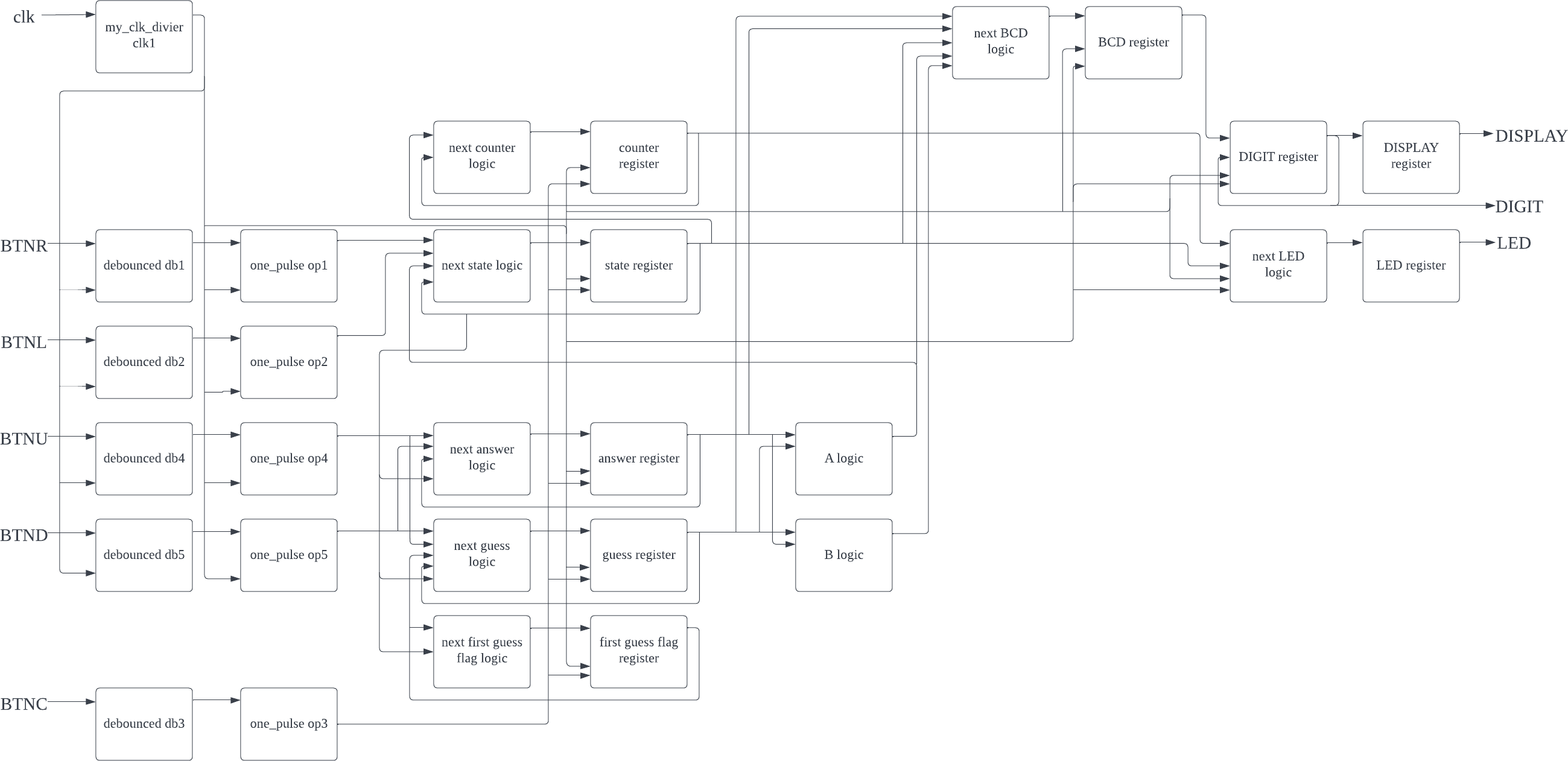
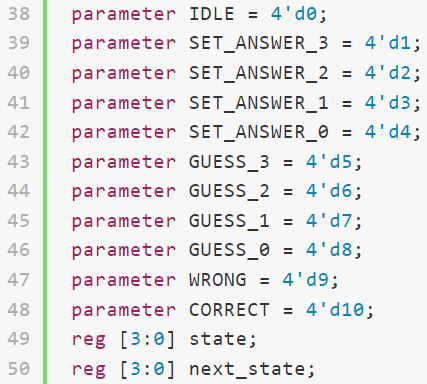
|  |  |
| --- | --- |
|  | **Lab 6** |
| 學號: 110062131 | 姓名: 馬毓昇 |

