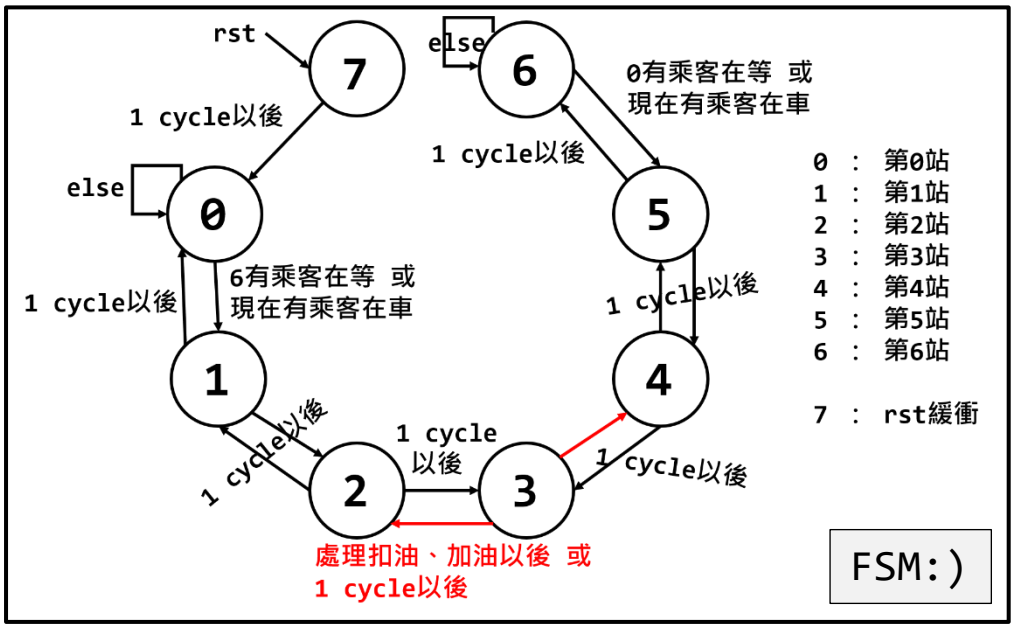
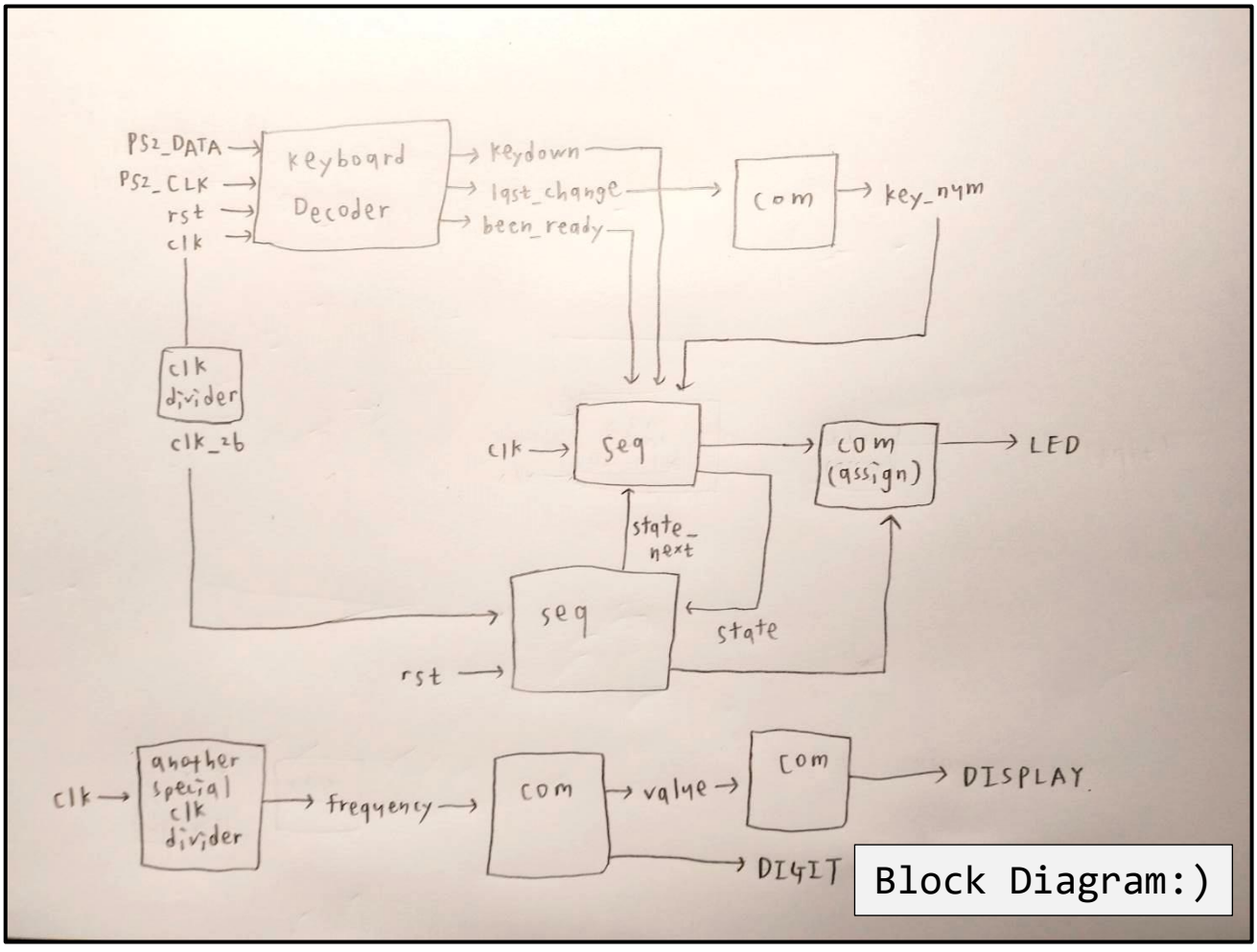
Lab 6

