DSD HW 3 資電四 107504507 吳葦誠

a. Explain your design

使用一個 FSM 進行 state 的切換

一個 sequential always 讓電梯及手裡劍在 state != die 的時候持續向下及向上

一個 combinatorial always 偵測現在位置是不是發生了 drop 或 touch

一個 sequential always 偵測現在位置以及 input 是否會讓我得分或是拿到鑰匙並且如果 on_Elevator 是 true 就 NinjaY += 1(並且判斷是下個 cycle 上升或下下個)

一個 sequential always 偵測現在位置以及 input 是否會改變 on_Elevator 的值

一個 sequential always 偵測 input 來改變 NinjaX

本作業因無特別說明，因此當我死亡後手裡劍與電梯均會重置

b. Verilog code

```
`timescale 1ns / 1ps

module HW_3(
    input clk_input,        //clock
    input reset,             //for initial
    input start,        //for starting the game
    input go_right, input go_left,    ////for direction movement
    output reg [2:0] NinjaX, output reg [3:0] NinjaY,      //for ninja position
    output reg [3:0] Elevator1Y, output reg [3:0] Elevator2Y, output reg [3:0] Elevator3Y,        //for elevator position
    output reg [3:0] Shurikan1Y, output reg [3:0] Shurikan2Y, output reg [3:0] Shurikan3Y,        //for shurikan position
    output reg touch,      //true for shurikan touch ninja
    output reg drop,        //true for ninja drop dead
    output reg [6:0] score,      //saving the score 50 for 0110010, 100 for 1100100
    output reg [1:0] chance,    //chance start with 2
    output reg key, //true for key get
    output reg on_Elevator,    //true for on elevator
    output reg [1:0] CS
    );

    reg [1:0] NS;
    parameter ST0 = 2'd0, ST1 = 2'd1, ST2 = 2'd2, ST3 = 2'd3;

    initial begin
```

```
        CS = 2'b00;
        NinjaX = 3'd6; NinjaY = 4'd0;
        Elevator1Y = 4'b0010; Elevator2Y = 4'b0110; Elevator3Y = 4'b1010;
        Shurikan1Y = 4'b1000; Shurikan2Y = 4'b0101; Shurikan3Y = 4'b0010;
        touch = 1'b0;      drop = 1'b0;      score = 7'b0;
        chance = 2'b10; key = 1'b0;        on_Elevator = 1'b0;
end

//FSM start
always @(start or touch or drop or on_Elevator or chance or CS) begin :COMB
        NS = CS;
        case(CS)
        ST0:begin
                if(start)
                        NS = ST1;
                else
                        NS = ST0;
        end
        ST1:begin
                if(on_Elevator)
                        NS = ST2;
                else if(touch || drop) begin
                        NS = ST3;
                end
        end
        ST2:begin
                if(!on_Elevator)
                        NS = ST1;
                else if(touch || drop) begin
                        NS = ST3;
                end
        end
        ST3:begin
                if(chance == 2'b0)
                        NS = ST0;
                else
                        NS = ST1;
        end
```

```
        endcase
    end

    always @(posedge clk_input or posedge reset) begin :SEQ
        if(reset) begin
            CS <= ST0;
        end
        else
            CS <= NS;
    end

    always @(CS) begin :OUT

    end
    //FSM end

    always @(posedge clk_input or posedge reset) begin
        if(reset    || CS == ST3) begin
            Elevator1Y = 4'b0010; Elevator2Y = 4'b0110; Elevator3Y = 4'b1010;
            Shurikan1Y = 4'b1000; Shurikan2Y = 4'b0101; Shurikan3Y = 4'b0010;
        end
        else if(NS != ST0) begin
            Elevator1Y <= elevator_move(Elevator1Y); Elevator2Y <=
elevator_move(Elevator2Y); Elevator3Y <= elevator_move(Elevator3Y);
            Shurikan1Y <= shurikan_move(Shurikan1Y); Shurikan2Y <=
shurikan_move(Shurikan2Y); Shurikan3Y <= shurikan_move(Shurikan3Y);
        end
    end

    always @(posedge clk_input or posedge reset) begin
        if (reset) begin
            chance = 2'b10;
        end
        else if(touch || drop)
            chance <= chance - 1'b1;
        else
            chance <= chance;
    end
```

```
    always @(*) begin
        touch = 1'b0;
        drop = 1'b0;
        if(reset) begin
            touch = 1'b0;
            drop = 1'b0;
