Lab 5 Report

Advanced 1：

Dec1 偵測到 101 且前面沒有 1111 的時候就輸出 1

Dec2 偵測到 1101 就輸出 1

d1_state, d2_state 是 dec1,dec2 當前的 state

d1_nstate, d2_nstate 是 dec1,dec2 下一個 state

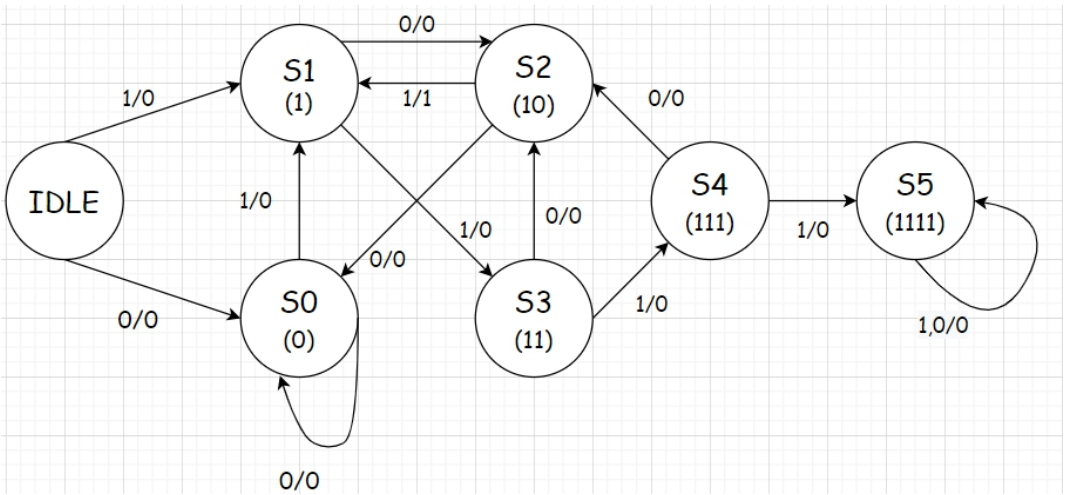
```
21 always @ (posedge clk)begin
22     if(!rst_n)begin
23         d1_state <= IDLE;
24         d1_state <= IDLE;
25     end
26     else begin
27         d1_state <= d1_nstate;
28         d2_state <= d2_nstate;
29     end
30 end
```

```
32 always @ (*)begin
33     case(d1_state)
34         S0:begin
35             dec1 = 0;
36             if(in)
37                 d1_nstate = S1;
38             else
39                 d1_nstate = S0;
40         end
41         S1:begin
42             dec1 = 0;
43             if(in)
44                 d1_nstate = S3;
45             else
46                 d1_nstate = S2;
47         end
48         S2:begin
49             if(in)begin
50                 d1_nstate = S1;
51                 dec1 = 1;
52             end
53             else begin
54                 d1_nstate = S0;
55                 dec1 = 0;
56             end
57         end
```

```
58         S3:begin
59             dec1 = 0;
60             if(in)
61                 d1_nstate = S4;
62             else
63                 d1_nstate = S2;
64         end
65         S4:begin
66             dec1 = 0;
67             if(in)
68                 d1_nstate = S5;
69             else
70                 d1_nstate = S2;
71         end
72         S5:begin
73             dec1 = 0;
74             d1_nstate = S5;
75         end
76         default:begin
77             dec1 = 0;
78             if(in)
79                 d1_nstate = S1;
80             else
81                 d1_nstate = S0;
82         end
83     endcase
```

