Lab09_Keyboard

106061146 陳兆廷

1.1 輸入鍵盤的 0~9 並輸出至七段顯示器；

Keyboard.v 首先要讀取鍵盤由 USB 接口輸入的數值，並判斷其輸入的為何種按鍵。USB 輸入值有 USB_CLK 及 USB_DATA，USB_CLK 為每一個 DATA 的值輸出的週期，而 DATA 則是 11 個為一組的值，而第 8 位至第 1 位是可以判讀的按鍵值，由兩個 16 進位的數值表示，而最後一位則是 odd parity，用以檢測前面讀得的數值是否正確，因此我設計每當 USB_CLK 改變時，USB_DATA 就存入 scan_code 並向左 shift 一位，並計算輸入 11 位後，開始存入所要判讀的值，並且判斷此讀值是否正確，如圖(一)，當輸入完成後依照講義所寫的各個按鍵的 16 進位配置，比較後輸出至七段顯示器，如圖(三)，而後再以大 module 輸出至 ssd、ssd_clt 即可。

1.2 輸入鍵盤的 A、S、M 並輸出至七段顯示器，增加 enter 刷新功能；

此題方法同上題，只需增加選項 A、S、M、enter，並在輸入 enter 時輸出使得 ssd 全部關閉的值即可，如圖(四)。

```
if (PS2_CLK != PREVIOUS_STATE) begin            //if the state of Clock pin changed from previous state
    if (!PS2_CLK) begin                         //and if the keyboard clock is at falling edge
        read <= 1;                              //mark down that it is still reading for the next bit
        scan_err <= 0;                          //no errors
        scan_code[10:0] <= {PS2_DATA, scan_code[10:1]}; //add up the data received by shifting bits and addin
        COUNT <= COUNT + 1;                     //
    end
end
else if (COUNT == 11) begin                     //if it already received 11 bits
    COUNT <= 0;
    read <= 0;                                  //mark down that reading stopped
    TRIG_ARR <= 1;                              //trigger out that the full pack of 11bits was received
    //calculate scan_err using parity bit
    if (!scan_code[10] || scan_code[0] || !(scan_code[1]^scan_code[2]^scan_code[3]^scan_code[4]
        ^scan_code[5]^scan_code[6]^scan_code[7]^scan_code[8]
        ^scan_code[9]))
        scan_err <= 1;
    else
        scan_err <= 0;
end
```

圖(一)

```
if (TRIG_ARR) begin             //and
    if (scan_err) begin         //BUT
        CODEWORD <= 8'd0;       //then
    end
    else begin
        CODEWORD <= scan_code[8:1]; /
    end                         //notice, that the code
end                             //is supposed to be the
else CODEWORD <= 8'd0;          /
```

圖(二)

```
if (CODEWORD == ONE) NUM <= 4'd1;
  else if (CODEWORD == TWO) NUM <= 4'd2;
  else if (CODEWORD == THREE) NUM <= 4'd3;
  else if (CODEWORD == FOUR) NUM <= 4'd4;
  else if (CODEWORD == FIVE) NUM <= 4'd5;
  else if (CODEWORD == SIX) NUM <= 4'd6;
  else if (CODEWORD == SEVEN) NUM <= 4'd7;
  else if (CODEWORD == EIGHT) NUM <= 4'd8;
  else if (CODEWORD == NINE) NUM <= 4'd9;
  else if (CODEWORD == ZERO) NUM <= 4'd0;
```

圖(三)

```
else if (CODEWORD == A) ASM <= 4'd0;
else if (CODEWORD == S) ASM <= 4'd1;
else if (CODEWORD == M) ASM <= 4'd2;
else if (CODEWORD == ENTER) begin ASM <= 4'd3; NUM <= 4'd15; end
```

圖(四)

2.1 個位數加法器：

同第一題之方法讀取輸入值後，以類似 FSM 的方法，定義 state，第一個數字輸入完用加號作為進入下個 state 的 trigger，輸入第二個數字後以 enter 輸出答案。如圖(五)。

3.1 二位數計算機：

同第一、二題之方法，只需要再多幾個 state，我設計輸入的兩個二位

數會在左邊兩位及右邊兩位的七段顯示器顯示，輸入完兩個數字後，按下運算子符號，再輸入兩個數字，按下 enter 便可以將計算出來的值 assign 至每一位的七段顯示器。

碰到的難題：按鍵按下瞬間，FPGA 會讀到很多個相同的值，因為按下的那瞬間其實讀進去許多值，因此我在測試時常常一次就輸入了兩個數字，這時我將一個 freqency_divider 加入，只有在 on==0 時才會讀值，並當按下一個鍵後，將 on(判案值)變成 1，當經過一定時間(一秒)後 on 才會變回 0，因此只要在 1 秒內輸入的值就只會讀取一次，才解決了問題。

