Lab 5

學號: 108071003 **姓名: 李彥璋**

1. 實作過程

- **onepulse & debounce：**

  已於 lab4 report 詳細介紹，此不再贅述。

- **clock divider：**

  因為題目規定「The operating frequency of each debouncing or one-pulse converter should be the same as your FSM's」，所以我決定使用 100Hz 的 clock (1 / 100 = 0.01 sec)。

```
module clock_divider(
  input clk,
  output wire clk_div
  );

  reg [19:0] cnt;
  parameter n = 20;

  always @(posedge clk) begin
    if(cnt < 1000000-1'b1) begin
      cnt <= cnt + 1'b1;
    end else begin
      cnt <= 20'd0;
    end
  end
  assign clk_div = cnt[n-1];
endmodule
```

code explaination：

板子的 clock 是 100MHz，想要得到 100Hz 的 clk_div，要除以 1000000。

1000000 換成 binary 有 20 bits，因此設定一個 cnt 計數器，從 0 開始，每次加 1，加到 999999 後歸 0。

而 output 的 clk_div，只要取 cnt 計數器最前面的 bit，就能分成 0 / 1 兩半了。

- **7-segment：**

```
reg [19:0] frequency_cnt = 20'd0;
wire [1:0] frequency;

always@(posedge clk) begin
  frequency_cnt <= frequency_cnt + 20'd1;
end

assign frequency = frequency_cnt[19:18];
```

code explaination：

我以一個 20 bits 的 frequency_cnt 計數器，循環加 1。四個 7-segment 以該計數器前兩 bits (00、01、10、11) 輪流顯示 (實際上會達到視覺暫留，看起來是同時)。

- **切換 state：**

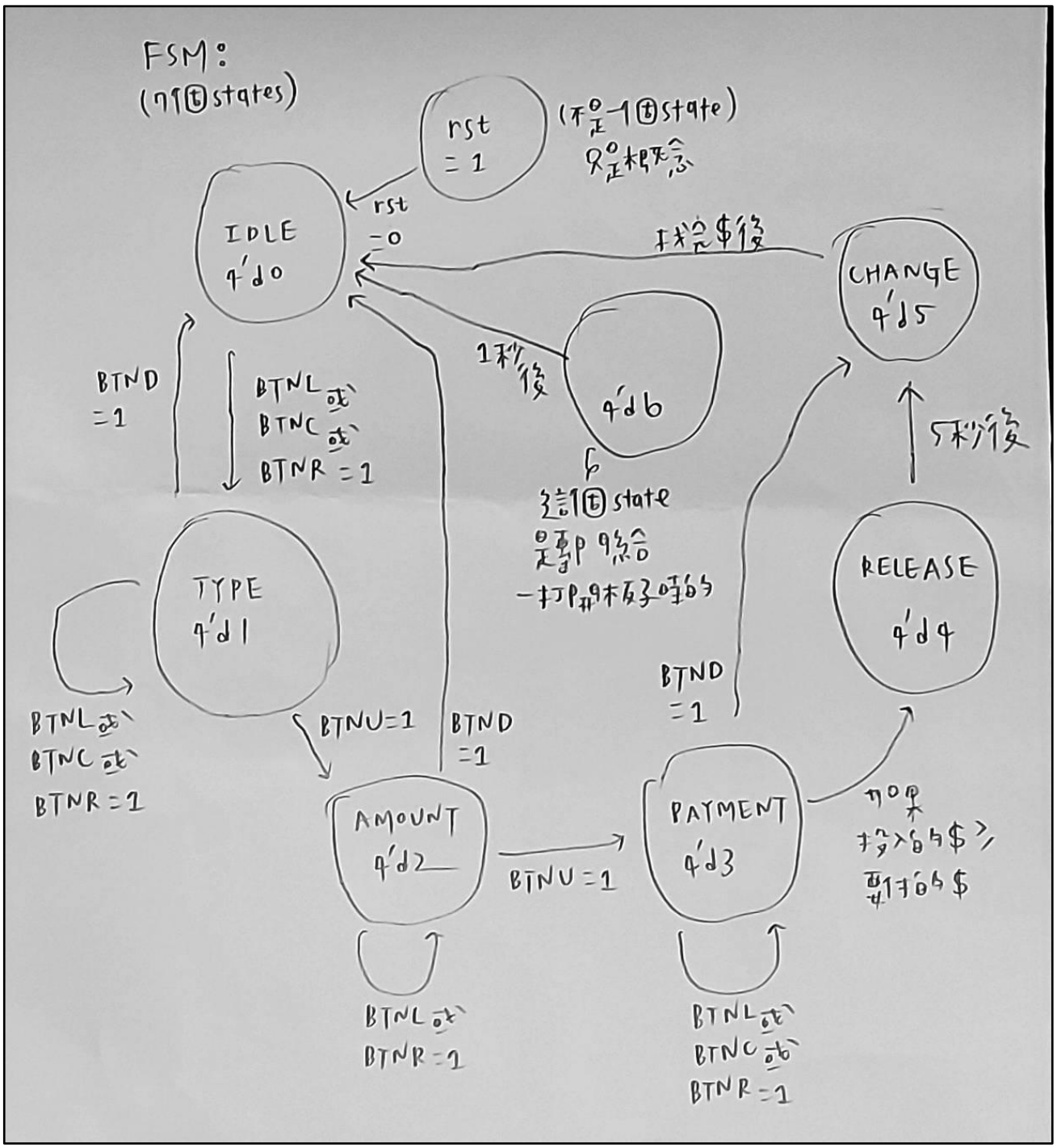
```
always @(posedge clk) begin
   states <= states_next;
end
```