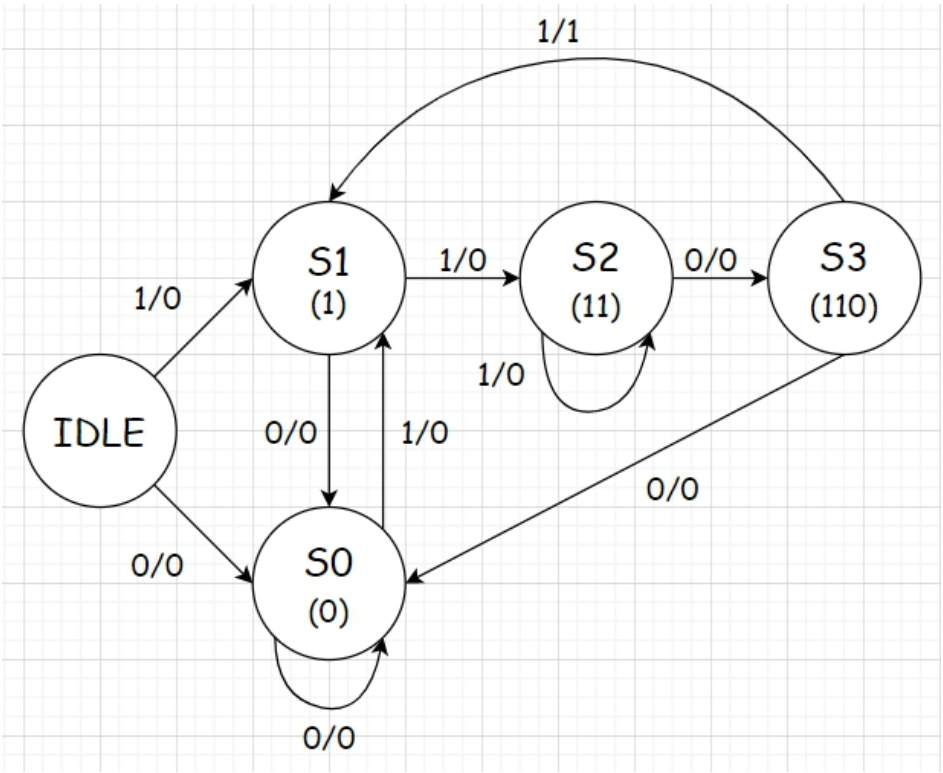
左圖為 sequential 的部分，d1 跟 d2 的 state 會在這裡做改變。

中間、右圖是 dec1 的 Mealy machine，按照下圖 dec1 的 state transition diagram 來編寫。



```
case(d2_state)
    S0:begin
        dec2 = 0;
        if(in)
            d2_nstate = S1;
        else
            d2_nstate = S0;
    end
    S1:begin
        dec2 = 0;
        if(in)
            d2_nstate = S2;
        else
            d2_nstate = S0;
    end
    S2:begin
        dec2 = 0;
        if(in)
            d2_nstate = S2;
        else
            d2_nstate = S3;
    end
    S3:begin
        if(in)begin
            d2_nstate = S1;
            dec2 = 1;
        end
        else begin
            d2_nstate = S0;
            dec2 = 0;
        end
    end
    default:begin
        dec2 = 0;
        if(in)
            d2_nstate = S1;
        else
            d2_nstate = S0;
    end
endcase
```

上圖是 dec2 的 Mealy machine. 下圖是 dec2 的 state transition diagram.