# Lab Implementation

* 1. Block diagram of the design with explanation



* + 1. 每個 input button 都用debounced 跟 one\_pulse 處理。
    2. FSM 部分有 state 處理 state 間的變化；counter 作為計時器使用；answer 讓玩家調整每一位數；guess 也讓玩家調整每一位數；first guess flag 判斷是否重新進到 GUESS state 讓guess 歸 0；BCD 製作輸出用的 BCD；LED 輸出。
    3. FSM 以外的 A、B logic 部分直接計算幾 A 幾 B，DIGIT 與 DISPLAY 搭配來做七段顯示。
  1. Partial code screenshot with the explanation: you don’t need to paste the entire code into the report. Just explain the kernel part.
     1. 設計了 11 個 states，把 SET\_ANSWER 依據位數分成 4 個狀態，GUESS 也同樣分成 4 個狀態。



* + 1. wire A、B。A := (正解第一位 == 猜第一位) + (正解第二位 == 猜第二位) + (正解第三位

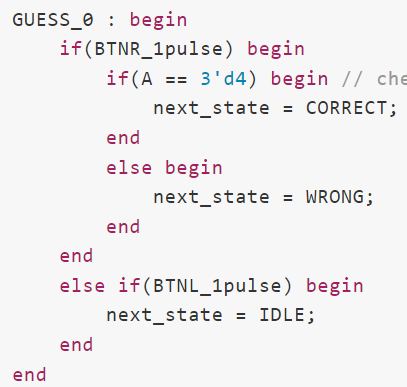
== 猜第三位) + (正解第四位 == 猜第四位)，每個括號內會回傳 0 or 1，所以如果 4 位都相等 A 就是 4，3 位相等 A 就是 3，以此類推。B := (正解第一位 == 猜其他位) + (正解第

二位 == 猜其他位) + (正解第三位 == 猜其他位) + (正解第四位 == 猜其他位)，因為保

證不會出現重複的數字，所以每個括號只會回傳 0 or 1，與 A 同理，這樣就能夠計算出有多少 B。此外，有用到 concat 的方式來把 0 or 1 擴成 4bit 來做加法計算。