code explaination：

這個 always block 是拿來切換 FSM – state 的。我原本是用 100Hz 的 clock（跟其他地方一樣），但是一直有問題，我感覺是 state 的轉換太慢了（其實也不確定），但後來改成原本的 100MHz 就沒問題了。

- **FSM：**

先附圖。



```
if(rst == 1'b1) begin
   counter <= 10'd0;
   flashing <= 1'b1;
   states_next <= 4'd0;

   LED <= 16'b0000000000000000;
   BCD0 <= 4'd15;
   BCD1 <= 4'd15;
   BCD2 <= 4'd15;
   BCD3 <= 4'd15;
end else begin
```

reset：

無論何時，只要切換至 reset = 1，板子整個都是暗的，button 都無效。

因為當切換至 reset = 0，要回到 IDLE state，所以一些 IDLE state 要用到的變數（flashing、counter）都重置。

++

```
4'd0: begin
   if(pulse_BTNL == 1'b1) begin
      states_next <= 4'd1;
      type <= 4'd13;
   end else if(pulse_BTNC == 1'b1) begin
      states_next <= 4'd1;
      type <= 4'd14;
   end else if(pulse_BTNR == 1'b1) begin
      states_next <= 4'd1;
      type <= 4'd12;
   end else begin
      states_next <= 4'd0;
   end

   if(counter == 100) begin
      counter <= 0;
      flashing <= ~flashing;
   end
   else counter <= counter + 1;

   if(flashing == 1'b1) begin
      BCD0 = 4'd11;
      BCD1 = 4'd11;
      BCD2 = 4'd11;
      BCD3 = 4'd11;
      LED <= 16'b1111111111111111;
   end else begin
      BCD0 = 4'd15;
      BCD1 = 4'd15;
      BCD2 = 4'd15;
      BCD3 = 4'd15;
      LED <= 16'b0000000000000000;
   end
end
```

IDLE state（4’d0）：

按下 BTNL / BTNC / BTNR，切換至 TYPE state，並賦值 type 變數相對應的票型

4’d13 = child；

4’d14 = student；

4’d12 = adult。

如果沒有按按鈕，停留在 IDLE state。

counter 變數用來控制 flashing 變數。

flashing 變數 = 1，LED 亮、7-seg = dash；

flashing 變數 = 0，LED 暗、7-seg = 暗。

題目說「The flashing frequency of LEDs and 7-segment display is 0.5Hz」，0.5Hz 是 2 秒，所以亮一秒、暗一秒。