Testbench：

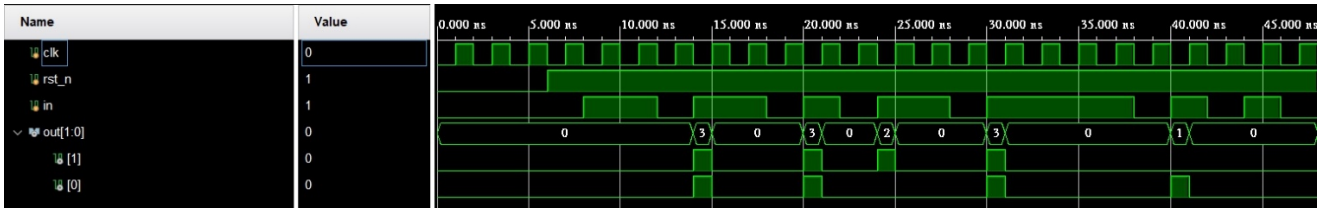
輸入一串 in 來觀察 dec1、dec2 結果是否正確

Out[1]是 dec1 的結果

Out[2]是 dec2 的結果

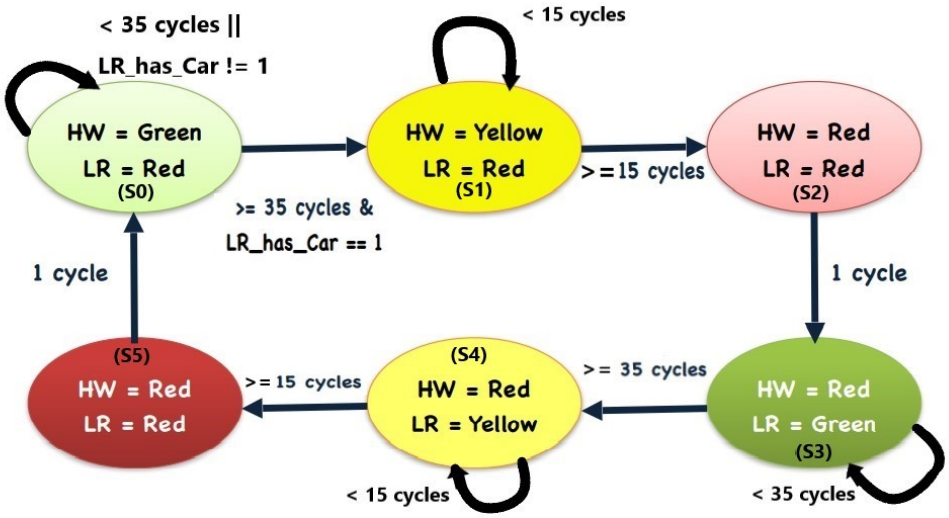
```
#5
@(negedge clk) rst_n = 1;

@(negedge clk) in = 1;
@(negedge clk) in = 1;
@(negedge clk) in = 0;
@(negedge clk) in = 1;
@(negedge clk) in = 1;
@(negedge clk) in = 0;
@(negedge clk) in = 1;
@(negedge clk) in = 0;
@(negedge clk) in = 1;
@(negedge clk) in = 1;
@(negedge clk) in = 0;
@(negedge clk) in = 1;
@(negedge clk) in = 1;
@(negedge clk) in = 1;
@(negedge clk) in = 1;
@(negedge clk) in = 0;
@(negedge clk) in = 1;
@(negedge clk) in = 0;
@(negedge clk) in = 1;
@(negedge clk) in = 0;
@(negedge clk) in = 1;
```

Waveform:



Advanced 2：

FSM：




```
17  reg [2:0] state,nstate,nhw_light,nlr_light;
18  reg [5:0] g_count;
19  reg [5:0] ng_count;
20  reg [3:0] y_count;
21  reg [3:0] ny_count;
```

State 是 HW 和 LR 的當前狀態 nstate 是 HW 和 LR 的下個狀態
g_count(綠燈)、y_count(黃燈)是用來記錄是否達到所需的 cycles
ng_count、ny_count 是下一個 cycles 數

```
23  always @ (posedge clk)begin
24      if(!rst_n)begin
25          g_count <= 0;
26          y_count <= 0;
27          ng_count <= 0;
28          ny_count <= 0;
29          nstate <= S0;
30      end
31      else begin
32          hw_light <= nhw_light;
33          lr_light <= nlr_light;
34          g_count <= ng_count;
35          y_count <= ny_count;
36          state <= nstate;
37      end
38  end
```

