DSD Lab3 107504507 資電四 吳葦誠

- Verilog code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: IPEECS
// Engineer: Wei Cheng
//////////////////////////////////////////////////////////////////////////////////


module Lab_3(
    input reset,
    input fpga_clock,
    input ps2_clock,
    input ps2_data,
    input [3:0] button,    //S4 S3 S2 S1
    output reg [7:0] Enable,
    output reg [7:0] SevenSeg_left,
    output reg [7:0] SevenSeg_right,
    output reg [15:0] LED
    );

    wire [7:0] drink_type;
    reg [7:0] price;
    reg [7:0] price_left;
    reg [7:0] price_right;
    reg [7:0] money;       //99 = 0110_0011
    reg [3:0] money_left;
    reg [3:0] money_right;
    reg [27:0] counter28;
    reg buy;                        //succes buy the = 1
    reg run_LED;                    //for not enough money
    reg [5:0] counterLED;     //tesetbench show LED counter
    reg [27:0] counterLED_fpga; //fpga show LED counter
    reg [23:0] debounce;       //counter use for debounce
    reg push;     //state means bounce time
    reg get_drink;
```

```
	reg [7:0] SevenSeg_Name_1;
	reg [7:0] SevenSeg_Name_2;
	reg [7:0] SevenSeg_Name_3;
	reg [7:0] SevenSeg_Name_4;
	reg [7:0] SevenSeg_Name_5;
	reg [7:0] SevenSeg_Name_6;
	reg [7:0] SevenSeg_Price_1;
	reg [7:0] SevenSeg_Price_2;

	ps2 ps2_1(ps2_clock, ps2_data, reset, drink_type); //detect keyboard

	always @(posedge fpga_clock)
	begin
	//case(counter28[2:1])
	case(counter28[20:19])
		2'b00:begin Enable = 8'b0001_0001; SevenSeg_left =
SevenSeg_Name_4; SevenSeg_right = SevenSeg_Price_2; end
		2'b01:begin Enable = 8'b0010_0010; SevenSeg_left =
SevenSeg_Name_3; SevenSeg_right = SevenSeg_Price_1; end
		2'b10:begin Enable = 8'b0100_0100; SevenSeg_left =
SevenSeg_Name_2; SevenSeg_right = SevenSeg_Name_6; end
		2'b11:begin Enable = 8'b1000_1000; SevenSeg_left =
SevenSeg_Name_1; SevenSeg_right = SevenSeg_Name_5; end
	endcase
	end

	always @(posedge fpga_clock or negedge reset)    //counter++ for timing
	begin
		if(!reset) begin
			counter28 <= 28'b0;
		end
		else begin
			counter28 <= counter28 + 1'b1;
		end
	end

	always @(posedge fpga_clock or negedge reset) begin
	if(!reset) begin
```

```
        SevenSeg_Price_1 <= 8'b0;
        SevenSeg_Price_2 <= 8'b0;
    end
    else begin
        if(money > 0) begin
            money_left <= money / 10;
            money_right <= money % 10;
            SevenSeg_Price_1 <= SevenSet(money_left);
            SevenSeg_Price_2 <= SevenSet(money_right);
        end
        else if(get_drink) begin
            SevenSeg_Price_1 <= SevenSet(4'd0);
            SevenSeg_Price_2 <= SevenSet(4'd0);
        end
        else if(price) begin
            price_left <= price / 10;
            price_right <= price % 10;
            SevenSeg_Price_1 <= SevenSet(price_left);
            SevenSeg_Price_2 <= SevenSet(price_right);
        end
    end
    end

    always @(posedge fpga_clock or negedge reset) begin
    if(!reset || get_drink) begin
        SevenSeg_Name_1 <= 8'b0;
        SevenSeg_Name_2 <= 8'b0;
        SevenSeg_Name_3 <= 8'b0;
        SevenSeg_Name_4 <= 8'b0;
        SevenSeg_Name_5 <= 8'b0;
        SevenSeg_Name_6 <= 8'b0;
    end
    else begin
        SevenSeg_Name_1 <= DinkSet1(drink_type);
        SevenSeg_Name_2 <= DinkSet2(drink_type);
        SevenSeg_Name_3 <= DinkSet3(drink_type);
        SevenSeg_Name_4 <= DinkSet4(drink_type);
        SevenSeg_Name_5 <= DinkSet5(drink_type);
```

