|  |
| --- |
| **EECS 2070 02 Digital Design Labs 2019**  **Lab 6** |
| **學號：107000115 姓名：林珈卉** |

1. 實作過程
   1. 設計概念
      1. 製作出一款猜數字的遊戲
      2. 數字為「隨機」生成，在 0 與 99 之間
      3. 一開始會顯示「－－－－」，直到按下 Start 按鈕進入遊戲
      4. 在進入遊戲後，左邊兩個 Digit 顯示左界，右邊兩個顯示右界
      5. 按下鍵盤可以輸入猜測的數字
         1. 輸入第一個數字時，只有最右邊的 digit 會顯示輸入的數字，其他 digit 則不亮
         2. 輸入第二個數字後，右邊兩個 digit 會顯示輸入的數字，左邊兩個不亮
         3. 輸入超過兩個數字，則只會記憶最後輸入的兩個數字
         4. 按下 Enter 鍵可以將輸入的數字送進程式檢查是否符合答案
      6. 判斷輸入的數字是否與答案相同
         1. 若不相同則縮小猜測範圍
         2. 若相同則勝利
      7. 勝利畫面的七段顯示器會顯示答案數字（左右各一組），16 顆 LED 燈會亮起，以 的週期顯示一個 cycle
   2. 實作方法
      1. 製造隨機的數字

宣告 reg 變數 count、next\_count，隨著 posedge clk 往上加 1，若 count 為 97，則 next\_count 設為 0，這樣就可以將 count 的數值設定在 0~97 的範圍之間，屆時要設 answer 的數值時，就可以直接設為 count+1

* + 1. 設計五個 state，分別為 init、range、keyin1、keyin2、win
    2. 將 rst、start 訊號接上 debounce module 跟 one\_pulse module，分別命名為 rst\_1 以及 start\_1
    3. cheat 訊號只要接上 debounce module，並命名為 cheat\_d
    4. 宣告 wire 變數 clk\_FSM，若 state 為 win，clk\_FSM = clk\_25，若為其他 state，則 clk\_FSM = clk\_16
    5. 製作三個 always block
       1. 一個負責將要顯示的數字訊號傳送給 seven segment module
          1. 宣告 4 個 reg [3:0] 變數 BCD0, BCD1, BCD2, BCD3，在不同 state 的情況下賦予不同的值
          2. 宣告一個 reg 變數 dig\_on，負責將要亮的 digit 傳給 seven segment module
          3. init state

四個 digit 皆顯示「—」，四個皆亮

* + - * 1. range state

判斷是否有按下cheat 按鈕 (cheat\_d == 1’b1)

按下 cheat 按鈕則左邊兩個 digit 顯示答案，右邊兩個不亮

若沒有按下 cheat 則左邊兩個顯示左界，右邊兩個顯示右界，四個皆亮

* + - * 1. keyin1 state

最右邊的 digit 顯示輸入的數字，左邊三個不亮

* + - * 1. keyin2 state

最右邊兩個 digit 顯示輸入的數字，左邊兩個不亮

* + - * 1. win state

左右皆顯示答案的數字，四個皆亮

* + - 1. 一個以 posedge rst\_1, posedge clk\_FSM trigger，用來更新 state、lnum、rnum（左右邊界的值）
      2. 一個 always block 則負責改變 next\_xxx 的值
         1. 最上方先寫上 next\_xxx = state、lnum、rnum，以應付預設狀況
         2. 再來用 case (state) 來判斷現在在哪一個 state

init state

先判斷是否有按下 start 按鈕，若無則不做事

if (start\_1)

將 next\_state 設成 range

next\_lnum 設為 0，next\_rnum 設為 99

設置 answer 的數值

range state

從 sample display module 拉出訊號以辨認是否按下按鍵，若無則不做事

若按下按鍵則 next\_state = keyin1

keyin1 state

判斷是否按下 enter，若按下則 next\_state 回到 range，並不改變邊界的數值

若沒有按下 enter 再辨認是否按下按鍵，若無則不做事

若按下按鍵則 next\_state = keyin2

keyin2 state

判斷是否按下 enter 鍵，若無則不做事

若按下 enter，next\_state 回到 range，並判斷輸入的數字大小

若數字在範圍之外，則不改變邊界的數值

若數字在範圍之內，則確認是否等於答案

等於答案的狀況，next\_state = win

大於答案的狀況，將 next\_rnum 設為輸入的數字

小於答案的狀況，將 next\_lnum 設為輸入的數字

win state，則 next\_state = init

1. 學到的東西與遇到的困難

* 由於將數字送給 seven segment 時，是傳入 BCD 的數值，若邊界、答案跟輸入的數字有些是使用十進位去存、有些是使用 BCD，就必須做轉換，而這中間很容易出錯，導致賦值時出問題，較好的方法是都使用 BCD，減少轉換的問題。
* Demo 時，助教提醒我的 Clock selector 會導致 Clock 不穩定，應該要使用 Counter 來處理從 Win 回到 Init 的 State 轉換。
* 由於 Keyboard Decoder 是在頻率為 100M 的 clk 下運行， 用來更新的 always block 也應該用 clk 去 trigger，而不是 clk\_16。
* 額外再寫出一個 module 並將 sample display 的訊號拉出來做判斷，是比較複雜且容易出錯的方法，較適合的方法是修改 sample display 的內容，將裡面 always block 判斷數字變化的部份拉進 range state，並只需要用到兩個 always block。

1. 想對老師或助教說的話

助教辛苦了！除了協助同學找出問題、Debug，還要忍受閱讀奇奇怪怪的 coding style，也很感謝助教告訴我我的大問題所在，還有告訴我其實只要按 Generate Bitstream 並把 Number of Jobs 拉到最高就可以了......