DSD Lab 5 資電四 107504507 吳葦誠

主程式碼:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: IPEECS
// Engineer: WEI CHENG
//////////////////////////////////////////////////////////////////////////////////

module Lab_5(clk, n_rst, button, hsync, vsync, vga_r, vga_g, vga_b, Enable,
SevenSeg_right, LED);    //,lightctrl didn't know what for
    input                clk;
    input                n_rst;
    input [3:0]          button;
    //input [6:0]lightctrl;

    output               hsync,vsync;
    output [3:0]         vga_r, vga_g, vga_b;
    output reg [3:0] Enable;
    //output reg [7:0] SevenSeg_left;
    output reg [7:0] SevenSeg_right;
    output reg [15:0] LED;

    wire                 pclk;
    wire                 valid;
    wire [9:0]           h_cnt,v_cnt;
    reg [11:0]           vga_data;
    wire [11:0]          cherry_dout, ghost_dout, pac_man_dout;
    reg [11:0]           cherry_addr, ghost_addr; //4095
    reg [12:0]           pac_man_addr;                    //8191
    wire                 cherry_area, ghost_area, pac_man_area, edge_area,
wall_area, test_area;
    reg [9:0]          cherry_x,cherry_y, ghost_x,ghost_y,next_ghost_x,next_ghost_y,
    pac_man_x,pac_man_y,next_pac_man_x,next_pac_man_y;
    reg [3:0]            flag_direction; //up down left right

    reg [26:0] counter27;
    reg [7:0] SevenSeg_Score_1;
    reg [7:0] SevenSeg_Score_2;
```

```
reg [7:0] SevenSeg_Steps_1;
reg [7:0] SevenSeg_Steps_2;

reg [23:0] debounce;        //counter use for debounce
reg push;      //state means bounce time
reg Score;        //1 for showing the score
wire push_new;

reg [2:0] pac_man_x_position;
reg [2:0] pac_man_y_position;
reg [2:0] ghost_y_position;

wire button0, button1, button2, button3;

wire rst;
assign rst = !n_rst;

reg [1:0] CS;
reg [1:0] NS;
reg eat;
reg touch;
parameter ST0 = 2'd0, ST1 = 2'd1, ST2 = 2'd2;      //move    success die

reg [5:0] step;
reg [3:0] step_left;
reg [3:0] step_right;

reg [2:0] LED_counter;

parameter [9:0] pac_man_length=7'd70;
parameter [9:0] pac_man_height=7'd70;
parameter [9:0] cherry_length=6'd60;
parameter [9:0] cherry_height=6'd60;
parameter [9:0] ghost_length=6'd60;
parameter [9:0] ghost_height=6'd60;

debounce_better_version(.pb_1(button[0]), .clk(clk), .pb_out(button0));
debounce_better_version(.pb_1(button[1]), .clk(clk), .pb_out(button1));
```

```
debounce_better_version(.pb_1(button[2]), .clk(clk), .pb_out(button2));
debounce_better_version(.pb_1(button[3]), .clk(clk), .pb_out(button3));

dcm_25M u0 (
	// Clock in ports
	.clk_in1(clk),		// input clk_in1
	// Clock out ports
	.clk_out1(pclk),	// output clk_out1
	// Status and control signals
	.reset(rst));

pac_man_rom u1 (
	.clka(pclk),
	.addra(pac_man_addr),
	.douta(pac_man_dout)
	);

ghost_rom u2 (
	.clka(pclk),
	.addra(ghost_addr),
	.douta(ghost_dout)
	);

cherry_rom u3 (
	.clka(pclk),
	.addra(cherry_addr),
	.douta(cherry_dout)
	);

SyncGeneration u4 (
	.pclk(pclk),
	.reset(rst),
	.hSync(hsync),
	.vSync(vsync),
	.dataValid(valid),
	.hDataCnt(h_cnt),
	.vDataCnt(v_cnt)
	);
```

```
//FSM start
always @(eat or touch or CS) begin :COMB
	NS = CS;
	case(CS)
	ST0:begin
		Score = 1'b0;
		if(eat)
			NS = ST1;
		else if(touch)
			NS = ST2;
		else
			NS = ST0;
	end
	ST1:begin
		Score = 1;
	end
	ST2:begin
	end
	endcase
end

always @(posedge clk or posedge rst) begin :SEQ
	if(rst) begin
		CS <= ST0;
	end
	else
		CS <= NS;
end
//FSM end

always @(posedge counter27[23] or posedge rst) begin
	if(rst) begin
		LED = 16'b0000_0000_0000_0000;
	end
	else if(CS == ST2) begin
		LED_counter <= LED_counter + 1;
		case(LED_counter)
```

