Lab 6

學號: 109000168 姓名: 許媄香

1. 實作過程

In this lab, I use 2^{10} clock divider for seven segment and 2^{26} for the rest (except for keyboard decoder).

- 7-segments Display

```
always @ (posedge clk_divider[9]) begin
    case (digit)
        4'b1110 : begin
                display_num <= nums[7:4];
                digit <= 4'b1101;
            end
        4'b1101 : begin
                display_num <= nums[11:8];
                digit <= 4'b1011;
            end
        4'b1011 : begin
                display_num <= nums[15:12];
                digit <= 4'b0111;
            end
        4'b0111 : begin
                display_num <= nums[3:0];
                digit <= 4'b1110;
            end
        default : begin
                display_num <= nums[3:0];
                digit <= 4'b1110;
            end
    endcase
end
```

I copy this module directly from the sample code given by TAs and make a bit change of the code to fit the requirement for this lab. I delete the **rst** event in the always block and make the clock speed faster.

Since I delete **rst** event in the always block as the picture on the left, so I need to reset **nums** to '0' when reset button is pressed. I change **nums** value in lab06 module considering that I instantiate the seven segments module in lab06 (will explain more later).

- Keyboard Decoder

I use exactly the same code as the sample code for this design. Then, I instantiate keyboard decoder module inside lab06 just as below.

```
SevenSegment seven_seg (.display(DISPLAY), .digit(DIGIT), .nums(BCD), .rst(rst), .clk(clk));
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));
```

- Use the keyboard number '1' and '2' to increase the number of people waiting at the Bus Stops B1 and B2, respectively. The maximum capacity of each bus stop is two (2) persons. LED[15:14] and LED[12:11] present the number of passengers at B1 and B2, respectively.
- The maximum capacity of the bus is also two (2).

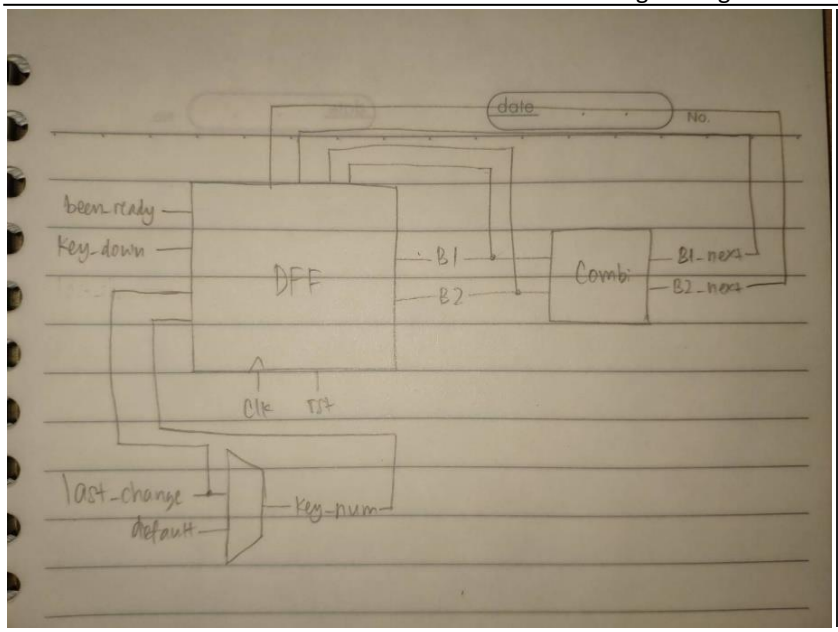
In order to design the always block as the requirement above, I declare 2-bit reg to keep in track how many people have been waiting at the bus stop. **B1** for LED[15:14] and **B2** for LED[12:11].

- If keyboard number '1' is pressed
  Check the condition of bus stop B1. If it's empty, then **B1** will become 2'b10. And if there is one person there, then B1 will become 2'b11. Otherwise, it will stay the same or depend on the condition later (in the combinational block). Here I assign it to the next state.
- If keyboard number '2' is pressed
  Same as the condition of number '1' keyboard, just different bus stop and reg.