sequential 的部分
rst_n 為 0 的時候初始化

rst_n 為 1 的時候就會變成 state 改變狀態
g_count、y_count 也會改變 cycles 數量

下面是 state transition 的部分

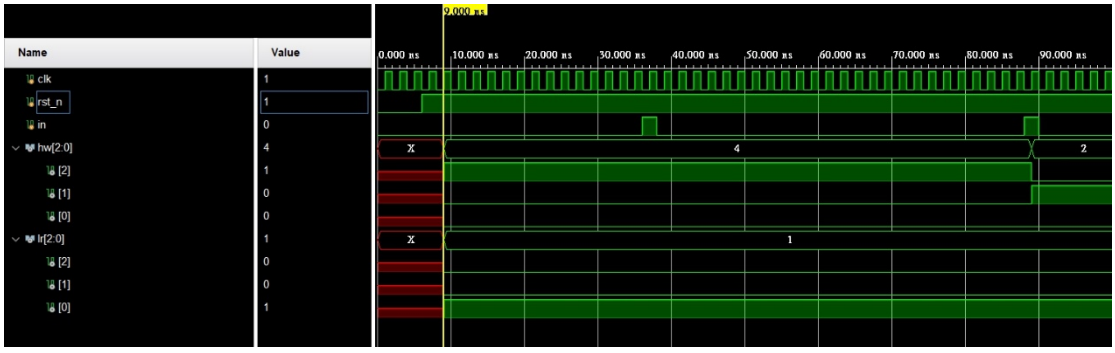
```
40  always @ (*)begin
41      case(state)
42          S0:begin
43              if(g_count >= 34 && lr_has_car)begin
44                  nstate = S1;
45                  ng_count = 0;
46                  nhw_light = 3'b010;
47                  nlr_light = 3'b001;
48              end
49              else begin
50                  nstate = S0;
51                  ng_count = g_count + 1;
52                  nhw_light = 3'b100;
53                  nlr_light = 3'b001;
54              end
55          end
56          S1:begin
57              if(y_count >= 14)begin
58                  nstate = S2;
59                  ny_count = 0;
60                  nhw_light = 3'b001;
61                  nlr_light = 3'b001;
62              end
63              else begin
64                  nstate = S1;
65                  ny_count = y_count + 1;
66                  nhw_light = 3'b010;
67                  nlr_light = 3'b001;
68              end
69          end
70          S2:begin
71              nstate = S3;
72              nhw_light = 3'b001;
73              nlr_light = 3'b100;
74          end
75          S3:begin
76              if(g_count >= 34)begin
77                  nstate = S4;
78                  ng_count = 0;
79                  nhw_light = 3'b001;
80                  nlr_light = 3'b010;
81              end
82              else begin
83                  nstate = S3;
84                  ng_count = g_count + 1;
85                  nhw_light = 3'b001;
86                  nlr_light = 3'b100;
87              end
88          end
```

如果是 HW 或 LR 是綠燈的部分(S0,S3)，每次 g_count 就會加 1，直到 35 次，但是 HW 要變成下一個 state 的話，LR_has_Car 要等於 1。

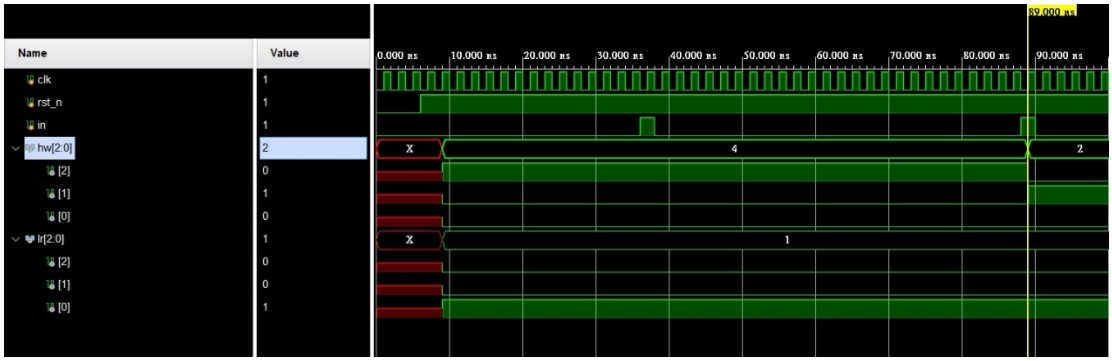
如果是 HW 或 LR 是黃燈的部分(S1,S4)，每次 g_count 就會加 1，直到 15 次，接著經過 sequential 變成下個 state。

```
89          S4:begin
90              if(y_count >= 14)begin
91                  nstate = S5;
92                  ny_count = 0;
93                  nhw_light = 3'b001;
94                  nlr_light = 3'b001;
95              end
96              else begin
97                  nstate = S4;
98                  ny_count = y_count + 1;
99                  nhw_light = 3'b001;
100                 nlr_light = 3'b010;
101             end
102         end
103         S5:begin
104             nstate = S0;
105             nhw_light = 3'b100;
106             nlr_light = 3'b001;
107         end
108     endcase
```