學號：108071003 姓名：李彥璋

1. 實作過程

Lab6 要實作一個 shuttle bus，以下附上 FSM & block diagram：





接著講述程式碼各個部份：

++

- **LED**

```
assign B1_LED = ( B1_passenger_v2 == 2'b11 ) ? 2'b11 : ( (B1_passenger_v2 == 2'b10) ? 2'b10 : 2'b00 );
assign B2_LED = ( B2_passenger_v2 == 2'b11 ) ? 2'b11 : ( (B2_passenger_v2 == 2'b10) ? 2'b10 : 2'b00 );
assign bus_LED = ( passenger_on_bus == 2'b11 ) ? 2'b11 : ( (passenger_on_bus == 2'b10) ? 2'b10 : 2'b00 );
assign LED = { B1_LED , 1'b0 , B2_LED , bus_LED , 2'b00 , bus_location };
```

因為 LED 是即時顯示的，所以我想說用 assign 會比較好，

透過其他變數紀錄各 LED 該是亮 or 暗，隨時 assign。

++

- **鍵盤**

```
KeyboardDecoder key_de (
        .key_down(key_down),
        .last_change(last_change),
        .key_valid(been_ready),
        .PS2_DATA(PS2_DATA),
        .PS2_CLK(PS2_CLK),
        .rst(rst),
        .clk(clk)
);
```

這個部分主要是輸入 PS2_DATA、PS2_CLK、rst、clk，

得到輸出 key_down、last_change、been_ready。

其中 key_down 用來判斷 user 有沒有按鍵盤；

lastchange 用來判斷是按下哪個按鍵；

been_ready 用來判斷鍵盤是不是 valid(準備好的感覺)。

```
parameter [8:0] KEY_CODES [0:19] = {
                9'b0_0100_0101,     // 0 => 45
                9'b0_0001_0110,     // 1 => 16
```

這個部分用來定義各個按鍵的 key_code。

```
always @* begin
        case (last_change)
                KEY_CODES[00] : key_num = 4'b0000;
                KEY_CODES[01] : key_num = 4'b0001;
```

