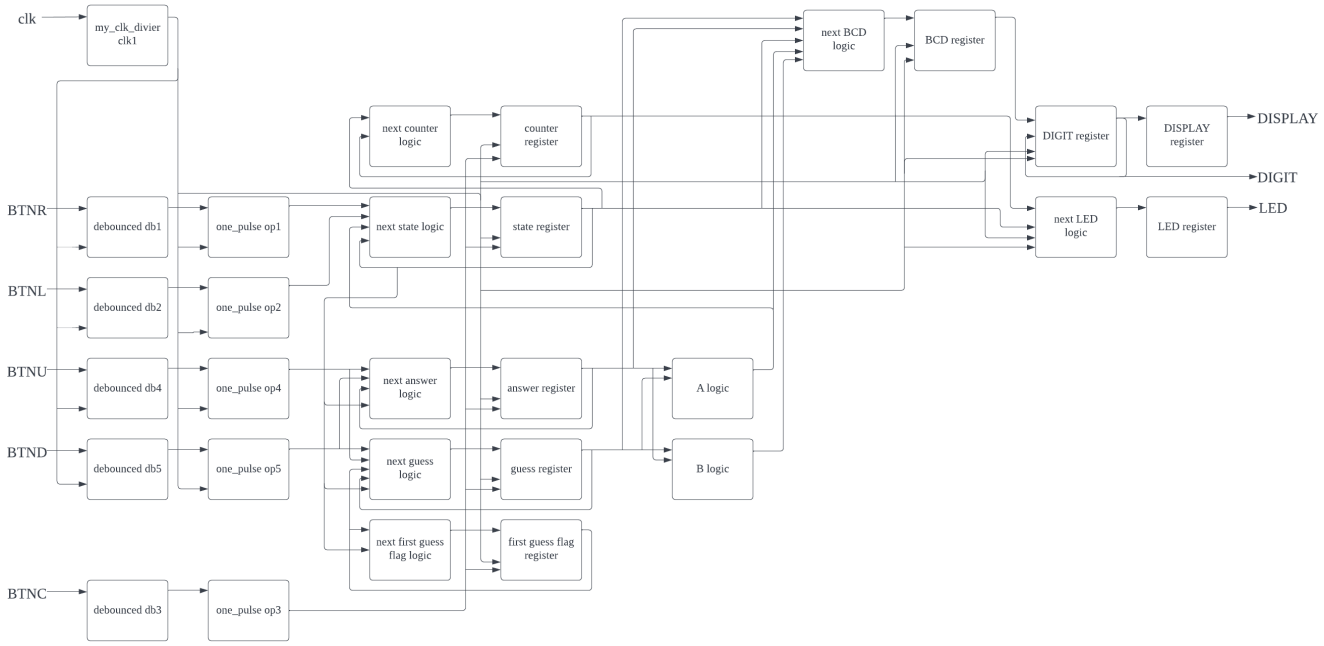
Lab 5

學號: 110062131 姓名: 馬毓昇

A. Lab Implementation

1. Block diagram of the design with explanation



a. 每個 input button 都用 debounced 跟 one_pulse 處理。

b. 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。

c. 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。

2. Partial code screenshot with the explanation: you don't need to paste the entire code into the report. Just explain the kernel part.

a. 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。

```
38  parameter IDLE = 4'd0;
39  parameter SET_ANSWER_3 = 4'd1;
40  parameter SET_ANSWER_2 = 4'd2;
41  parameter SET_ANSWER_1 = 4'd3;
42  parameter SET_ANSWER_0 = 4'd4;
43  parameter GUESS_3 = 4'd5;
44  parameter GUESS_2 = 4'd6;
45  parameter GUESS_1 = 4'd7;
46  parameter GUESS_0 = 4'd8;
47  parameter WRONG = 4'd9;
48  parameter CORRECT = 4'd10;
49  reg [3:0] state;
50  reg [3:0] next_state;
```

b. 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。

```
wire [3:0] A = {3'b000, answer[15:12] == guess[15:12]}
             + {3'b000, answer[11:8] == guess[11:8]}
             + {3'b000, answer[7:4] == guess[7:4]}
             + {3'b000, answer[3:0] == guess[3:0]};
wire [3:0] B = {3'b000, answer[15:12] == guess[11:8]}
             + {3'b000, answer[15:12] == guess[7:4]}
             + {3'b000, answer[15:12] == guess[3:0]}
             + {3'b000, answer[11:8] == guess[15:12]}
             + {3'b000, answer[11:8] == guess[7:4]}
             + {3'b000, answer[11:8] == guess[3:0]}
             + {3'b000, answer[7:4] == guess[15:12]}
             + {3'b000, answer[7:4] == guess[11:8]}
             + {3'b000, answer[7:4] == guess[3:0]}
             + {3'b000, answer[3:0] == guess[15:12]}
             + {3'b000, answer[3:0] == guess[11:8]}
             + {3'b000, answer[3:0] == guess[7:4]};
```