To connect it to LED light, I assign it as below.

```
assign LED = {B1, 1'b0, B2, passenger, 2'b00, busway};
```

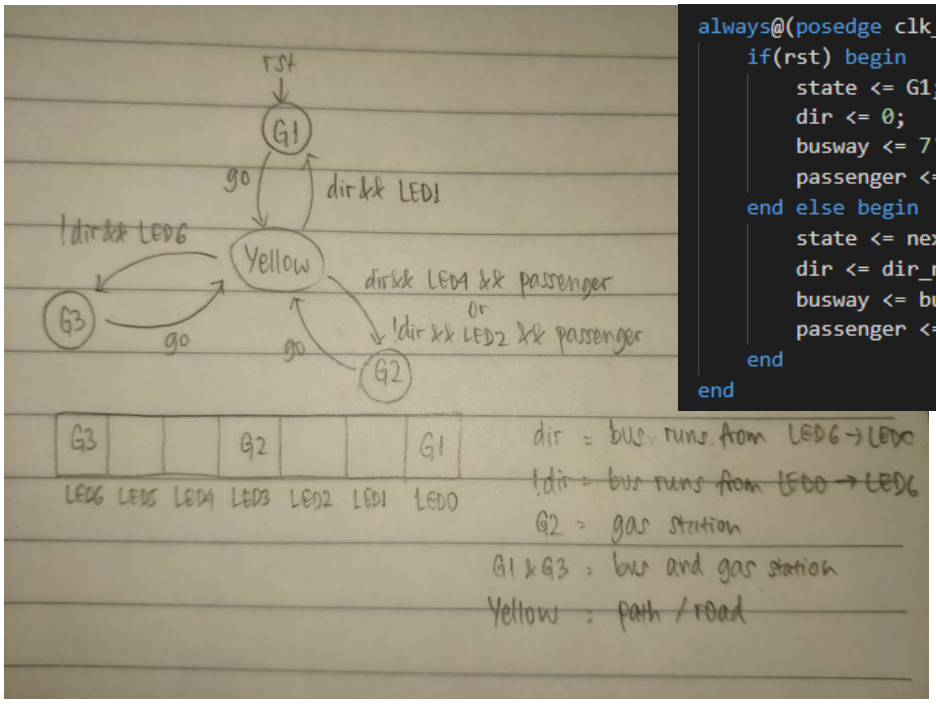
Also, LED13, LED8 and LED7 are not used, I've turned it off all the time.




```
always@(posedge clk or posedge rst) begin
    if(rst) begin
        B1 <= 2'b00;
        B2 <= 2'b00;
    end else if (been_ready && key_down[last_change] == 1'b1) begin
        if (key_num != 4'b1111) begin
            if(key_num == 4'b0001) begin
                B2 <= B2_next;
                if(B1 == 2'b00) B1 <= 2'b10;
                else if(B1 == 2'b10) B1 <= 2'b11;
                else B1 <= B1_next;
            end else if(key_num == 4'b0010) begin
                B1 <= B1_next;
                if(B2 == 2'b00) B2 <= 2'b10;
                else if(B2 == 2'b10) B2 <= 2'b11;
                else B2 <= B2_next;
            end else begin
                B1 <= B1_next;
                B2 <= B2_next;
            end
        end else begin
            B1 <= B1_next;
            B2 <= B2_next;
        end
    end else begin
        B1 <= B1_next;
        B2 <= B2_next;
    end
end
```

Besides, there also another condition, when reset button is pressed, **B1** and **B2** will become 2'b00. And neither reset and the keyboard number '1' and '2' is pressed, then **B1** and **B2** will change depend on the next state.