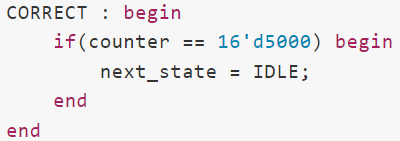


* + 1. GUESS 到最後一位數按下 OK 時用 A 來判斷下一個 state 要進 CORRECT 還是 WRONG。



* + 1. CORRECT 時等到 counter 數到 5000(因為整個 FSM 是用 1000 hz 在跑)才讓下個 state 回到

IDLE，達成在 CORRECT 等待 5 秒的效果。

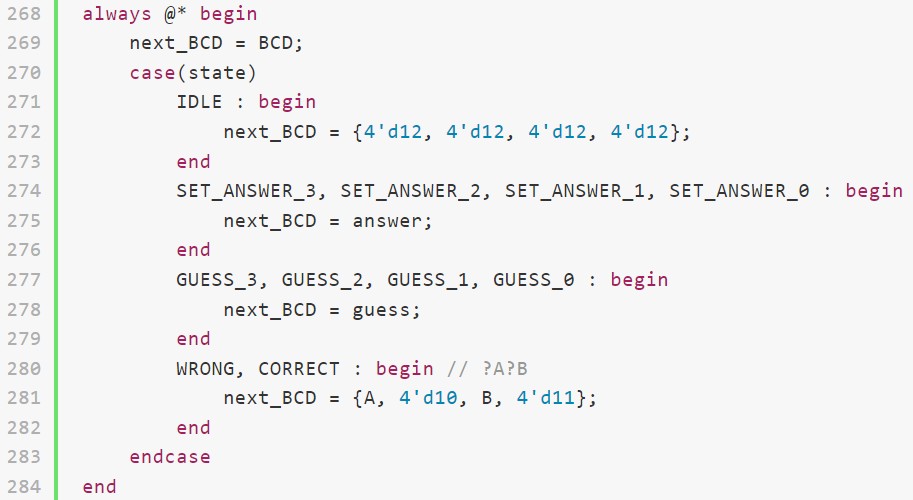


* + 1. 7 段顯示器多支援顯示 A、B 與一橫。

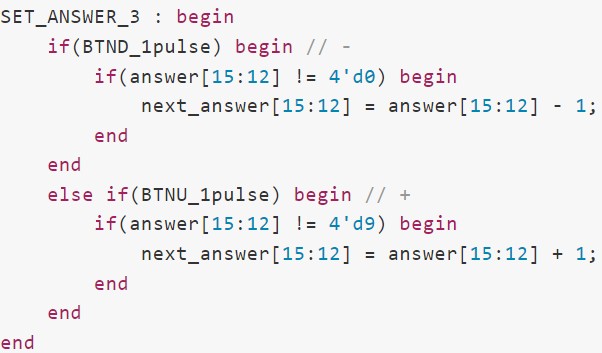


* + 1. BCD 在 IDLE 時顯示四條橫、SET\_ANSWER 時顯示 answer、GUESS 時顯示 guess、WRONG

與 CORRECT 時顯示幾 A 幾 B。

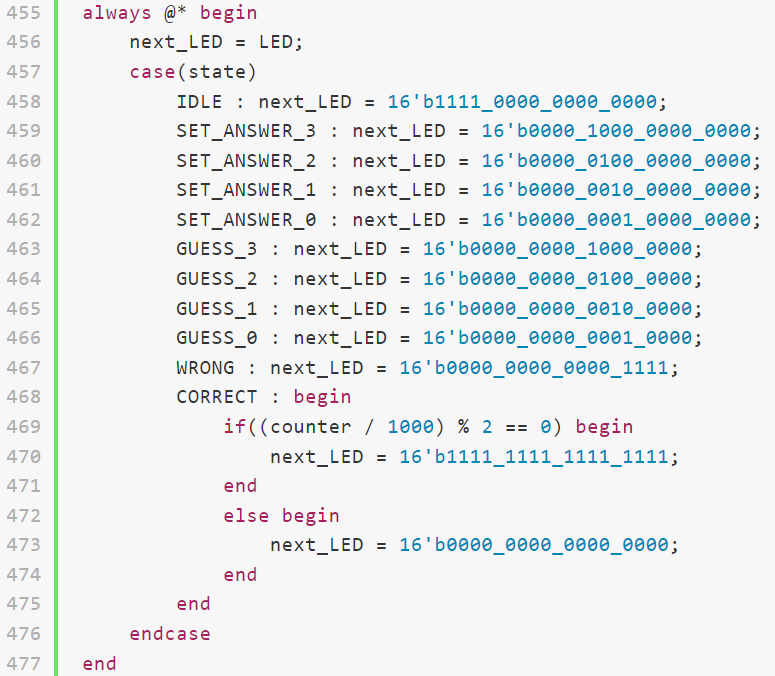


* + 1. 調整位數部分，加的時候判斷這 4bit BCD 是不是上限(9)或是在減的時候判斷是不是下限(0)，如果不是就加一或減一。這裡用 SET\_ANSWER\_3 舉例，其他位數的調整與 GUESS 皆同理。

