 當 reg\_dir & count == 99，assign max 為 1，其餘為 0
      2. 當 ~reg\_dir & count == 0，assign min 為 1，其餘為 0
         1. 由於 state 為「初始狀態」時，reg\_dir 恆為 0，無法用 dir\_d 控制，所以 min 會等於 0，即 LED 燈泡不會亮起
  1. lab4\_3.v
     1. 設計概念
        1. 設計一個 4-digit 的倒數計時器
        2. 左邊兩個 digit 為分鐘數，右邊兩個為秒數，使用 60 進位
        3. 由 switch 控制 mode，共有兩種 mode
        4. 在 setting mode 時，可以按下按鈕增加一分鐘或是一秒鐘
        5. 在 counting mode 時，數字開始往下倒數，可以使用按鈕停止倒數及繼續倒數
     2. 實作方法
        1. 宣告變數 reg [11:0] count，紀錄目前的秒數
           1. 因為範圍為 00:00~59:59，換算成秒數後，範圍為 0~3599
           2. 11 < < 12 ，所以 count 必須宣告為 12 個 bits
        2. 宣告變數 wire [3:0] BCD0, BCD1, BCD2, BCD3
           1. BCD0 為秒數的個位數，assign 為 count%60%10
           2. BCD1 為秒數的十位數，assign 為 count%60/10
           3. BCD2 為分鐘數的個位數，assign 為 count/60%10
           4. BCD3 為分鐘數的十位數，assign 為 count/60/10
        3. 將由按鈕獲得的訊號 en、min\_plus、sec\_plus 分別接上 debounce、onepulse 及 delay\_signal
           1. debounce 使用 clk\_13，onepulse 使用 clk\_23，delay\_signal 使用 clk\_23
           2. 在初始狀態無法使用這三個控制訊號
           3. 經過三個 module 後的訊號稱為 xxx\_d
        4. 宣告兩個 always block
           1. 一個由 clk\_25 trigger，將 next\_xxx 訊號送給 state 及 count
           2. 另一個為 always@\*

最上方有 next\_count = count 及 next\_state = state

若按下 reset，next\_count = 12’b0

以 case 判斷 state 的數值，有三種狀態，分別為「設定」、「計算」及「暫停」

若 state 為「設定」

判斷是否有按下 min\_plus、sec\_plus 或是兩個按鈕皆按下，再判斷 count 是否有在範圍內，將 next\_count 設為 count + 61、count + 1 或 count + 60

若 mode 為 1，next\_state = 「暫停」

若 state 為「暫停」

若 mode 為 1，當 en 被按下後，next\_state =「計算」

若 mode 為 0，則 next\_state =「設定」，且 next\_count 設為 12’b0

若 state 為「計算」

若 mode 為 1

當 count > 0，next\_count = count - 1

按下 en 或是 reset，next\_state =「暫停」

若 mode 為 0，則 next\_state =「設定」，且 next\_count 設為 12’b0

* 1. lab4\_bonus.v
     1. 設計概念

將 lab4\_3 的倒數計時器的倒數速度設計成準確的一秒鐘

* + 1. 實作方法
       1. 修改原始的 clock\_divider，製作原始 clk倍週期的 clk\_div
       2. 26 < < 27 ，所以要宣告一個 27 個 bits 的變數 reg [26:0] count，用來計算 input clk 已經過了幾個週期
       3. always block 由 posedge clk trigger
       4. 若 count =，下一個 count 回到 0 重新計算
       5. 其他狀況則 count = count + 1
       6. 若 count < ，clk\_div = 0，其餘 clk\_div = 1’b1

1. 學到的東西與遇到的困難

* 在設計 lab4\_bonus 時，需要秒的 clock 來控制程式狀態的更新，但由於 parameter 必須為整數，呼叫 module 時不能寫 #(0.25)

必須將新的 clk\_div 中 的數值皆乘上 4，呼叫 module 時寫 #(4) 可以得到週期為剛好 1 秒的 clock，寫 #(1) 可以得到週期為秒的 clock

* 原本有對 en 宣告一個 reg\_en 來記錄目前的狀態，但後來了解到其實這就是 state 的概念，就將程式設計成如 FSM 的寫法

1. 想對老師或助教說的話

* 老師與助教在上週 lab3 Demo 時給了我幫助，點出按鈕訊號不流暢的問題、找出可能的錯誤，並給我很多的時間修改、詢問問題，謝謝！