- Finite State Machine




```
always@(posedge clk_26 or posedge rst) begin
    if(rst) begin
        state <= G1;
        dir <= 0;
        busway <= 7'b0000_001;
        passenger <= 2'b00;
    end else begin
        state <= next_state;
        dir <= dir_next;
        busway <= busway_next;
        passenger <= passenger_next;
    end
end
```

When the machine is reset, it will go to **G1** state, bus position (**busway**) at **LED0**, **dir** become '0', also **passenger** will be cleared (**passenger** reg work the same as **B1** and **B2** in the keyboard decoder). **passenger** indicated LED[10:9] light in the board (number of passenger in the bus). Otherwise, all of them will go to next state.

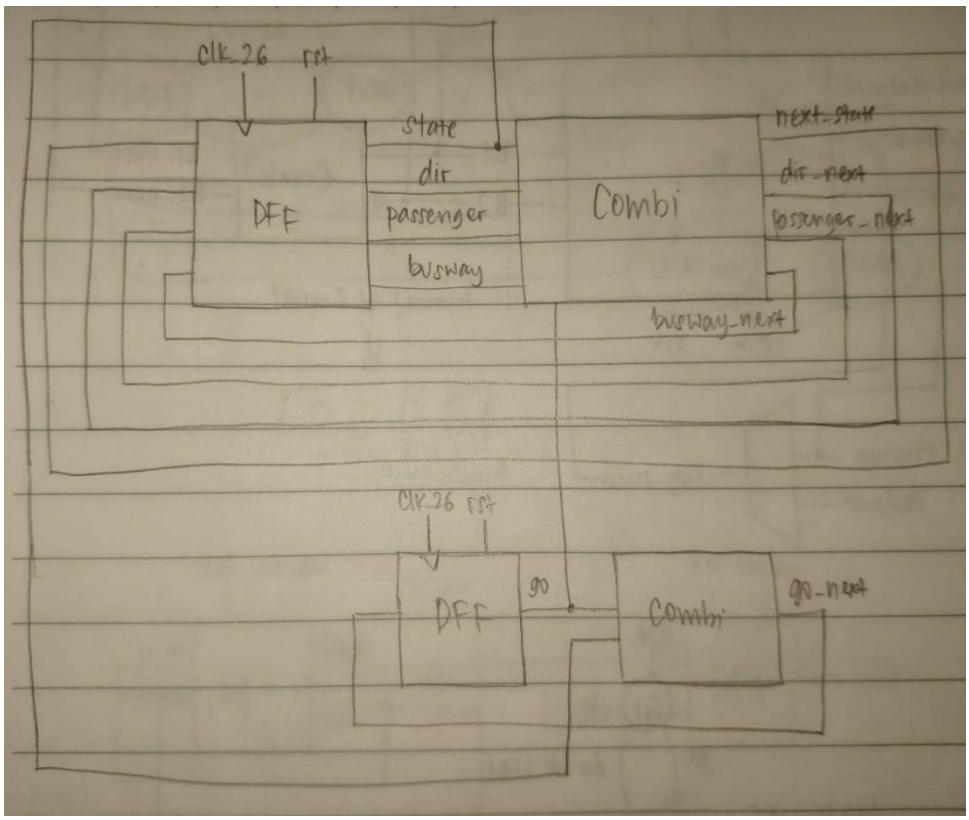
- G1, G2 and G3
  Will go to **Yellow** state if **go** signal is '1'.
- Yellow
  It will shift the position of the bus by one depend on the direction signal.
  - Go to state **G1** if the bus is in **LED1** position and **dir** is '1'.
  - Go to **G2** only if there is passenger in the bus and with the condition **dir** is '1' and bus in **LED4** or **dir** is '0' and bus position in **LED2**.