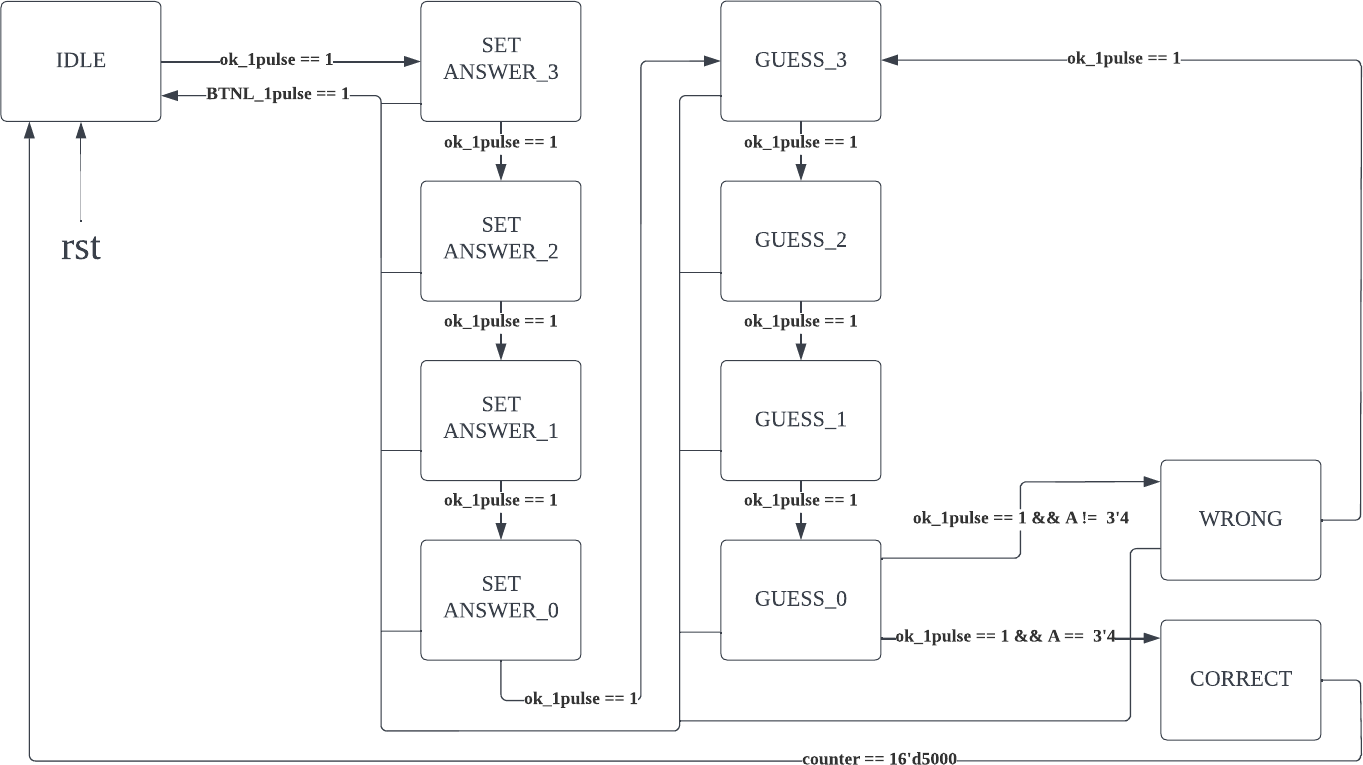


* + 1. 依據設計的 11 個 state 來顯示 LED，在 CORRECT 額外判斷 counter 的千位述其偶來達成

LED 1 hz 閃爍的效果。



* 1. Event-based finite state machine (FSM) with the explanation:



共有 11 個 state: IDLE，每個位數的 SET\_ANSWER，每個位數的 GUESS，WRONG， CORRECT。

1. IDLE -> SET\_ANSWER\_3: 按下 OK。
2. SET\_ANSWER\_3 -> SET\_ANSWER\_2: 按下 OK。
3. SET\_ANSWER\_2 -> SET\_ANSWER\_1: 按下 OK。
4. SET\_ANSWER\_1 -> SET\_ANSWER\_0: 按下 OK。
5. SET\_ANSWER\_0 -> GUESS\_3: 按下 OK。
6. GUESS\_3 -> GUESS\_2: 按下 OK。
7. GUESS\_2 -> GUESS\_1: 按下 OK。
8. GUESS\_1 -> GUESS\_0: 按下 OK。
9. GUESS\_0 -> WRONG: 按下 OK 但猜錯。
10. GUESS\_0 -> CORRECT: 按下 OK 且猜對。
11. WRONG -> GUESS\_3: 按下 OK。
12. CORRECT -> IDLE: 等 5 秒之後直接轉化。
13. SET\_ANSWER\_3 -> IDLE: 按下 CANCEL。
14. SET\_ANSWER\_2 -> IDLE: 按下 CANCEL。
15. SET\_ANSWER\_1 -> IDLE: 按下 CANCEL。
16. SET\_ANSWER\_0 -> IDLE: 按下 CANCEL。
17. GUESS\_3 -> IDLE: 按下 CANCEL。
18. GUESS\_2 -> IDLE: 按下 CANCEL。
19. GUESS\_1 -> IDLE: 按下 CANCEL。
20. GUESS\_3 -> IDLE: 按下 CANCEL。
21. WRONG -> IDLE: 按下 CANCEL。22 任何 state -> IDLE: 按下 rst。