```
        SevenSeg_Name_6 <= DinkSet6(drink_type);
end
end

always @(posedge fpga_clock or negedge reset) begin
        if(!reset)
                price <= 0;
        case(drink_type)
        8'h21: begin        //coca
                price <= 7'd12;
        end
        8'h1B: begin        //sprite
                price <= 7'd34;
        end
        8'h2B: begin        //fanta
                price <= 7'd56;
        end
        8'h4D: begin        //pepsi
                price <= 7'd78;
        end
        default: begin        //error
                price <= 7'd00;
        end
        endcase
end

always @(posedge fpga_clock or negedge reset) begin
        if(!reset) begin
                LED = 16'b0;
        end
        else if(buy) begin
                //case(counterLED[5:2])
                case(counterLED_fpga[27:24])
                        4'b0000: LED <= 16'b1000_0000_0000_0000;
                        4'b0001: LED <= 16'b0100_0000_0000_0000;
                        4'b0010: LED <= 16'b0010_0000_0000_0000;
                        4'b0011: LED <= 16'b0001_0000_0000_0000;
```

```
                4'b0100: LED <= 16'b0000_1000_0000_0000;
                4'b0101: LED <= 16'b0000_0100_0000_0000;
                4'b0110: LED <= 16'b0000_0010_0000_0000;
                4'b0111: LED <= 16'b0000_0001_0000_0000;

                4'b1000: LED <= 16'b0000_0000_1000_0000;
                4'b1001: LED <= 16'b0000_0000_0100_0000;
                4'b1010: LED <= 16'b0000_0000_0010_0000;
                4'b1011: LED <= 16'b0000_0000_0001_0000;

                4'b1100: LED <= 16'b0000_0000_0000_1000;
                4'b1101: LED <= 16'b0000_0000_0000_0100;
                4'b1110: LED <= 16'b0000_0000_0000_0010;
                4'b1111: LED <= 16'b0000_0000_0000_0001;
            endcase
        end
        else if(run_LED) begin
            //case(counterLED[4:2])
            case(counterLED_fpga[27:25])
                3'b000: LED <= 16'b1111_1111_1111_1111;
                3'b001: LED <= 16'b0;
                3'b010: LED <= 16'b1111_1111_1111_1111;
                3'b011: LED <= 16'b0;
                3'b100: LED <= 16'b1111_1111_1111_1111;
                3'b101: LED <= 16'b0;
                3'b110: LED <= 16'b0;
                3'b111: LED <= 16'b0;
            endcase
        end
        else
            LED <= 16'b0;
    end

    always @(posedge fpga_clock or negedge reset) begin
    if(!reset) begin
        //counterLED <= 0;
        counterLED_fpga <= 0;
    end
```

```
        else begin
            if(run_LED || buy)
                //counterLED <= counterLED + 1;
                counterLED_fpga <= counterLED_fpga + 1;
            else
                //counterLED <= 0;
                counterLED_fpga <= 0;
        end

        end

        always @(posedge fpga_clock or negedge reset) begin
        if(!reset) begin
            money <= 8'b0;
            buy <= 0;
            run_LED <= 0;
            debounce <= 0;
            push <= 0;
            get_drink <= 0;
        end
        else begin
            if(counterLED_fpga == 28'b1111_1111_1111_1111_1111_1111_1111)
begin
            //if(counterLED == 6'b111111) begin
                run_LED <= 0;
                buy <= 0;
            end
            if(push == 1) begin
                debounce <= debounce +1;
                if(debounce == 24'b1111_1111_1111_1111_1111_1111) begin
                //if(debounce == 24'b0000_0000_0000_0000_0000_1000) begin
                    push <= 0;
                    debounce <= 0;
                end
            end
            else if(button[0]    && push == 0 && !get_drink) begin
//buy
                push <= 1;
```

```
                if(price <= money) begin
                        get_drink <=1;
                        buy <= 1;
                        money <= money - price;
                end
                else begin
                        run_LED <= 1;
                end
        end
        else if(button[1] && push == 0 && !get_drink) begin        //$1
                push <= 1;
                if(money > 98)
                        money <= 7'd99;
                else
                        money <= money + 7'd1;
        end
        else if(button[2] && push == 0 && !get_drink) begin        //$10
                push <= 1;
                if(money > 89)
                        money <= 7'd99;
                else
                        money <= money + 7'd10;
        end
        else if(button[3] && push == 0 && !get_drink)begin        //$50
                push <= 1;
                if(money > 49)
                        money <= 7'd99;
                else
                        money <= money + 7'd50;
        end
    end
    end