```
            3'b000: begin LED = 16'b1010_1010_1010_1010; end
            3'b001: begin LED = 16'b0101_0101_1010_0100; end
            3'b010: begin LED = 16'b0010_1010_1010_1000; end
            3'b011: begin LED = 16'b0001_0101_0101_0000; end
            3'b100: begin LED = 16'b0000_1010_1010_0000; end
            3'b101: begin LED = 16'b0000_0101_0100_0000; end
            3'b110: begin LED = 16'b0000_0010_1000_0000; end
            3'b111: begin LED = 16'b0000_0001_0000_0000; end
            endcase
      end
end

//Seven Segment Show Start
always @(posedge clk or posedge rst)    //counter++ for timing
begin
      if(rst) begin
            counter27 <= 27'b0;
      end
      else begin
            counter27 <= counter27 + 1'b1;
      end
end

always @(posedge clk)
begin
case(counter27[20:19])
      2'b00:begin Enable = 4'b0001; SevenSeg_right = SevenSeg_Steps_1; end
      2'b01:begin Enable = 4'b0010; SevenSeg_right = SevenSeg_Steps_2; end
      2'b10:begin Enable = 4'b0100; SevenSeg_right = SevenSeg_Score_1; end
      2'b11:begin Enable = 4'b1000; SevenSeg_right = SevenSeg_Score_2; end
endcase
end

always @(posedge clk or posedge rst)    //counter++ for timing
begin
      if(rst) begin
            SevenSeg_Score_2 <= 0;
            SevenSeg_Score_1 <= 0;
```

```
            SevenSeg_Steps_2 <= 0;
            SevenSeg_Steps_1 <= 0;
        end
        else begin
            if(Score) begin
                SevenSeg_Score_2 <= SevenSet(4'h5);
                SevenSeg_Score_1 <= SevenSet(4'h0);
            end
            else begin
                SevenSeg_Score_2 <= SevenSet(4'h0);
                SevenSeg_Score_1 <= SevenSet(4'h0);
            end
            step_left <= step /10;
            step_right = step%10;
            SevenSeg_Steps_2 <= SevenSet(step_left);
            SevenSeg_Steps_1 <= SevenSet(step_right);
        end
end
//Seven Segment Show End

debounce u5(.sig_in(push), .clk(clk), .sig_out(push_new));

//Button Start
always @(posedge clk or posedge rst) begin
if(rst) begin
        debounce <= 0;
        push <= 0;
        next_pac_man_x <= 10'd6;
        next_pac_man_y <= 10'd406;
        pac_man_x_position <= 0;
        pac_man_y_position <= 0;
        step = 6'b0;
end
else begin
        if(push == 1) begin
            debounce <= debounce +1;
            if(debounce == 24'b1111_1111_1111_1111_1111_1111) begin
                push <= 0;
```

```
            debounce <= 0;
            flag_direction <= 4'b0000;
        end
    end
    else if(button0    && push_new == 0) begin                //S0 button right
        push <= 1;
        if(pac_man_x_position == 1 && pac_man_y_position == 5 ||
        pac_man_x_position == 2 && pac_man_y_position == 4 ||
        pac_man_x_position == 2 && pac_man_y_position == 3 ||
        pac_man_x_position == 2 && pac_man_y_position == 1 ||
        pac_man_x_position == 5 && pac_man_y_position == 2 ||
        pac_man_x_position == 5 && pac_man_y_position == 3||
        pac_man_x_position == 7) begin
            pac_man_x_position <= pac_man_x_position;
            pac_man_y_position <= pac_man_y_position;
        end
        else begin
            next_pac_man_x <= next_pac_man_x + 10'd80;
            pac_man_x_position <= pac_man_x_position + 1;
            pac_man_y_position <= pac_man_y_position;
            step <= step + 1;
        end
    end
    else if(button1 && push_new == 0) begin        //S1 button down
        push <= 1;
        if(pac_man_x_position == 0 && pac_man_y_position == 4 ||
        pac_man_x_position == 3 && pac_man_y_position == 2 ||
        pac_man_x_position == 4 && pac_man_y_position == 2 ||
        pac_man_x_position == 6 && pac_man_y_position == 4 ||
        pac_man_x_position == 7 && pac_man_y_position == 4 ||
        pac_man_y_position == 0) begin
            pac_man_x_position <= pac_man_x_position;
            pac_man_y_position <= pac_man_y_position;
        end
        else begin
            next_pac_man_y <= next_pac_man_y + 10'd80;
            pac_man_x_position <= pac_man_x_position;
            pac_man_y_position <= pac_man_y_position - 1;
```