- Go to **G3** state if **dir** is '0' and position in **LED6.**

Whenever it changes to **G1**, **G2** or **G3**, it will update or decrease the fuel at that moment. The weight of fuel to decrease depend on how many people in the bus. It's 5 fuel / person.

```
Yellow : begin
    go_next = 1'b0;
    clear_next = 1'b0;
    pay_next = 1'b0;
    if(dir) begin
        if(busway == LED1 || busway == LED4) begin
            if(passenger == 2'b11) BCD_next[15:12] = BCD[15:12] - 4'd1;
            else if(passenger == 2'b10) begin
                if(BCD[11:8] == 4'd5) BCD_next[11:8] = 4'd0;
                else begin
                    BCD_next[15:12] = BCD[15:12] - 4'd1;
                    BCD_next[11:8] = 4'd5;
                end
            end
        end
    end else begin
        if(busway == LED2 || busway == LED6) begin
            if(passenger == 2'b11) BCD_next[15:12] = BCD[15:12] - 4'd1;
            else if(passenger == 2'b10) begin
                if(BCD[11:8] == 4'd5) BCD_next[11:8] = 4'd0;
                else begin
                    BCD_next[15:12] = BCD[15:12] - 4'd1;
                    BCD_next[11:8] = 4'd5;
                end
            end
        end
    end
end
```

```
always@(*) begin
    next_state = state;
    dir_next = dir;
    busway_next = busway;
    case (state)
        G1 : begin
            if(go) begin
                next_state = Yellow;
                busway_next = LED1;
            end
        end
        G2 : if(go) next_state = Yellow;
        G3 : begin
            if(go) begin
                next_state = Yellow;
                busway_next = LED5;
            end
        end
        Yellow : begin
            if(dir) begin
                busway_next = busway >> 1;
                if(busway == LED1) begin
                    next_state = G1;
                    dir_next = 1'b0;
                end else if(busway == LED4 && passenger != 2'b00) next_state = G2;
            end else begin
                busway_next = busway << 1;
                if(busway == LED2 && passenger != 2'b00) next_state = G2;
                else if(busway == LED6) begin
                    busway_next = LED6;
                    next_state = G3;
                    dir_next = 1'b1;
                end
            end
        end
    endcase
end
```



In **G1** and **G3** states, there are three things to do.

- Let the passengers go out from the bus (if there is)
  Go out one by one.

```
if(!clear) begin
    if(passenger == 2'b11) passenger_next = 2'b10;
    else if(passenger == 2'b10) passenger_next = 2'b00;
    else if(passenger == 2'b00) clear_next = 1'b1;
```

  clear = 1 (all passenger already out).
  clear = 0 (there's someone inside the bus).
- If there's someone waiting in the bus stop, let them in and pay the fee.

```
if(!pay) begin
    if(B1 == 2'b11) begin
        passenger_next = B1;
        pay_next = 1'b1;
        if(BCD[7:4] > 4'd2) begin
            BCD_next[7:4] = 4'd9;
            BCD_next[3:0] = 4'd0;
        end else BCD_next[7:4] = BCD[7:4] + 4'd6;
    end else if (B1 == 2'b10) begin
        passenger_next = B1;
        pay_next = 1'b1;
        if(BCD[7:4] > 4'd5) begin
            BCD_next[7:4] = 4'd9;
            BCD_next[3:0] = 4'd0;
        end else BCD_next[7:4] = BCD[7:4] + 4'd3;
     end else if (B2 != 2'b00) begin
        go_next = 1'b1;
     end
```