clock 是 0.01 秒，所以數到 100 才是一秒。

++

```
4'd1: begin
   if(pulse_BTNU == 1'b1) begin
      states_next <= 4'd2;
      amount <= 4'd1;
   end else if(pulse_BTND == 1'b1) begin
      states_next <= 4'd0;
      counter <= 10'd0;
      flashing <= 1'b1;
   end else begin
      states_next <= 4'd1;
      if(pulse_BTNL == 1'b1) begin
         type <= 4'd13;
      end else if(pulse_BTNC == 1'b1) begin
         type <= 4'd14;
      end else if(pulse_BTNR == 1'b1) begin
         type <= 4'd12;
      end
   end

   LED <= 16'b0000000000000000;
   if(type == 4'd13) begin
      BCD0 = 4'd5;
      BCD1 = 4'd0;
   end else if(type == 4'd14) begin
      BCD0 = 4'd0;
      BCD1 = 4'd1;
   end else if(type == 4'd12) begin
      BCD0 = 4'd5;
      BCD1 = 4'd1;
   end
   BCD2 = 4'd15;
   BCD3 = type;
end
```

TYPE state (4'd1):

按下 BTNU，切換至 AMOUNT state，並將 AMOUNT state 會用到的 amount 變數設置為 1。

按下 BTND，切換至 IDLE state，並將 IDLE state 會用到的變數(flashing、counter)重置。

按下 BTNL / BTNC / BTNR，停留在 TYPE state，並賦值 type 變數相對應的票型

4'd13 = child；

4'd14 = student；

4'd12 = adult。

在 TYPE state，

LED 一直是全暗；

BCD0、BCD1 根據選擇的票型 (type)，顯示價錢 (兒童 05 元、學生 10 元、成人 15 元)；

BCD2 一直是全暗；

BCD3 顯示票型 (兒童 C、學生 S、成人 A)。

++

```
4'd2: begin
   if(pulse_BTNU == 1'b1) begin
      states_next <= 4'd3;
      deposite <= 6'd0;
      if(type == 4'd12) price <= 15 * amount;
      else if(type == 4'd13) price <= 5 * amount;
      else if(type == 4'd14) price <= 10 * amount;
   end else if(pulse_BTND == 1'b1) begin
      states_next <= 4'd0;
      counter <= 10'd0;
      flashing <= 1'b1;
   end else begin
      states_next <= 4'd2;
      if(pulse_BTNL == 1'b1) begin
         if(amount > 4'd1) amount <= amount - 1;
      end else if(pulse_BTNR == 1'b1) begin
         if(amount < 4'd3) amount <= amount + 1;
      end
   end

   BCD0 <= amount;
   BCD1 <= 4'd15;
   BCD2 <= 4'd15;
   BCD3 <= type;
end
```

AMOUNT state (4'd2):

按下 BTNU，切換至 PAYMENT state，並將 PAYMENT state 會用到的變數 (deposite 表投入多少$、price 表買的票多少$) 設置好。

按下 BTND，切換至 IDLE state，並將 IDLE state 會用到的變數 (flashing、counter) 重置。

其餘狀況停留在 AMOUNT state，按下 BTNL 減少購買張數 (最少 1)；按下 BTNR 增加購買張數 (最多 3)。

BCD0 顯數目前購買張數 (amount)；

BCD1、BCD2 一直暗；BDC3 顯示購買票型 (type)。

++

```
4'd3: begin
   if(deposite >= price) begin
      states_next <= 4'd4;
      changes <= deposite - price;
      counter <= 10'd0;
      flashing <= 1'b1;
      count_5sec <= 13'd0;
   end else if(pulse_BTND == 1'b1) begin
      states_next <= 4'd5;
      changes <= deposite;
      counter <= 0;
   end else begin
      states_next <= 4'd3;
      if(pulse_BTNL == 1'b1) begin
         deposite <= deposite + 1;
      end else if(pulse_BTNC == 1'b1) begin
         deposite <= deposite + 5;
      end else if(pulse_BTNR == 1'b1) begin
         deposite <= deposite + 10;
      end
   end

   BCD0 <= price % 10;
   BCD1 <= price / 10;
   BCD2 <= deposite % 10;
   BCD3 <= deposite / 10;
end
```

PAYMENT state (4'd3):

如果 (投入的$ > 購買票的$), 切換至 RELEASE state, 並將之後會用到的變數 (changes 表找錢、counter & flashing 控制 LED、count_5sec 控制 5 秒) 設定好。

按下 BTND, 切換至 CHANGE state, 並將 CHANGE state 會用到的變數 (changes 表找錢、counter 控制 1 秒) 設定好。