c. 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。

```
GUESS_0 : begin
    if(BTNR_1pulse) begin
        if(A == 3'd4) begin // che
            next_state = CORRECT;
        end
        else begin
            next_state = WRONG;
        end
    end
    else if(BTNL_1pulse) begin
        next_state = IDLE;
    end
end
```

d. 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。

```
CORRECT : begin
    if(counter == 16'd5000) begin
        next_state = IDLE;
    end
end
```

e. 7 段顯示器多支援顯示 A、B 與一橫。

```
// 7 segment display
always @* begin
    case(value)
            4'd0 : DISPLAY = 7'b100_0000;
            4'd1 : DISPLAY = 7'b111_1001;
            4'd2 : DISPLAY = 7'b010_0100;
            4'd3 : DISPLAY = 7'b011_0000;
            4'd4 : DISPLAY = 7'b001_1001;
            4'd5 : DISPLAY = 7'b001_0010;
            4'd6 : DISPLAY = 7'b000_0010;
            4'd7 : DISPLAY = 7'b111_1000;
            4'd8 : DISPLAY = 7'b000_0000;
            4'd9 : DISPLAY = 7'b001_0000;

            4'd10 : DISPLAY = 7'b000_1000; // A
            4'd11 : DISPLAY = 7'b000_0011; // B
            4'd12 : DISPLAY = 7'b011_1111; // -

            default : DISPLAY = 7'b111_1111;
        endcase
end
```

f. 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。

```
always @* begin
    next_BCD = BCD;
    case(state)
        IDLE : begin
            next_BCD = {4'd12, 4'd12, 4'd12, 4'd12};
        end
        SET_ANSWER_3, SET_ANSWER_2, SET_ANSWER_1, SET_ANSWER_0 : begin
            next_BCD = answer;
        end
        GUESS_3, GUESS_2, GUESS_1, GUESS_0 : begin
            next_BCD = guess;
        end
        WRONG, CORRECT : begin // ?A?B
            next_BCD = {A, 4'd10, B, 4'd11};
        end
    endcase
end
```

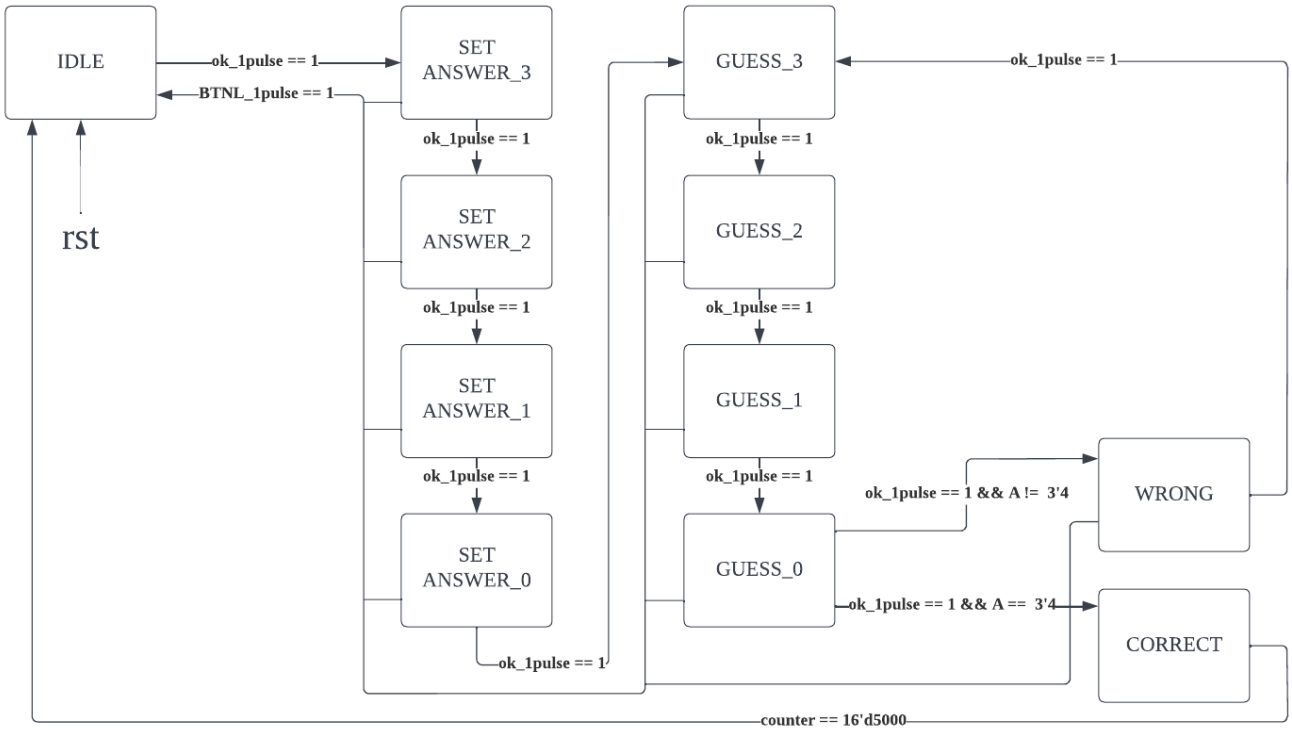
g. 調整位數部分，加的時候判斷這 4bit BCD 是不是上限(9)或是在減的時候判斷是不是下限(0)，如果不是就加一或減一。這裡用 SET_ANSWER_3 舉例，其他位數的調整與 GUESS 皆同理。

```
SET_ANSWER_3 : begin
    if(BTND_1pulse) begin // -
        if(answer[15:12] != 4'd0) begin
            next_answer[15:12] = answer[15:12] - 1;
        end
    end
    else if(BTNU_1pulse) begin // +
        if(answer[15:12] != 4'd9) begin
            next_answer[15:12] = answer[15:12] + 1;
        end
    end
end
```