function [7:0] SevenSet;
input [3:0] digits;

begin
    case(digits)
```

```
        4'h0: SevenSet = 8'b00111111;
        4'h1: SevenSet = 8'b00000110;
        4'h2: SevenSet = 8'b01011011;
        4'h3: SevenSet = 8'b01001111;
        4'h4: SevenSet = 8'b01100110;
        4'h5: SevenSet = 8'b01101101;
        4'h6: SevenSet = 8'b01111101;
        4'h7: SevenSet = 8'b00100111;
        4'h8: SevenSet = 8'b01111111;
        4'h9: SevenSet = 8'b01101111;
        default: SevenSet = 8'b0000_0000;

        endcase
end
endfunction

function [7:0] DinkSet1;
input [7:0] drinks;

begin
        case(drinks)
        8'h21: begin        //coca
                DinkSet1 = 8'b0011_1001;
        end
        8'h1B: begin        //sprite
                DinkSet1 = 8'b0110_1101;
        end
        8'h2B: begin        //fanta
                DinkSet1 = 8'b0111_0001;
        end
        8'h4D: begin        //pepsi
                DinkSet1 = 8'b0111_0011;
        end
        default: begin        //error
                DinkSet1 = 8'b0000_0000;
        end
        endcase
end
```

```
endfunction


function [7:0] DinkSet2;
input [7:0] drinks;

begin
	case(drinks)
	8'h21: begin	//coca
		DinkSet2 = 8'b0011_1111;
	end
	8'h1B: begin	//sprite
		DinkSet2 = 8'b0111_0011;
	end
	8'h2B: begin	//fanta
		DinkSet2 = 8'b0101_1111;
	end
	8'h4D: begin	//pepsi
		DinkSet2 = 8'b0111_1001;
	end
	default: begin	//error
		DinkSet2 = 8'b0000_0000;
	end
	endcase
end
endfunction

function [7:0] DinkSet3;
input [7:0] drinks;

begin
	case(drinks)
	8'h21: begin	//coca
		DinkSet3 = 8'b0011_1001;
	end
	8'h1B: begin	//sprite
		DinkSet3 = 8'b0101_0000;
	end
```

```
        8'h2B: begin        //fanta
                DinkSet3 = 8'b0101_0100;
        end
        8'h4D: begin        //pepsi
                DinkSet3 = 8'b0111_0011;
        end
        default: begin        //error
                DinkSet3 = 8'b0000_0000;
        end
        endcase
end
endfunction

function [7:0] DinkSet4;
input [7:0] drinks;

begin
        case(drinks)
        8'h21: begin        //coca
             DinkSet4 = 8'b0101_1111;
        end
        8'h1B: begin        //sprite
                DinkSet4 = 8'b0000_0110;
        end
        8'h2B: begin        //fanta
                DinkSet4 = 8'b0111_1000;
        end
        8'h4D: begin        //pepsi
                DinkSet4 = 8'b0110_1101;
        end
        default: begin        //error
                DinkSet4 = 8'b0000_0000;
        end
        endcase
end
endfunction

function [7:0] DinkSet5;
```

```
input [7:0] drinks;

begin
	case(drinks)
	8'h21: begin	//coca
	    DinkSet5 = 8'b0000_0000;
	end
	8'h1B: begin	//sprite
		DinkSet5 = 8'b0111_1000;
	end
	8'h2B: begin	//fanta
		DinkSet5 = 8'b0101_1111;
	end
	8'h4D: begin	//pepsi
		DinkSet5 = 8'b0000_0110;
	end
	default: begin	//error
		DinkSet5 = 8'b0000_0000;
	end
	endcase
end
endfunction

function [7:0] DinkSet6;
input [7:0] drinks;

begin
	case(drinks)
	8'h21: begin	//coca
		DinkSet6 = 8'b0000_0000;
	end
	8'h1B: begin	//sprite
	    DinkSet6 = 8'b0111_1001;
	end
	8'h2B: begin	//fanta
		DinkSet6 = 8'b0000_0000;
	end
	8'h4D: begin	//pepsi
```