```
            step <= step + 1;
        end
    end
    else if(button2 && push_new == 0) begin        //S3 button left
        push <= 1;
        if(pac_man_x_position == 1 && pac_man_y_position == 2 ||
        pac_man_x_position == 1 && pac_man_y_position == 3 ||
        pac_man_x_position == 4 && pac_man_y_position == 5 ||
        pac_man_x_position == 4 && pac_man_y_position == 4 ||
        pac_man_x_position == 4 && pac_man_y_position == 3 ||
        pac_man_x_position == 5 && pac_man_y_position == 1 ||
        pac_man_x_position == 0) begin
            pac_man_x_position <= pac_man_x_position;
            pac_man_y_position <= pac_man_y_position;
        end
        else begin
            next_pac_man_x <= next_pac_man_x - 10'd80;
            pac_man_x_position <= pac_man_x_position - 1;
            pac_man_y_position <= pac_man_y_position;
            step <= step + 1;
        end
    end
    else if(button3 && push_new == 0)begin        //S4 button up
        push <= 1;
        if(pac_man_x_position == 0 && pac_man_y_position == 1 ||
        pac_man_x_position == 2 && pac_man_y_position == 4 ||
        pac_man_x_position == 3 && pac_man_y_position == 2 ||
        pac_man_x_position == 3 && pac_man_y_position == 0 ||
        pac_man_x_position == 4 && pac_man_y_position == 0 ||
        pac_man_x_position == 6 && pac_man_y_position == 1 ||
        pac_man_x_position == 7 && pac_man_y_position == 1 ||
        pac_man_y_position == 5) begin
            pac_man_x_position <= pac_man_x_position;
            pac_man_y_position <= pac_man_y_position;
        end
        else begin
            next_pac_man_y <= next_pac_man_y - 10'd80;
            pac_man_x_position <= pac_man_x_position;
```

```
                pac_man_y_position <= pac_man_y_position + 1;
                step <= step + 1;
            end
        end
        else begin
            next_pac_man_x <= next_pac_man_x;
            next_pac_man_y <= next_pac_man_y;
            pac_man_x_position <= pac_man_x_position;
            pac_man_y_position <= pac_man_y_position;
            step <= step;
        end
    end
    end
    //Button End

    assign pac_man_area = ((v_cnt >= pac_man_y) & (v_cnt <= pac_man_y +
pac_man_height - 1) & (h_cnt >= pac_man_x) & (h_cnt <= pac_man_x +
pac_man_length - 1)) ? 1'b1 : 1'b0;
    assign ghost_area = ((v_cnt >= ghost_y) & (v_cnt <= ghost_y + ghost_height - 1)
& (h_cnt >= ghost_x) & (h_cnt <= ghost_x + ghost_length - 1)) ? 1'b1 : 1'b0;
    assign cherry_area = ((v_cnt >= cherry_y) & (v_cnt <= cherry_y + cherry_height -
1) & (h_cnt >= cherry_x) & (h_cnt <= cherry_x + cherry_length - 1)) ? 1'b1 : 1'b0;
    assign edge_area = (((v_cnt < 10'd6 || v_cnt > 10'd475) & (h_cnt <= 10'd80 ||
h_cnt > 10'd160)) || (h_cnt < 10'd6 || h_cnt > 10'd635)) ? 1'b1 : 1'b0;
    assign wall_area = ((v_cnt >= 10'd161) & (v_cnt <= 10'd320) & (h_cnt >= 10'd1)
& (h_cnt <= 10'd80)) |
    ((v_cnt >= 10'd1) & (v_cnt <= 10'd80) & (h_cnt >= 10'd161) & (h_cnt <=
10'd320)) |
    ((v_cnt >= 10'd81) & (v_cnt <= 10'd240) & (h_cnt >= 10'd241) & (h_cnt <=
10'd320)) |
    ((v_cnt >= 10'd321) & (v_cnt <= 10'd400) & (h_cnt >= 10'd241) & (h_cnt <=
10'd400)) |
    ((v_cnt >= 10'd161) & (v_cnt <= 10'd320) & (h_cnt >= 10'd481) & (h_cnt <=
10'd640)) ? 1'b1 : 1'b0;

    always @(posedge pclk or posedge rst)
    begin: pic_display
        if (rst == 1'b1) begin
```