h. 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。

```
always @* begin
    next_LED = LED;
    case(state)
        IDLE : next_LED = 16'b1111_0000_0000_0000;
        SET_ANSWER_3 : next_LED = 16'b0000_1000_0000_0000;
        SET_ANSWER_2 : next_LED = 16'b0000_0100_0000_0000;
        SET_ANSWER_1 : next_LED = 16'b0000_0010_0000_0000;
        SET_ANSWER_0 : next_LED = 16'b0000_0001_0000_0000;
        GUESS_3 : next_LED = 16'b0000_0000_1000_0000;
        GUESS_2 : next_LED = 16'b0000_0000_0100_0000;
        GUESS_1 : next_LED = 16'b0000_0000_0010_0000;
        GUESS_0 : next_LED = 16'b0000_0000_0001_0000;
        WRONG : next_LED = 16'b0000_0000_0000_1111;
        CORRECT : begin
            if((counter / 1000) % 2 == 0) begin
                next_LED = 16'b1111_1111_1111_1111;
            end
            else begin
                next_LED = 16'b0000_0000_0000_0000;
            end
        end
    endcase
end
```

3. 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。

B. 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。

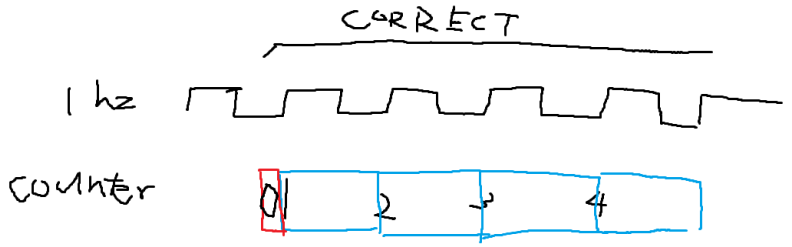
B := (正解第一位 == 猜其他位) + (正解第二位 == 猜其他位) + (正解第三位 == 猜其他位) + (正解第四位 == 猜其他位)，因為保證不會出現重複的數字，所以每個括號只會回傳 0 or 1，與 A 同理，這樣就能夠計算出有多少 B。此外，有用到 concat 的方式來把 0 or 1 擴成 4bit 來做加法計算。

```
wire [3:0] A = {3'b000, answer[15:12] == guess[15:12]}
             + {3'b000, answer[11:8] == guess[11:8]}
             + {3'b000, answer[7:4] == guess[7:4]}
             + {3'b000, answer[3:0] == guess[3:0]};
wire [3:0] B = {3'b000, answer[15:12] == guess[11:8]}
             + {3'b000, answer[15:12] == guess[7:4]}
             + {3'b000, answer[15:12] == guess[3:0]}
             + {3'b000, answer[11:8] == guess[15:12]}
             + {3'b000, answer[11:8] == guess[7:4]}
             + {3'b000, answer[11:8] == guess[3:0]}
             + {3'b000, answer[7:4] == guess[15:12]}
             + {3'b000, answer[7:4] == guess[11:8]}
             + {3'b000, answer[7:4] == guess[3:0]}
             + {3'b000, answer[3:0] == guess[15:12]}
             + {3'b000, answer[3:0] == guess[11:8]}
             + {3'b000, answer[3:0] == guess[7:4]};
```

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。

C. 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。

D. **Suggestions**

期中考好難，希望可以調分…