這個部分以 last_chage，取的表示按下哪個按鍵的 key_num。

++

- **Chage state always block**

```
1  always @(posedge clk) begin
2          states <= states_next;
3
4          if(states_next == 4'd7) begin
5              B1_passenger_v2 <= 2'b00;
6              B2_passenger_v2 <= 2'b00;
7          end
8          if(states == 4'd0) begin
9              if(B1_passenger == 0) B1_passenger_v2 <= 2'b00;
10             else if(B1_passenger == 1) B1_passenger_v2 <= B1_passenger_v2;
11             B2_passenger_v2 <= B2_passenger_v2;
12         end else if(states == 4'd6) begin
13             if(B2_passenger == 0) B2_passenger_v2 <= 2'b00;
14             else if(B2_passenger == 1) B2_passenger_v2 <= B2_passenger_v2;
15             B1_passenger_v2 <= B1_passenger_v2;
16         end
17
18         if(been_ready && key_down[last_change] == 1'b1) begin
19                 if(key_num != 4'b1111) begin
20                     if(key_num == 4'b0001) begin
21                         if(B1_passenger_v2 == 2'b11) B1_passenger_v2 <= 2'b11;
22                         else if(B1_passenger_v2 == 2'b10) B1_passenger_v2 <= 2'b11;
23                         else if(B1_passenger_v2 == 2'b00) B1_passenger_v2 <= 2'b10;
24                     end else if(key_num == 4'b0010) begin
25                         if(B2_passenger_v2 == 2'b11) B2_passenger_v2 <= 2'b11;
26                         else if(B2_passenger_v2 == 2'b10) B2_passenger_v2 <= 2'b11;
27                         else if(B2_passenger_v2 == 2'b00) B2_passenger_v2 <= 2'b10;
28                     end
29                 end
30         end
31     end
```

[2] 將 state 更改為下一個 state。

[4-7] state_next 如果等於 4’d7，代表剛剛有按下 rst，這時把在 B1、B2 等候的乘客重置 (其他變數的重置在另一個 ALWAYS BLOCK。這兩個變數之所以寫在這裡，是因為同一個變數不能出現在不同 ALWAYS BLOCK 的等號左邊)。

[8-11] 判斷在第 0 站的時候，若乘客上車了，B1 等候乘客重置；若沒有乘客上車，B1 等候乘客維持原值，而 B2 等候乘客一直都是維持原值。

[12-16] 同理[8-11]。

[18-30] 判斷當 user 按下按鍵 1 or 按鍵 2，B1/B2 等候乘客應該更新成多少。

++

- **FSM always block**

這個部分講述各個 STATE 在做什麼，**因為這次的 STATE 有對稱的特性，所以我只介紹其中一半。**

首先是 **rst** 處理的部分，主要是各個變數的重置，以下說明各個變數的功能。

（第 7 個 state，只有 state_next 改成 4'd0，其餘和 rst 相同）。

```
2  always @(posedge rst or posedge global_clk) begin
3      if(rst == 1'b1)
4          begin
5              states_next <= 4'd7;
6              B1_passenger <= 1;
7              B2_passenger <= 1;
8              passenger_on_bus <= 2'd0;
9              getoff_bus <= 1'b0;
10             geton_bus <= 1'b0;
11             up_hill <= 1'b1;
12             pay_ornot <= 1'b0;
13             bus_location <= 7'b0000001;
14             gas <= 5'd0;
15             revenue <= 7'd0;
16             minus_gas <= 1'b1;
17         end
```