        end
        else if(CS == ST3) begin
            touch = 1'b0;
            drop = 1'b0;
        end
        else if(NinjaX == 3'd2 && (Shurikan1Y == NinjaY || Shurikan2Y == NinjaY ||
Shurikan3Y == NinjaY)) begin
                touch = 1'b1;
        end
        else begin
            if(NinjaX == 3'd0 || NinjaX == 3'd1 || NinjaX == 3'd2) begin
                if(NinjaY != 4'd5) begin
                    drop = 1'b1;
                end
            end
            else if(NinjaX == 3'd3 || NinjaX == 3'd4) begin
                if(NinjaY == Elevator1Y || NinjaY == Elevator2Y || NinjaY ==
Elevator3Y ||
                (NinjaY == Elevator1Y + 1'b1) || (NinjaY == Elevator2Y + 1'b1) ||
(NinjaY == Elevator3Y + 1'b1))
                    drop = 1'b0;
                else begin
                    drop = 1'b1;
                end
            end
            else if(NinjaX == 3'd5 || NinjaX == 3'd6 || NinjaX == 3'd7) begin
                if(NinjaY == 4'd0 || NinjaY == 4'd11)
                    drop = 1'b0;
                else begin
                    drop = 1'b1;
                end
```

```
            end
        end
    end

    always @(posedge reset or posedge clk_input) begin
        if(reset || CS == ST3) begin
            if(reset)
                score = 7'b0;
            NinjaY = 4'd0;
        end
        else if((NinjaX == 3'd0 || NinjaX == 3'd1 || NinjaX == 3'd2) && NinjaY ==
4'd5) begin
            if(NinjaX == 3'd1 && go_left) begin
                score <= score + 7'd50;
                key <= 1'b1;
            end
        end
        else if(on_Elevator == 1'b1) begin
            if(NinjaY == Elevator1Y || NinjaY == Elevator2Y ||NinjaY ==
Elevator3Y) begin
                NinjaY <= elevator_move(NinjaY);
            end
            else
                NinjaY <= NinjaY;
        end
        else if((NinjaX == 3'd5 || NinjaX == 3'd6 || NinjaX == 3'd7)&&(NinjaY ==
4'd0 || 4'd11)) begin
            if(NinjaX == 3'd6 && NinjaY == 4'd11 && go_right && key) begin
                score <= score + 7'd50;
            end
        end
        else if((NinjaX == 3'd3 && NinjaY == 4'd5) || (NinjaX == 3'd4 && NinjaY ==
4'd11)) begin

        end
    end

    always @(negedge clk_input or posedge reset) begin
```

```
        if(reset) begin
            on_Elevator = 1'b0;
        end
        else if(NinjaX == 4'd2 && go_right &&
            (Elevator1Y == 4'd4 || Elevator1Y == 4'd5 ||
            Elevator2Y == 4'd4 || Elevator2Y == 4'd5 ||
            Elevator3Y == 4'd4 || Elevator3Y == 4'd5)) begin
            on_Elevator = 1'b1;
        end
        else if(NinjaX == 4'd3 && NinjaY == 4'd5 && go_left) begin
            on_Elevator = 1'b0;
        end
        else if(NinjaX == 4'd4 && NinjaY == 4'd11 && go_right) begin
            on_Elevator = 1'b0;
        end
        else if(NinjaX == 4'd5 && NinjaY == 4'd0 && go_left) begin
            on_Elevator = 1'b1;
        end
    end

    always @(posedge clk_input or posedge reset) begin
        if(reset || CS == ST3) begin
            NinjaX = 3'd6;
        end
        else if(CS != ST0) begin
            if(go_right)
                NinjaX <= NinjaX + 1'b1;
            else if(go_left) begin
                NinjaX <= NinjaX - 1'b1;
            end
        end
    end

    //finite state machine must use 2 process or 3 process

function [3:0] elevator_move;
input [3:0] old;
begin
```

```
        if(old == 4'b1011) begin
                elevator_move = 4'b0;
        end
        else begin
                elevator_move = old + 4'b0001;
        end
end
endfunction

function [3:0] shurikan_move;
input [3:0] old;
begin
        if(old == 4'b0000) begin
                shurikan_move = 4'b1000;
        end
        else begin
                shurikan_move = old - 4'b0001;
        end
end
endfunction

endmodule
```