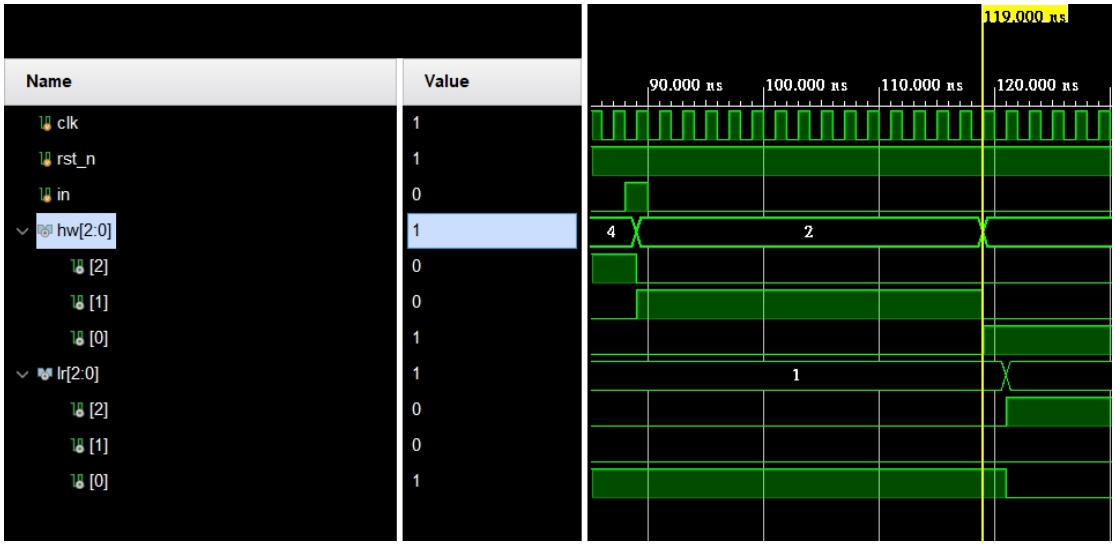
Testbench：



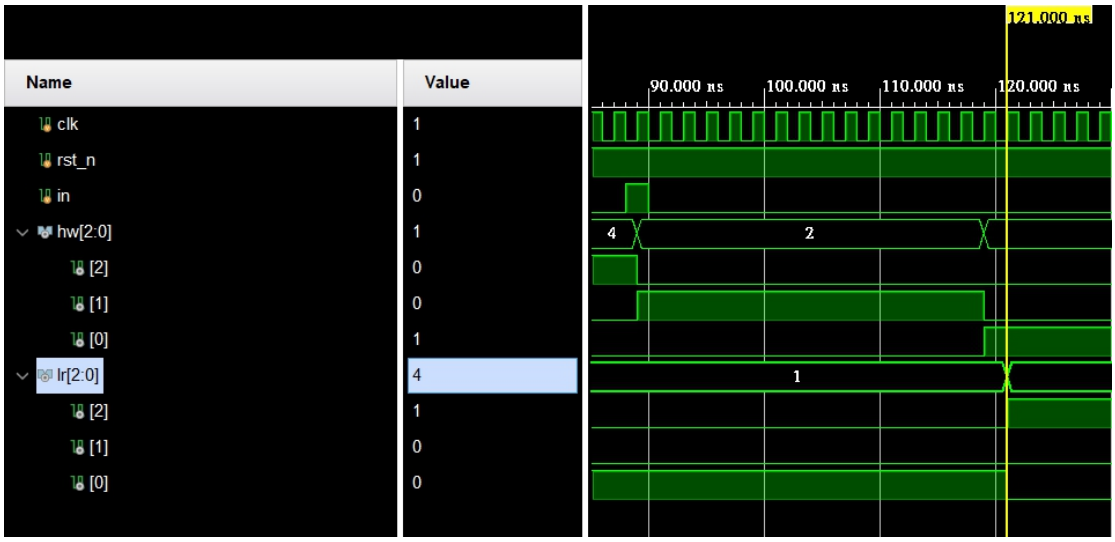
在 9.000ns 的時候 HW 為綠燈，LR 為紅燈



可以看到在 36.000ns 的時候 LR_has_Car = 1，但是因為 HW 還沒等到 35 cycles, 所以燈號沒改變，直到 89.000ns 的時候，此時 HW_light 已經經過了 40 個 cycles((89 – 9)/2) >= 35，所以會開始改變燈號，HW 進入黃燈的狀態。



到了 119.000ns 的時候 HW 黃燈的狀態結束了，進入紅燈的狀態



當 HW 變成了紅燈時，LR 會再等一個 clk 才轉為綠燈狀態((121-119)/2) = 1 cycle

後面的狀態就跟 state diagram 一樣進行變化。

Lab 5 Report

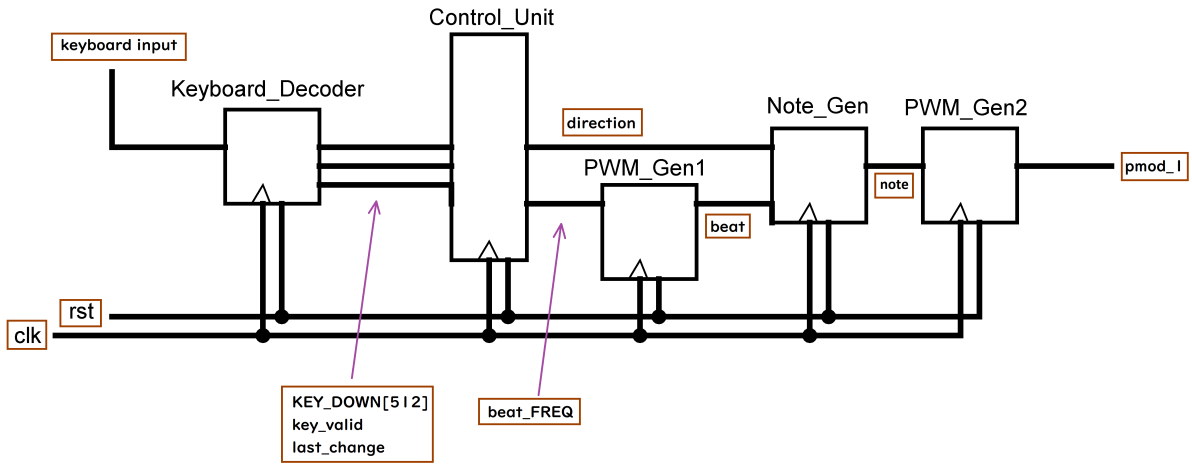
Course	Logic Design Lab
Done	☑
Last edited time	@Nov 26, 2020 11:35 PM
URL	
file	

FPGA 1

藉由鍵盤控制目前發出聲音的上階、下階、速度。

Circuit

為了方便辨識，wire的名稱全部用橘框框起。



Description

- *Keyboard_Decoder*：對鍵盤輸入進行解碼，轉換成`KEY_DOWN[512]`，`key_valid`，`last_change`轉發給*Control_Unit*
- *Control_Unit*：讀取當前的`last_change`，視情況轉換內建的`reg direction`（數字0與1）或改變內建的`reg beat_FREQ`（數字2）

    - `direction`：決定下一個note是向上還是向下。
    - `beat_FREQ`：內容為`32'd1`或`32'd2`，可以改變*PWM_Gen1*輸出`beat`的頻率。
    - `rst`：將`direction` reset成`1'b1`，`beat_FREQ` reset成`32'd2`。
- *PWM_GEN1*：依據`beat_FREQ`給定的值，來決定除頻的速度（1Hz or 2Hz），輸出給`beat`。
- *Note_Gen*：內建`reg cur_note`，紀錄上一個輸出的`note`，配上`direction`後可以輸出下一個`note`的頻率。
    - `trigger` and `prev`： 這裡對`trigger`（即為`beat`）作處理，利用`prev`來判斷是否當前是`trigger`的正緣，可以做出等同於`always @(posedge trigger)`的效果。