其餘狀況停留在 PAYMENT state,
按下 BTNL, 投入$1; 按下 BTNC, 投入$5; 按下 BTNR, 投入$10。

BCD0、BCD1 顯示要付的錢。
BCD2、BCD3 顯示目前投入的錢。

++

```
4'd4: begin
   if(counter == 100) begin
      counter <= 0;
      flashing <= ~flashing;
   end
   else counter <= counter + 1;

   if(flashing == 1'b1) LED <= 16'b1111111111111111;
   else LED <= 16'b0000000000000000;


   if(count_5sec >= 500) begin
      states_next <= 4'd5;
      counter <= 0;
   end else begin
      states_next <= 4'd4;
   end

   count_5sec <= count_5sec + 1;

   BCD0 <= amount;
   BCD1 <= 4'd15;
   BCD2 <= 4'd15;
   BCD3 <= type;
end
```

RELEASE state (4'd4):

counter 變數用來控制 flashing 變數, 數到 100 是 1 秒。

flashing 變數 = 1, LED 亮;
flashing 變數 = 0, LED 暗。

Count_5sec 用來數 5 秒, 數到 500 是 5 秒, 切換至 CHANGE state, 並將 CHANGE state 會用到的 counter 變數 (數 1 秒) 重置。
沒數到 5 秒前停留在 RELEASE state。

BCD0 顯示購買的張數;
BCD1、BCD2 一直都暗;
BCD3 顯示購買票的類型。

++

```
4'd5: begin
   LED <= 16'b0000000000000000;
   BCD0 <= changes % 10;
   BCD1 <= changes / 10;
   BCD2 <= 4'd15;
   BCD3 <= 4'd15;

   if(changes == 0) begin
      if(counter == 100) begin
         states_next <= 4'd0;
         counter <= 10'd0;
         flashing <= 1'b1;
      end else states_next <= 4'd5;
   end else begin
      if(counter == 100) begin
         if(changes >= 5) changes <= changes - 5;
         else if(changes >= 1) changes <= changes - 1;
      end
      states_next <= 4'd5;
   end

   if(counter == 100) counter <= 0;
   else counter <= counter + 1;
end
```

CHANGE state (4'd5):

在 CHANGE state，

LED 一直暗；

BCD0、BCD1 顯示還剩多少錢要找；

BCD2、BCD3 一直暗。

如果找完錢了，切換至 IDLE state；

如果還沒，停留在 CHANGE state，若要找的錢還>=5，一次找 5 元；若要找的錢<5 && >=1，一次找 1 元。

每個數字（包含 0）都是停留一秒。

++

```
4'd6: begin
   if(counter == 100) begin
      states_next <= 4'd0;
      counter <= 0;
      flashing <= 1'b0;
   end else begin
      states_next <= 4'd6;
      counter <= counter + 1;
   end
   LED <= 16'b1111111111111111;
   BCD0 <= 4'd11;
   BCD1 <= 4'd11;
   BCD2 <= 4'd11;
   BCD3 <= 4'd11;
end

default: begin
   states_next <= 4'd6;
   counter <= 0;
   LED <= 16'b1111111111111111;
   BCD0 <= 4'd11;
   BCD1 <= 4'd11;
   BCD2 <= 4'd11;
   BCD3 <= 4'd11;
end
```

(4'd6 & default):

這兩個 state 主要是給予初值，default 的 next_state 是 4'd6；而 4'd6 經過 1 秒後會跳到 IDLE state。

在這兩個 state，

LED 全亮；

BCD0~3 都是 dash。

++

2. 學到的東西與遇到的困難

要說遇到甚麼具體的困難其實有點難，我前後整個 code 刪掉重寫了 3 次，可能是因為這次的 FSM 太多了、腦袋又不太清楚所致（其實 Lab5 沒有什麼新的東西，只是小小的複雜了一點點）。

不過我這次最後是寫完一個 state 就燒一次板子，並馬上 debug。因為 Lab4 之前我都是全部寫完再一起燒一起 debug（錯誤常常要找很久），我覺得這次的方法比較好，算是學到一課。

3. 想對老師或助教說的話

謝謝老師&助教。

期中考好恐怖.___.。

之後要實體上課，好像要調整一下心境了！