```
            DinkSet6 = 8'b0000_0000;
        end
        default: begin        //error
            DinkSet6 = 8'b0000_0000;
        end
        endcase
end
endfunction

endmodule

module ps2(
        input clk,
        input data,
        input reset,
        output reg [7:0] drink
        );

        reg [7:0] data_curr;
        reg [7:0] data_pre;
        reg [3:0] b;
        reg flag;
        reg start;

        always @(negedge clk or negedge reset) begin
        if(!reset) begin
            b<=4'h1;
            flag<=1'b0;
            data_curr<=8'hf0;
            data_pre<=8'hf0;
            drink <= 0;
            start <= 0; //keyboard signal start
        end
        else begin
            if(data == 0 && !start)begin
                start <= 1;
                b <= 2;
            end
```

```
            if(data_curr == 8'hf0)
                drink <= data_pre;
            else if(flag)
                data_pre <= data_curr;

            case(b)
            1:;
            2: data_curr[0] <= data;
            3: data_curr[1] <= data;
            4: data_curr[2] <= data;
            5: data_curr[3] <= data;
            6: data_curr[4] <= data;
            7: data_curr[5] <= data;
            8: data_curr[6] <= data;
            9: data_curr[7] <= data;
            10: flag <= 1'b1;
            11: flag <= 1'b0;
            endcase
            if(b<=10) begin
                if(start)
                    b <= b + 1;
            end
            else begin
                b <= 1;
                start <= 0;
            end
        end
        end

endmodule
```

- Tesetbench

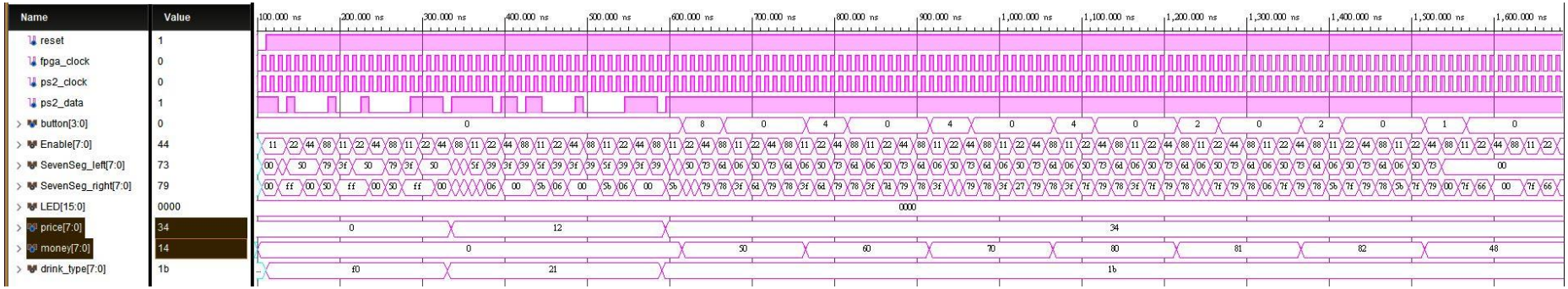
```
`timescale 1ns / 1ps
module Lab_3_tb;
    //Input
    reg reset;
    reg fpga_clock;
```

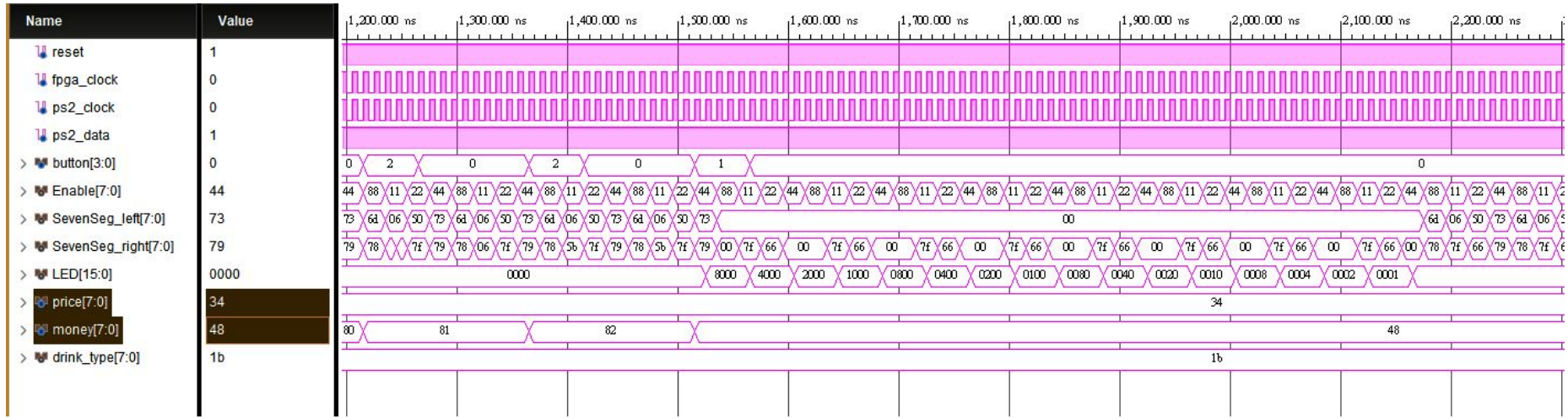
```
        reg ps2_clock;
        reg ps2_data;
        reg [3:0] button;    //S4 S3 S2 S1