> 這裡不直接使用`trigger`的原因是可能會造成delay（嗎？我記得上課有說到盡量只寫`clk`的DFF就好，而實際上我寫`trigger`的DFF還真的會出問題）。

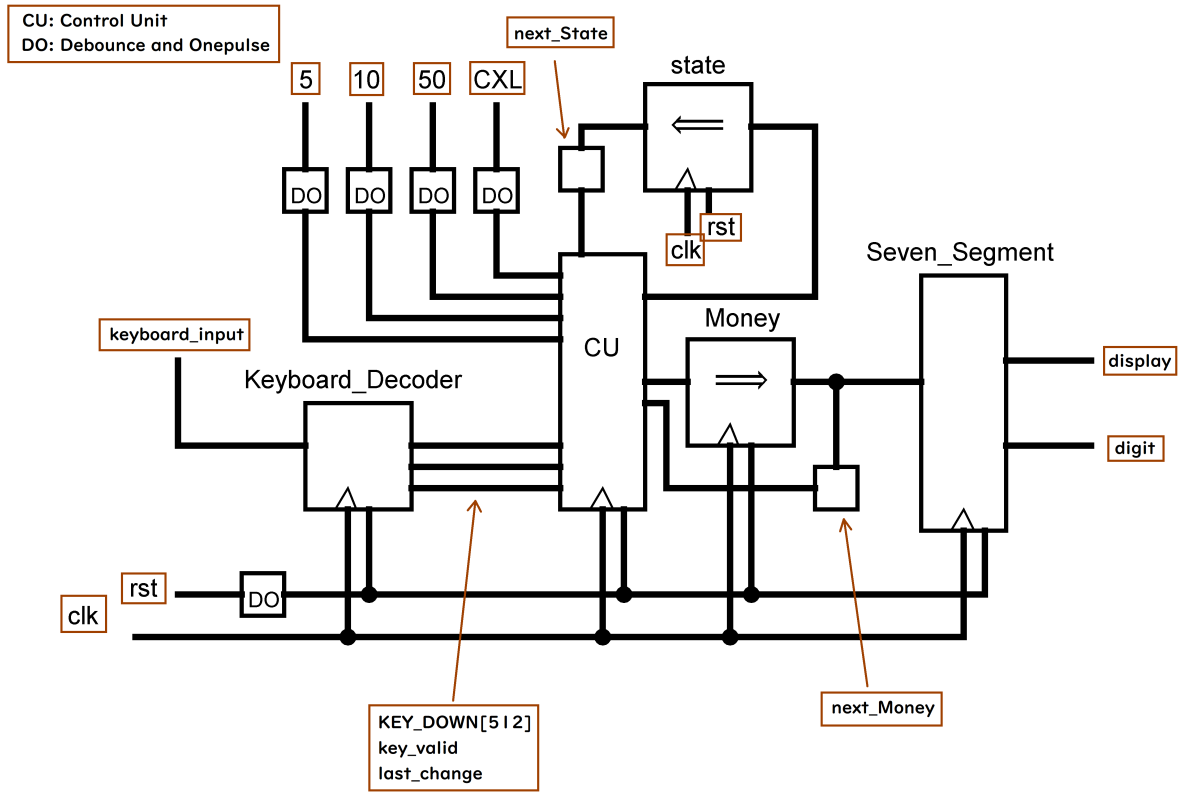
```
reg prev;
reg [4:0] note, n_note;

always @(posedge clk, posedge reset) begin
    prev <= trigger;

    if (reset) begin
        note <= 5'd0;
    end
    else begin
        if (trigger && !prev) begin
            note <= n_note;
        end
        else begin
            note <= note;
        end
    end
end
```