```
        pac_man_addr<=13'd0;
        ghost_addr<=12'd0;
        cherry_addr<=12'd0;
        vga_data <= 12'h000;
    end
    else begin
        if (valid == 1'b1) begin
            if (ghost_area == 1'b1) begin
                ghost_addr <= ghost_addr + 12'd1;
                vga_data <= ghost_dout;
            end
            else if (pac_man_area == 1'b1 && CS != ST2) begin
                pac_man_addr <= pac_man_addr + 13'd1;
                vga_data <= pac_man_dout;
            end
            else if (cherry_area == 1'b1    && CS != ST1) begin
                cherry_addr <= cherry_addr + 12'd1;
                vga_data <= cherry_dout;
            end
            else if (edge_area == 1'b1) begin
                vga_data <= 12'h00f;
            end
            else if (wall_area == 1'b1) begin
                vga_data <= 12'h00f;
            end
            else begin
                pac_man_addr <= pac_man_addr;
                ghost_addr <= ghost_addr;
                cherry_addr <= cherry_addr;
                vga_data <= 12'b000000000000;
            end
        end
        else begin
            vga_data <= 12'h000;
            if (v_cnt == 0) begin
                pac_man_addr<=14'd0;
                ghost_addr<=14'd0;
                cherry_addr<=14'd0;
```

```
                end
                else begin
                        pac_man_addr <= pac_man_addr;
                        ghost_addr <= ghost_addr;
                        cherry_addr <= cherry_addr;
                end
            end
        end
end

assign {vga_r,vga_g,vga_b} = vga_data;

always@(posedge counter27[25] or posedge rst)    begin
        if(ghost_y_position == 3'd5) begin
            ghost_y_position <= 3'd0;
            next_ghost_y <= 10'd331;
        end
        else if(ghost_y_position == 3'd4) begin
            next_ghost_y <= 10'd411;
            ghost_y_position <= ghost_y_position + 1;
        end
        else if(ghost_y_position == 3'd3) begin
            next_ghost_y <= 10'd11;
            ghost_y_position <= ghost_y_position + 1;
        end
        else if(ghost_y_position == 3'd2) begin
            next_ghost_y <= 10'd91;
            ghost_y_position <= ghost_y_position + 1;
        end
        else if(ghost_y_position == 3'd1) begin
            next_ghost_y <= 10'd171;
            ghost_y_position <= ghost_y_position + 1;
        end
        else if(ghost_y_position == 3'd0) begin
            next_ghost_y <= 10'd251;
            ghost_y_position <= ghost_y_position + 1;
        end
```

```
            if (rst) begin
                pac_man_x <= 10'd6;
                pac_man_y <= 10'd406;
                ghost_x <= 10'd91;
                ghost_y <= 10'd331;
                cherry_x <= 10'd571;
                cherry_y <= 10'd11;
                next_ghost_y <= 10'd251;
                ghost_y_position <= 1;
                eat = 1'b0;
                touch = 1'b0;
            end
            else if(CS == ST0) begin

                if(pac_man_x == 10'd566 && pac_man_y == 10'd6) begin
                    eat = 1;
                end
                else if(pac_man_y == ghost_y - 10'd5 && pac_man_x == ghost_x -
10'd5) begin
                    touch <= 1;
                    ghost_y <= ghost_y;
                end
                else begin
                    pac_man_x <= next_pac_man_x;
                    pac_man_y <= next_pac_man_y;
                    ghost_y <= next_ghost_y;
                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'b1111_1111;

        endcase
end
endfunction

endmodule

module debounce_better_version(input pb_1,clk,output pb_out);
wire slow_clk_en;
wire Q1,Q2,Q2_bar,Q0;
clock_enable u1(clk,slow_clk_en);
my_dff_en d0(clk,slow_clk_en,pb_1,Q0);

my_dff_en d1(clk,slow_clk_en,Q0,Q1);
my_dff_en d2(clk,slow_clk_en,Q1,Q2);
assign Q2_bar = ~Q2;
assign pb_out = Q1 & Q2_bar;
endmodule
// Slow clock enable for debouncing button
module clock_enable(input Clk_100M,output slow_clk_en);
        reg [26:0]counter=0;
        always @(posedge Clk_100M)
        begin
            counter <= (counter>=249999)?0:counter+1;
        end
        assign slow_clk_en = (counter == 249999)?1'b1:1'b0;
endmodule
// D-flip-flop with clock enable signal for debouncing module
module my_dff_en(input DFF_CLOCK, clock_enable,D, output reg Q=0);
        always @ (posedge DFF_CLOCK) begin
```

```
    if(clock_enable==1)
                    Q <= D;
        end
endmodule
```