  - Passengers get in all at once and pay together.
    pay = 1 (have paid).
    pay = 0 (haven't paid).
  - The price for **G1** bus stop is 30 per person and for **G3** is 20 per person. If the amount of money now is 70 and need to add by 30, if will added until 90 only.
    BCD[7:4] is the second right most digit.
    BCD[3:0] is the right most digit.
  - If there is no one waiting in the current bus stop but there's someone waiting in the next bus stop, then **go** reg will become '1' and the bus will run towards the next bus stop without refuel.
- Refill the fuel until full (if possible), but only when there is passenger in the bus at that moment. (**G2** state will only use this condition)
  The bus will refuel only if there is passenger inside, means that the people waiting in the bus stop have get inside. Therefore, let's make the light off (either **B1_next** or **B2_next**, depend which bus stop is it).

```
end else begin
    B1_next = 2'b00;
    if((BCD[15:12] == 4'd2) || (BCD[15:12] == 4'd1 && BCD[7:4] == 4'd0)) go_next = 1'b1;
    else if(BCD[7:4] != 4'd0) begin
        if(BCD[15:12] == 4'd1 && BCD[11:8] == 4'd5) begin
            BCD_next[15:12] = 4'd2;
            BCD_next[11:8] = 4'd0;
            BCD_next[7:4] = BCD[7:4] - 4'd1;
        end else begin
            BCD_next[15:12] = BCD[15:12] + 4'd1;
            BCD_next[7:4] = BCD[7:4] - 4'd1;
        end
    end
end
```

  If the fuel already full or don't have enough money to refuel, then the bus will start running. Otherwise, while still have money, keep refill until it's full. Each refill cost about $10 and it can add for 10 amounts of fuel.
  BCD[15:12] and **BCD[11:8]** are the left most and second left most digit, to display fuel status in 7segment. Also, **BCD** will be used as **nums** in seven segment module.

2. 學到的東西與遇到的困難

Last time I was confuse why when I pressed the keyboard button, it didn't light up the LED in the board. At that time, I design my code like below.

```
if(!pay) begin
    B1_next = 2'b00;
    if(B1 == 2'b11) begin
        passenger_next = B1;
        pay_next = 1'b1;
        if(BCD[7:4] > 4'd2) begin
            BCD_next[7:4] = 4'd9;
            BCD_next[3:0] = 4'd0;
        end else BCD_next[7:4] = BCD[7:4] + 4'd6;
    end else if (B1 == 2'b10) begin
        passenger_next = B1;
        pay_next = 1'b1;
        if(BCD[7:4] > 4'd5) begin
            BCD_next[7:4] = 4'd9;
            BCD_next[3:0] = 4'd0;
        end else BCD_next[7:4] = BCD[7:4] + 4'd3;
     end else if (B2 != 2'b00) begin
        go_next = 1'b1;
     end
```

I turn off the LED in the bus stop immediately and make the LED for the passenger in the bus light up. The initial position is G1 or LED0. At that moment, the LED in B2 bus stop worked fine but not for B1. In the end, I keep watching the demo video with slow motion :). At I realize that in the video, the LED in B1 turn off after the LED in bus light up. Therefore, I tried to do the same and change my code as below.

```
if(!pay) begin
    if(B2 == 2'b11) begin
        passenger_next = B2;
        pay_next = 1'b1;
        if(BCD[7:4] > 4'd4) begin
            BCD_next[7:4] = 4'd9;
            BCD_next[3:0] = 4'd0;
        end else BCD_next[7:4] = BCD[7:4] + 4'd4;
    end else if (B2 == 2'b10) begin
        passenger_next = B2;
        pay_next = 1'b1;
        if(BCD[7:4] > 4'd6) begin
            BCD_next[7:4] = 4'd9;
            BCD_next[3:0] = 4'd0;
        end else BCD_next[7:4] = BCD[7:4] + 4'd2;
    end else if (B1 != 2'b00)begin
        go_next = 1'b1;
    end
end else begin
    B2_next = 2'b00;
    if((BCD[15:12] == 4'd2) || (BCD[15:12] == 4'd1 && BCD[7:4] == 4'd0)) go_next = 1'b1;
    else if(BCD[7:4] != 4'd0) begin
        if(BCD[15:12] == 4'd1 && BCD[11:8] == 4'd5) begin
```

And it works, but I don't know why. However, in the next day, I suddenly know why the code before didn't work. I forget that I design the **B1** and **B2** reg updated in clk or board clock cycle. That's why the LEDs for bus passenger didn't update because those LEDs were updated every clk/2^{26} clock cycle. In the future, I hope I'll remember to pay attention to the clock cycle difference.

3. 想對老師或助教說的話