[5] states_next 用來設定下一個 state。

[6] B1_passenger 用來判斷在第 0 站，是否有乘客上車 (0 是有；1 是沒)。

[7] B2_passenger 用來判斷在第 6 站，是否有乘客上車 (0 是有；1 是沒)。

[8] passenger_on_bus 用來判斷有多少人在車上。

[9] getoff_bus 用來判斷在第 0 站、第 6 站乘客是否下過車了。

[10] geton_bus 用來判斷在第 0 站、第 6 站乘客是否上過車了。

[11] up_hill 用來判斷現在是上山還是下山。

[12] pay_ornot 用來判斷上車的乘客付過錢了沒。

[13] bus_location 用來記錄 bus 現在的位置。

[14] gas 用來記錄剩餘多少油。

[15] revenue 用來記錄有多少收入。

[16] minus_gas 用來判斷在第 0、3、6 站時，是否扣過車油了。

第 0 站（第 6 站同理）

```
1            if(states == 4'd0)
2                begin
3                    up_hill <= 1;
4                    states_next <= 4'd0;
5                    bus_location <= 7'b0000001;
6
7                    if(minus_gas == 1'b0 && passenger_on_bus != 0) begin
8                        if(passenger_on_bus == 2'b11) gas <= gas - 10;
9                        else if(passenger_on_bus == 2'b10) gas <= gas - 5;
10                       minus_gas <= 1'b1;
11                   end else begin
12                       minus_gas <= 1'b1;
13                       if( passenger_on_bus != 0 && getoff_bus == 0 ) begin // 車上有人 ， 還沒下過車
14                           if(passenger_on_bus == 2'b11) passenger_on_bus <= 2'b10;
15                           else if(passenger_on_bus == 2'b10) passenger_on_bus <= 2'b00;
16
17                           if( passenger_on_bus == 0 ) getoff_bus <= 1; // 下過車了
18                       end else begin
19                           getoff_bus <= 1; // 本來就沒乘客 ， 也是下過車了
20                           if( B1_passenger_v2 != 0 && geton_bus == 0 ) begin // 有人在等車 ， 還沒上過車
21                               passenger_on_bus <= B1_passenger_v2;
22                               B1_passenger <= 0;
23                               geton_bus <= 1; // 上過車了
24                           end else begin
25                               B1_passenger <= 1;
26                               if( geton_bus == 1'b1 && pay_ornot == 1'b0) begin // 真的有乘客上車 ， 還沒付錢
27                                   if(passenger_on_bus == 2'b11) begin // 付錢
28                                       if(revenue + 60 >= 90) revenue <= 90;
29                                       else revenue <= revenue + 60;
30                                   end else if(passenger_on_bus == 2'b10) begin
31                                       if(revenue + 30 >= 90) revenue <= 90;
32                                       else revenue <= revenue + 30;
33                                   end
34                                   pay_ornot <= 1'b1; // 付過錢了
35
36                                   if( gas < 20 ) states_next <= 4'd0;
37                                   else states_next <= 4'd1;
38                               end else begin
39                                   if( pay_ornot == 1'b1 && gas < 20 ) begin // 付過錢了 ， 要加油
40                                       revenue <= revenue - 10;
41                                       if( gas + 10 >= 20 ) begin
42                                           gas <= 20;
43                                           states_next <= 4'd1;
44                                       end else begin
45                                           gas <= gas + 10;
46                                           if( revenue >= 10 ) states_next <= 4'd0;
47                                           else states_next <= 4'd1;
48                                       end
49                                   end else begin
50                                       if( B2_passenger_v2 != 2'd0 ) states_next <= 4'd1;
51                                   end
52                               end
53                           end
54                       end
55                   end
56               end
```