c. Test bench

(a).

```
        `timescale 1ns / 1ps

        module HW_3_tb;
                //input
                reg clk;
                reg reset;
                reg start;
                reg go_right;
                reg go_left;
                //output
                wire [2:0] NinjaX; wire [3:0] NinjaY;
                wire [3:0] Elevator1Y; wire [3:0] Elevator2Y; wire [3:0] Elevator3Y;
                wire [3:0] Shurikan1Y; wire [3:0] Shurikan2Y; wire [3:0] Shurikan3Y;
                wire touch;
```

```
	wire drop;
	wire [6:0] score;
	wire [1:0] chance;
	wire key;
	wire on_Elevator;
	wire [1:0] CS;

	HW_3 uut(
	.clk_input(clk),
	.reset(reset),
	.start(start),
	.go_right(go_right), .go_left(go_left),
	.NinjaX(NinjaX), . NinjaY( NinjaY),
	.Elevator1Y(Elevator1Y), .Elevator2Y(Elevator2Y), .Elevator3Y(Elevator3Y),
	.Shurikan1Y(Shurikan1Y), .Shurikan2Y(Shurikan2Y), .Shurikan3Y(Shurikan3Y
),
	.touch(touch), .drop(drop), .score(score),
	.chance(chance), .key(key), .on_Elevator(on_Elevator),
	.CS(CS)
	);

	initial begin
		#100;
		clk = 1'b1 ;
		forever
		#50 clk = ~clk ;
	end

	initial begin
		#100;
		reset = 1'b1; start = 1'b0; go_left = 1'b0; go_right = 1'b0;
		#50;	reset = 1'b0;
		#100;	start = 1'b1;
		#100;	start = 1'b0; go_left = 1'b1;
		#300;	go_left = 1'b0;
		#300;	go_left = 1'b1;
		#300;	go_left = 1'b0; go_right = 1'b1;
		#100;	go_right = 1'b0;
```

```
            #100;     go_right = 1'b1;
            #100;     go_right = 1'b0;
            #100;     go_right = 1'b1;
            #200;     go_right = 1'b0;
            #500;     go_right = 1'b1;
            #300;     go_right = 1'b0;
        end

endmodule
```

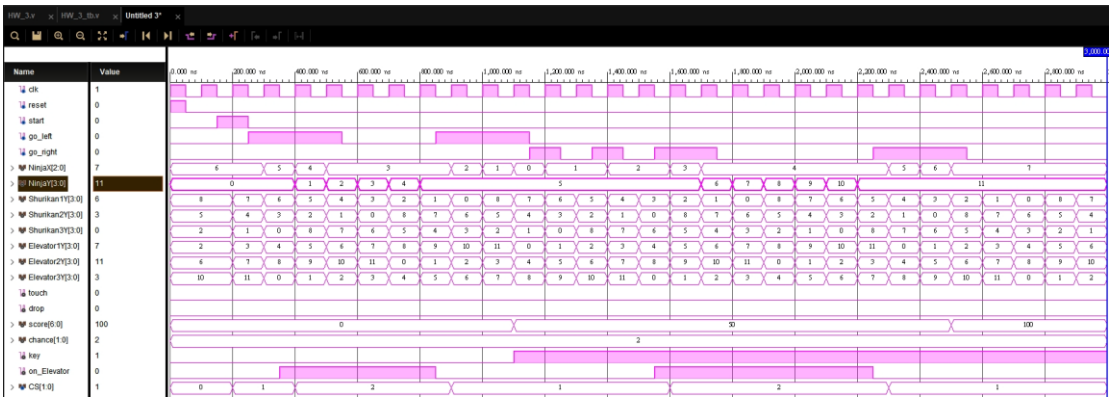
(b).

```
 `timescale 1ns / 1ps

 module HW_3_tb_b;
        //input
        reg clk;
        reg reset;
        reg start;
        reg go_right;
        reg go_left;
        //output
        wire [2:0] NinjaX; wire [3:0] NinjaY;
        wire [3:0] Elevator1Y; wire [3:0] Elevator2Y; wire [3:0] Elevator3Y;
        wire [3:0] Shurikan1Y; wire [3:0] Shurikan2Y; wire [3:0] Shurikan3Y;
        wire touch;
        wire drop;
        wire [6:0] score;
        wire [1:0] chance;
        wire key;
        wire on_Elevator;
        wire [1:0] CS;

        HW_3 uut(
        .clk_input(clk),
        .reset(reset),
        .start(start),
        .go_right(go_right), .go_left(go_left),
        .NinjaX(NinjaX), . NinjaY( NinjaY),
```