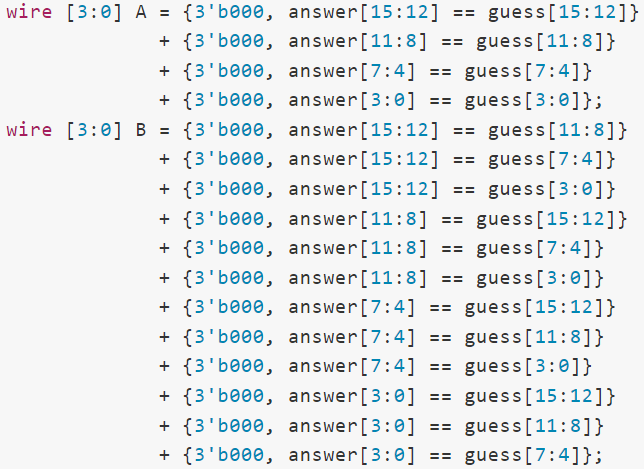
# Questions and Discussions

A: 我有做一個 reg 存 answer 與一個 reg 存 guess，wire A 根據兩個 reg 的內容一直計算有多少A，當猜完最後一位按下 OK 時，如果 A 是 4 代表 answer == guess，那麼就能確認正確答案了。

B: wire A 與 B。

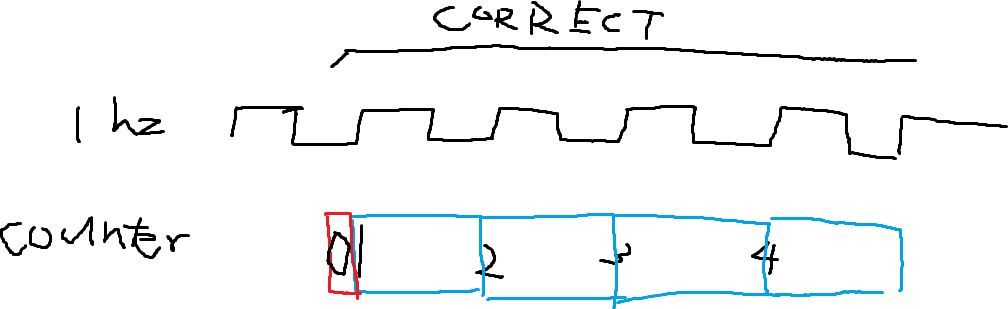
A := (正解第一位 == 猜第一位) + (正解第二位 == 猜第二位) + (正解第三位 == 猜第三位) + (正解第四位 == 猜第四位)，每個括號內會回傳 0 or 1，所以如果 4 位都相等 A 就是 4，3 位相等A 就是 3，以此類推。

B := (正解第一位 == 猜其他位) + (正解第二位 == 猜其他位) + (正解第三位 == 猜其他位) + (正解第四位 == 猜其他位)，因為保證不會出現重複的數字，所以每個括號只會回傳 0 or 1，與 A 同理，這樣就能夠計算出有多少 B。此外，有用到 concat 的方式來把 0 or 1 擴成 4bit 來做加法計算。



C: 我的 FSM 是以 1000 hz 的 clk 在跑，所以我設計了一個 counter，當 state 進到 CORRECT 時， counter 就從 0 開始數，0~999 代表經過 1000 個 cycle 就相當於經過了 1 秒鐘，1000~1999、2000~2999……接下來以此類推。再著我以 counter 的千位數奇偶來判斷 LED 該亮或暗，在控制LED 的部分，當 state 是 CORRECT 時判斷(counter/1000)%2，如果是 0 就表示千位數是偶數，1 是奇數，偶數的情況就讓 LED 亮，奇數就讓 LED 暗，如此一來便能達成讓 LED 以 1 hz 閃爍的效果。

# Problem Encountered

* 1. 我本來是用 100M/2^16 hz 來跑FSM，然後額外用 1hz 來數 counter，不過這樣就會出現一個問題：當 state 進到 CORRECT 時，有可能還卡在 1 hz clk 的週期中，造成來的第一個 posedge 會直接讓 counter 變成 1，也就是說 0 的階段會跑不滿整整 1 秒，這樣雖然也能讓 LED 有亮暗變化，但就是第一個亮的時間會不滿 1 秒鐘，不符合規格要求。

後來我把整個 FSM 用 1000 hz 跑，counter 就取千位數來關注，這樣既能讓 LED 有亮暗變化， 還能符合規格要求以 1 秒鐘為間隔閃爍。

# Suggestions

期中考好難，希望可以調分…