4.1 加入 caps，輸出大小寫的英文字母之 ASCII code：

需要兩個 state，一個是小寫，按下 caps 後進入下個 state，輸出值為大寫，再按下 caps 回到小寫 state。這題如第 3 題同樣有按下 caps 會快速跳回 state 0 的狀況，因此也需要多設一個變數，當在 caps state(state 1)下一定要至少按下另一個按鍵，才會允許回到 state 0。

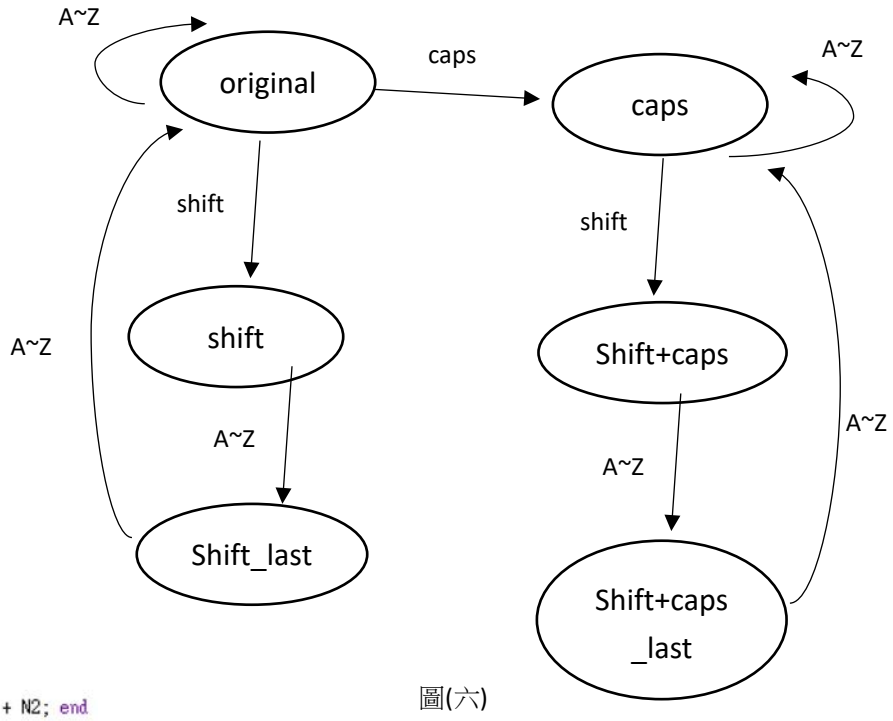
4.2 加入 shift，在當下的 caps 狀態下，輸出一個按鍵的大寫或小寫：

這時候需要更多的 state，包括上一個的 caps，還有這題的 shift、shift+caps，另外為了使在 shift 狀態下繼續顯示的 shift_last、shift+caps_last。Last 狀態的作用在於當按下 shift 加上一個字母後，要直到下一個按鍵跟上一個按鍵不一樣，才會跳離 shift 的狀態，回到原狀態。state diagram 如圖(六)。

這題在思考 state 時思考許久，也在 shift 狀態下無法跳離等狀況困擾許久，最後想出再多加一個 state 解決問題。

```
else if(state == 2'd0)begin
   if (CODEWORD == ONE) N1 <= 8'd1;
   else if (CODEWORD == TWO) N1 <= 8'd2;
   else if (CODEWORD == THREE) N1 <= 8'd3;
   else if (CODEWORD == FOUR) N1 <= 8'd4;
   else if (CODEWORD == FIVE) N1 <= 8'd5;
   else if (CODEWORD == SIX) N1 <= 8'd6;
   else if (CODEWORD == SEVEN) N1 <= 8'd7;
   else if (CODEWORD == EIGHT) N1 <= 8'd8;
   else if (CODEWORD == NINE) N1 <= 8'd9;
   else if (CODEWORD == ZERO) N1 <= 8'd0;
   else if (CODEWORD == PLUS)begin state <= 2'd1; end
end
else if(state == 2'd1)begin
   if (CODEWORD == ONE) N2 <= 8'd1;
   else if (CODEWORD == TWO) N2 <= 8'd2;
   else if (CODEWORD == THREE) N2 <= 8'd3;
   else if (CODEWORD == FOUR) N2 <= 8'd4;
   else if (CODEWORD == FIVE) N2 <= 8'd5;
   else if (CODEWORD == SIX) N2 <= 8'd6;
   else if (CODEWORD == SEVEN) N2 <= 8'd7;
   else if (CODEWORD == EIGHT) N2 <= 8'd8;
   else if (CODEWORD == NINE) N2 <= 8'd9;
   else if (CODEWORD == ZERO) N2 <= 8'd0;
   else if (CODEWORD == ENTER)begin state <= 2'd0; S <= N1 + N2; end
end
```

圖(五)



圖(六)