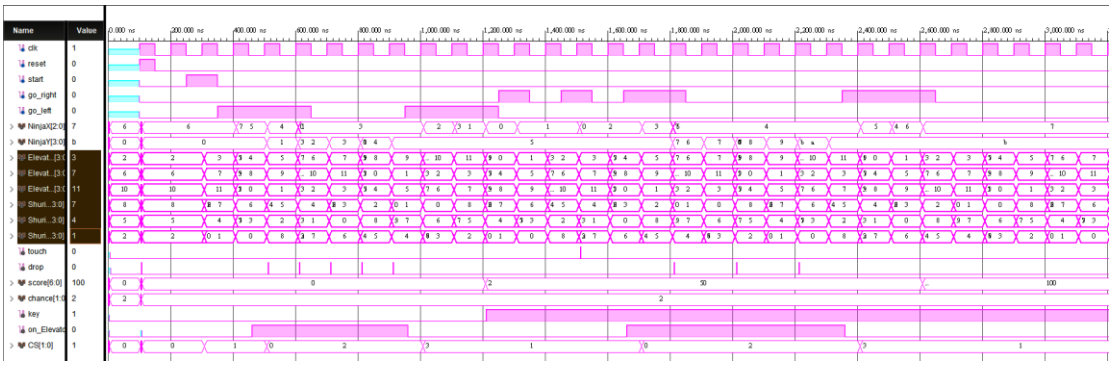
```
        .Elevator1Y(Elevator1Y), .Elevator2Y(Elevator2Y), .Elevator3Y(Elevator3Y),
        .Shurikan1Y(Shurikan1Y), .Shurikan2Y(Shurikan2Y), .Shurikan3Y(Shurikan3Y
 ),
        .touch(touch), .drop(drop), .score(score),
        .chance(chance), .key(key), .on_Elevator(on_Elevator),
        .CS(CS)
        );

        initial begin
            #100;
            clk = 1'b1 ;
            forever
            #50 clk = ~clk ;
        end

        initial begin
            #100;
            reset = 1'b1; start = 1'b0; go_left = 1'b0; go_right = 1'b0;
            #50;        reset = 1'b0;
            #100;       start = 1'b1;
            #100;       start = 1'b0;
            //round 1
            go_left = 1'b1;
            #300;       go_left = 1'b0;
            #300;       go_left = 1'b1; //850
            #100;       go_left = 1'b0;
            //round 2
            #300;
            go_left = 1'b1;
            #200; go_left = 1'b0;
            #1100; go_right = 1'b1;
            #300; go_right = 1'b0;
            #100; go_left = 1'b1;
            #400; go_left = 1'b0;
        end

endmodule
```

d. Q4 waveform

Behavior simulation

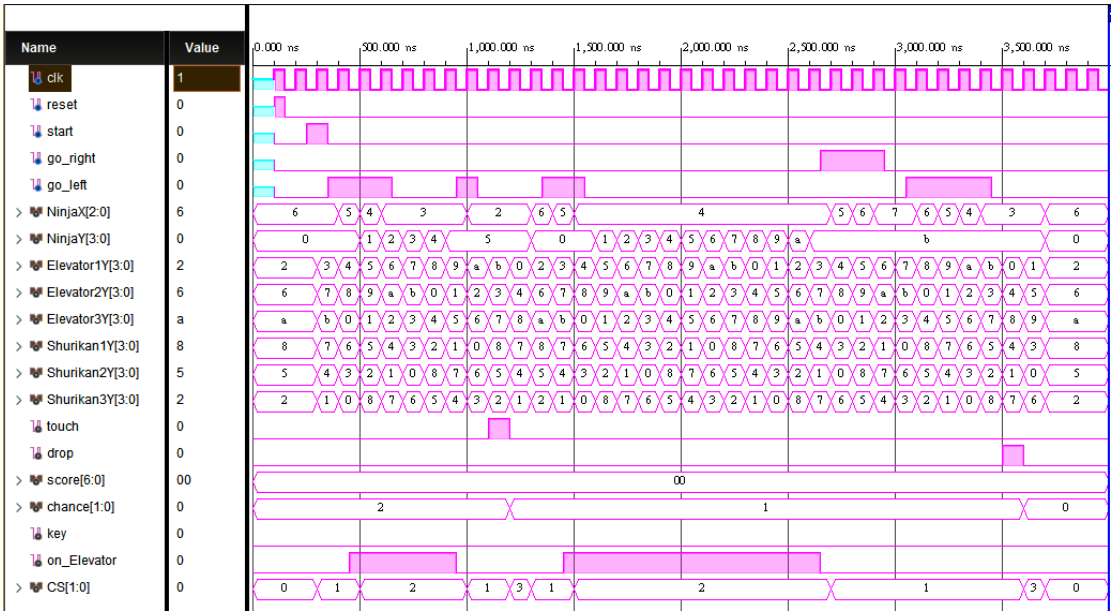


Post implements simulation



e. Q5 waveform

Behavior simulation



Post implements simulation(有些地方的值被暫態重疊所以放兩張)