- *PWM_GEN2*：依照*Note_Gen*給定的頻率進行除頻，將正確頻率輸出到`pmod_1`，即可發出聲音。

FPGA 2

實作販賣機

Circuit

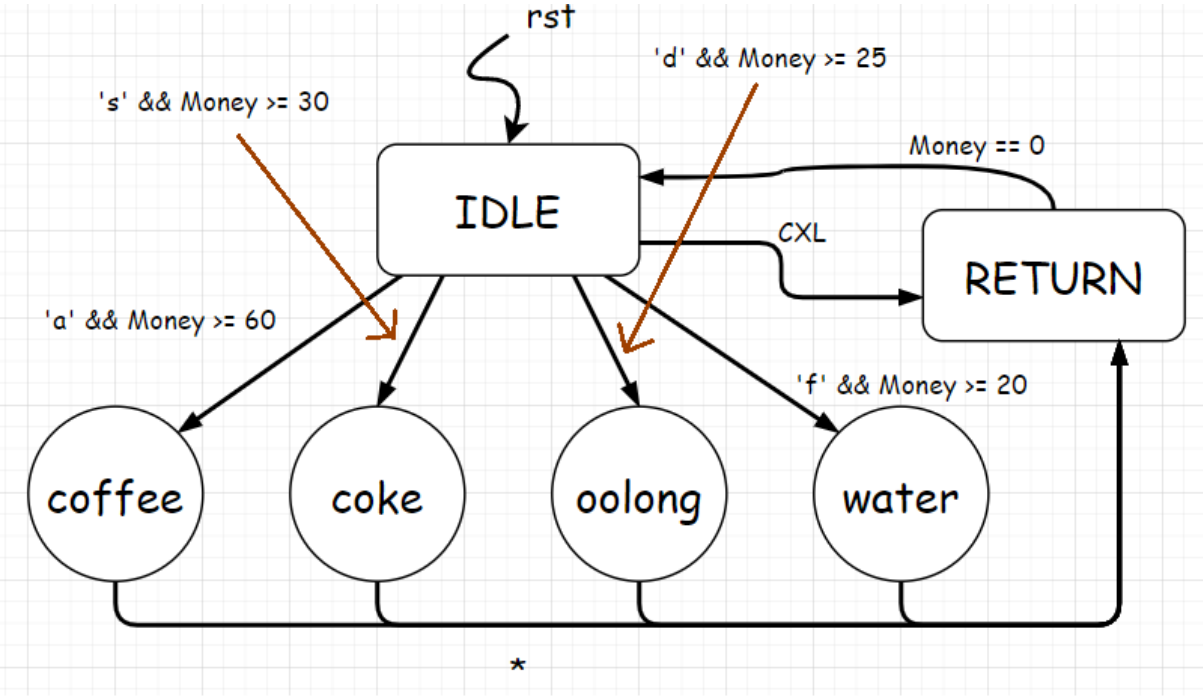


Description

- *Keyboard_Decoder*：同**FPGA1**。
- *Control_Unit*：根據目前的State做出相對應的動作。
  - **IDLE**：投錢階段，是否投入錢幣。
    - `5`、`10`、`50`：Money增加（上限99）
    - Keyboard `a` `s` `d` `f`：判斷Money足不足夠，如果可以的話，進入各自的飲料state，否則留在**IDLE**。
    - `CXL`：進入**RETURN** state。
  - **coffee, coke, oolong, water**：扣除飲料需要的錢，進入**RETURN** state。
  - **RETURN**：利用`count`計算1秒，每過1秒扣除5塊錢，直到歸零並進入**IDLE** state。

- *Seven_Segment*：提供的Sample，將輸入的數字轉換為BCD number（`display`），並不斷切換顯示的數字（`digit`）。

State Diagram



分工：
Advanced 1：林諭震
Advanced 2：林諭震
FPGA 1：莊景堯
FPGA 2：莊景堯
Report：林諭震、莊景堯

心得：
經過前面幾次 state 的題目練習過後，這一次的 advanced1、2 題寫得比較上手，這次一次 FPGA 變成了兩題而且還加入 Keyboard 和 Audio，在寫得時候花了比較多得時間，尤其在 debug 的時候沒辦法 simulation 因為有 inout 的變數類型，所以很難找到哪邊錯誤。