        //output
        wire [7:0] Enable;
        wire [7:0] SevenSeg_left;
        wire [7:0] SevenSeg_right;
        wire [15:0] LED;

        //uut
        Lab_3 uut(
        .reset(reset),
        .fpga_clock(fpga_clock),
        .ps2_clock(ps2_clock),
        .ps2_data(ps2_data),
        .button(button),
        .Enable(Enable),
        .SevenSeg_left(SevenSeg_left),
        .SevenSeg_right(SevenSeg_right),
        .LED(LED)
        );

initial begin
        //#100;
        fpga_clock = 1'b0;
        ps2_clock = 1'b0;
        forever
        #5 begin
                fpga_clock <= ~fpga_clock ;
                ps2_clock <= ~ps2_clock;
        end
end

initial begin
        reset = 1'b1;
        ps2_data = 1'b1;
        button = 4'b0;
```

```
#100; reset = 1'b0;
#10; reset = 1'b1;

#15;
ps2_data = 1'b0 ;//START                    //c 21
#10; ps2_data = 1'b1 ;//data0
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b1 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;//data7
#10; ps2_data = 1'b0 ;//parity check
#10; ps2_data = 1'b1 ;//stop
#10; ps2_data = 1'b0 ;//START            //break f0
#10; ps2_data = 1'b0 ;//data0
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b1 ;
#10; ps2_data = 1'b1 ;
#10; ps2_data = 1'b1 ;
#10; ps2_data = 1'b1 ;//data7
#10; ps2_data = 1'b0 ;//parity check
#10; ps2_data = 1'b1 ;//stop              //325
#50;//pauese
ps2_data = 1'b0 ;//START                    //s 1b
#10; ps2_data = 1'b1 ;//data0
#10; ps2_data = 1'b1 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b1 ;
#10; ps2_data = 1'b1 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;
#10; ps2_data = 1'b0 ;//data7
#10; ps2_data = 1'b0 ;//parity check
#10; ps2_data = 1'b1 ;//stop
```

```
        #10; ps2_data = 1'b0 ;//START             //break f0
        #10; ps2_data = 1'b0 ;//data0
        #10; ps2_data = 1'b0 ;
        #10; ps2_data = 1'b0 ;
        #10; ps2_data = 1'b0 ;
        #10; ps2_data = 1'b1 ;
        #10; ps2_data = 1'b1 ;
        #10; ps2_data = 1'b1 ;
        #10; ps2_data = 1'b1 ;//data7
        #10; ps2_data = 1'b0 ;//parity check
        #10; ps2_data = 1'b1 ;//stop              //585
        #10;

        #10; button = 4'b1000;
        #50; button = 4'b0000;

        #100; button = 4'b0100;
        #50; button = 4'b0000;

        #100; button = 4'b0100;
        #50; button = 4'b0000;

        #100; button = 4'b0100;
        #50; button = 4'b0000;

        #100; button = 4'b0010;
        #50; button = 4'b0000;

        #100; button = 4'b0010;
        #50; button = 4'b0000;
        #100; button = 4'b0001;
        #50; button = 4'b0000;
    end
    endmodule
```

- Simulatuin result


Name
Value
reset
1
fpga_clock
0
ps2_clock
0
ps2_data
1
button[3:0]
0
Enable[7:0]
44
SevenSeg_left[7:0]
73
SevenSeg_right[7:0]
79
LED[15:0]
0000
price[7:0]
34
money[7:0]
14
drink_type[7:0]
1b


Name
Value
reset
1
fpga_clock
0
ps2_clock
0
ps2_data
1
button[3:0]
0
Enable[7:0]
44
SevenSeg_left[7:0]
73
SevenSeg_right[7:0]
79
LED[15:0]
0000
price[7:0]
34
money[7:0]
48
drink_type[7:0]
1b