[3] 第 0 上之後一定是上山。

[4] 預設下一站還是第 0 站（因為可能沒人上車而第 6 站也沒有人在等，那 bus 就待在原地）。

[5] 設定 bus 的位置。

[7-10] 還沒扣過車油&&有人在車上，根據車上乘客數量扣油，並設定扣過油了。

[12] 如果一開始車上就沒有人，也要設定扣過油了。

[13-17] 車上有人&&還沒下過車，一次下一個人。如果下完了，設定下過車了。

[19] 本來車上就沒人，也要設定下過車了。

[20-23] 有人在等車&&還沒上過車，讓他們上車，且要提醒等候的人要歸零，並設定上過車了。

[25] 沒人要上車，也要設定上過車了。

[26-37] 剛剛有人上車&&還沒付錢，按照乘客數量增加 revenue，設定付過錢了。判斷有沒有要加油，若有，維持在第 0 站；若沒有，去往第 1 站。

[39-48] 付過錢了&&要加油，分次把油加滿，加完油就去往第 1 站。

[50] 如果第 0 站沒人上車，但第 6 站有人在等，去往第 1 站。

第 1 站（第 5 站同理）

```
2           else if(states == 4'd1)
3               begin
4                   bus_location <= 7'b0000010;
5                   if(up_hill == 1) states_next <= 4'd2;
6                   else if(up_hill == 0) begin
7                       states_next <= 4'd0;
8                       getoff_bus <= 0;
9                       geton_bus <= 0;
10                      pay_ornot <= 0;
11                      minus_gas <= 1'b0;
12                  end
13              end
```

[4] 設定 bus 的位置。

[5] 上果是上山，去往第 2 站。

[6-11] 如果是下山，去往第 0 站，並重置該重置的變數。

第 2 站（第 4 站同理）

```
2                   else if(states == 4'd2) begin
3                       bus_location <= 7'b0000100;
4                       if(up_hill == 1) begin
5                           states_next <= 4'd3;
6                           minus_gas <= 1'b0;
7                       end
8                       else if(up_hill == 0) states_next <= 4'd1;
9                   end
```

[3] 設定 bus 的位置。

[4] 如果是上山，去往第 3 站，並重置該重置的變數。

[8] 如果是下山，去往第 1 站。

第 3 站

```
2               else if(states == 4'd3) begin
3                   bus_location <= 7'b0001000;
4                   if(minus_gas == 1'b0 && passenger_on_bus != 0) begin
5                       if(passenger_on_bus == 2'b11) gas <= gas - 10;
6                       else if(passenger_on_bus == 2'b10) gas <= gas - 5;
7                       minus_gas <= 1'b1;
8                   end else begin
9                       minus_gas <= 1'b1;
10                      if( passenger_on_bus > 0 && revenue >= 10 && gas < 20 ) begin
11                          revenue <= revenue - 10;
12                          if( gas + 10 >= 20 ) begin
13                              gas <= 20;
14                              if(up_hill == 1) states_next <= 4'd4;
15                              else if(up_hill == 0) states_next <= 4'd2;
16                          end else begin
17                              gas <= gas + 10;
18                              if( revenue >= 10 ) states_next <= 4'd3;
19                              else begin
20                                  if(up_hill == 1) states_next <= 4'd4;
21                                  else if(up_hill == 0) states_next <= 4'd2;
22                              end
23                          end
24                      end else begin
25                          if(up_hill == 1) states_next <= 4'd4;
26                          else if(up_hill == 0) states_next <= 4'd2;
27                      end
28                  end
29              end
```

[3] 設定 bus 的位置。

[4-7] 還沒扣過油&&有人在車上，根據車上人數量扣油，設定扣過油了。

[9] 本來就沒人在車上，也要設定扣過油了。

[10-21] 幫車加油，並根據上山還是下山決定去往哪裡。

[24-26] 本來就沒有要加油，根據上山還是下山決定去往哪裡。

2. 遇到的困難 學到的東西

✧ 一開始寫完鍵盤完全沒反應 (按了之後 LED 不亮)，真的弄了很久，後來修好之後也不知道是什麼問題= =，因為是 code 整個重寫才修好的 TT。

✧ 乘客數量一直搞錯 (因為是要亮 LED，所以 2 個人是 11；一個人是 10)，然後我一直把它直接當成數字來算，想說到底為什麼怪怪的 XD。

✧ 遇到最大的困難是要怎麼把讓 B1、B2 等候的人歸 0，在按下 rst or 乘客上車之後，因為我在兩個 always block 都想使用同一個變數 (會壞掉)，後來是再多用一個變數來提醒它歸 0 (想了很久)。

✧ 學到的東西，主要就是鍵盤的使用了吧 030，雖然這次的 lab 只用到兩個按鍵，但其他道理應該相同，可能之後可以想想連按之類的。

3. 想對老師 助教說的話

謝謝老師&助教。

終於第一次畫了整個 code 的 block diagram，雖然也不確定是對不對 (？