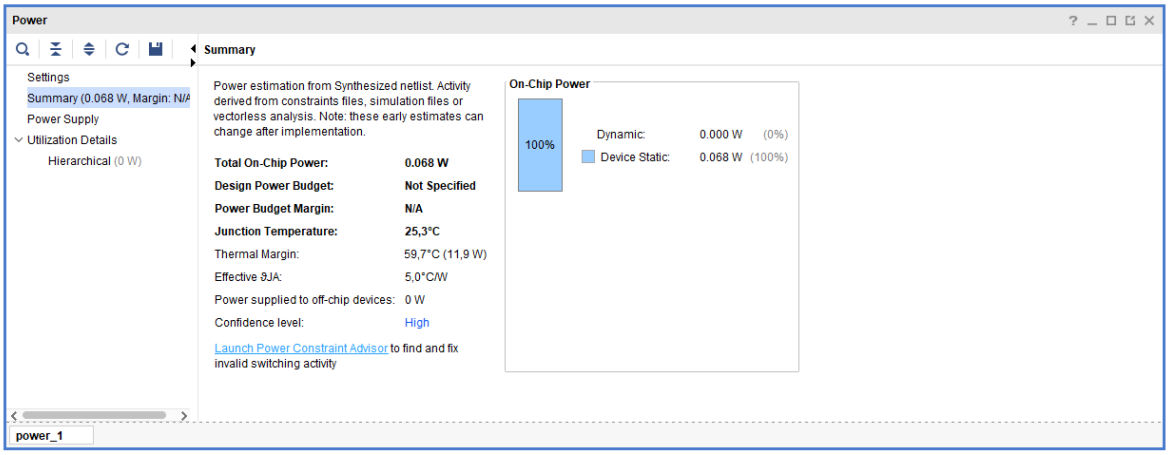
Lab 2: Sequential Logic

Question 1

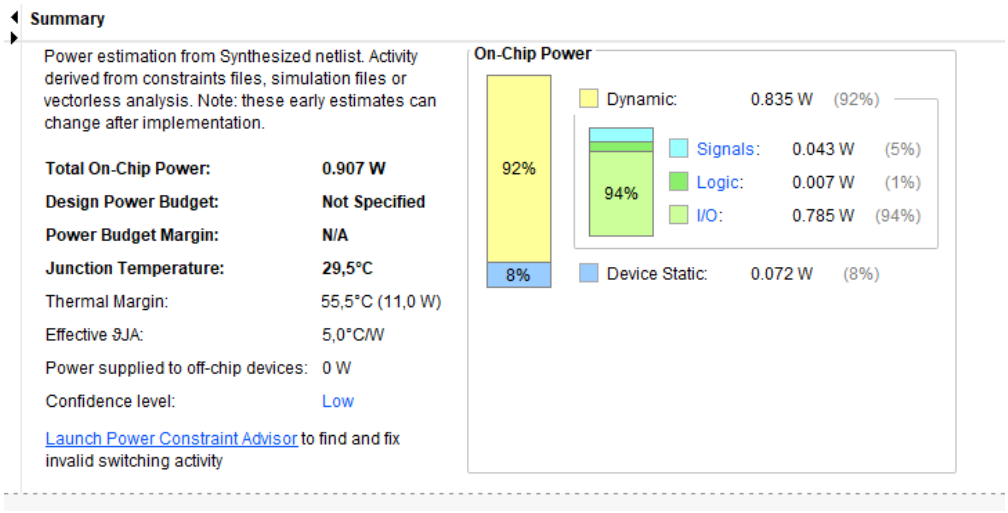
Academic D-FF

Power Report



Industry D-FF

Power Report



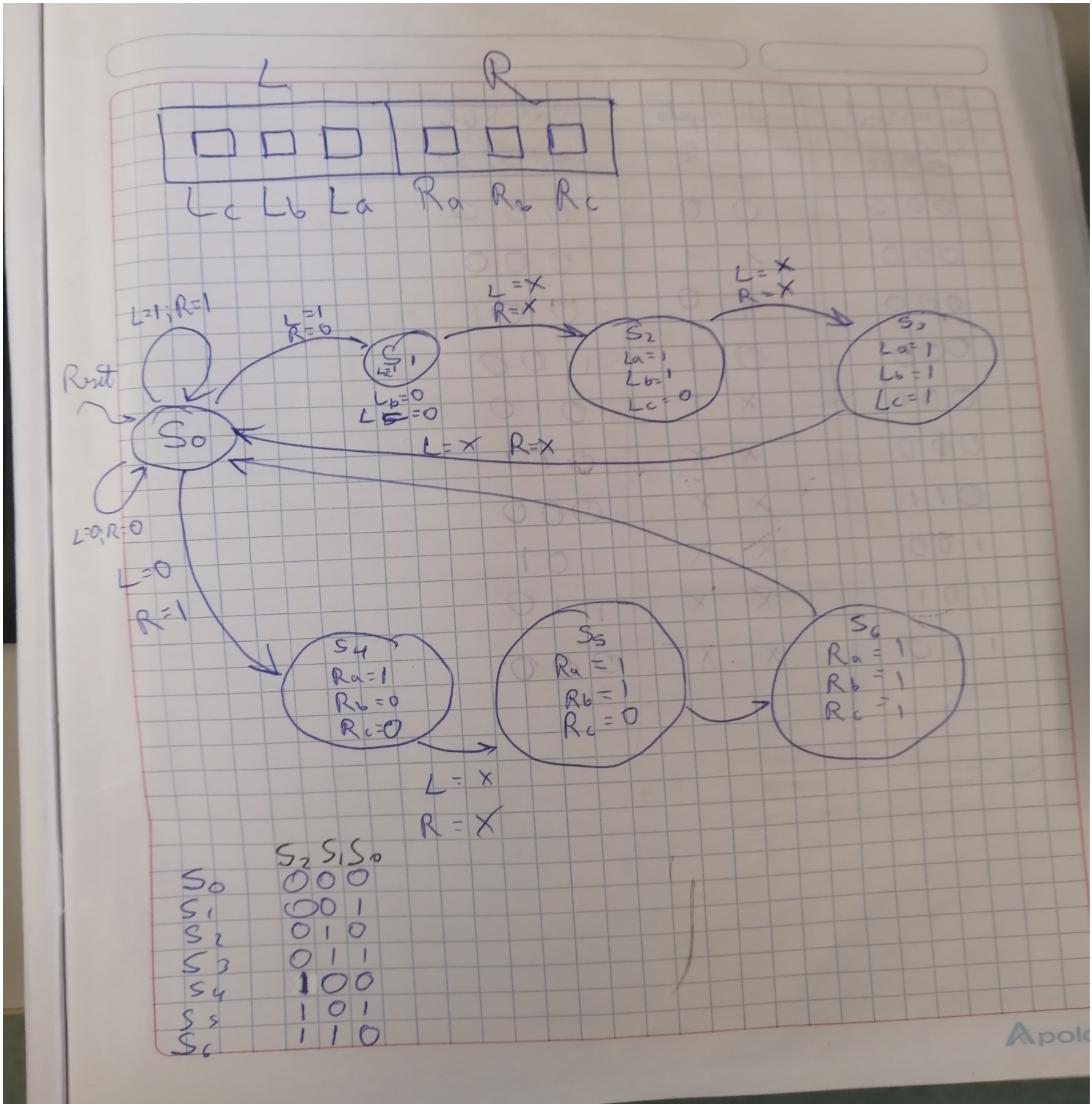
Differences

Vemos que el Industry Flip Flop posee señales, inputs/outputs y lógica, mientras que el academy, no. Podemos determinar que esto es concluyente para un mayor total on-power chip del industry (0.907W) y un menos consumo por parte del academy (0.068W).

Question 2

Comprobamos que funciona!

FSM Schematic



UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

State Encoding

S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110

Next Logic

State $S_2 S_1 S_0$	Inputs L R	Next State $S_2' S_1' S_0'$
0 0 0	0 0	0 0 0
0 0 0	0 1	1 0 0
0 0 0	1 0	0 0 1
0 0 0	1 1	0 0 0
0 0 1	X X	0 1 0
0 1 0	X X	0 1 1
0 1 1	X X	0 0 0
1 0 0	X X	1 0 1
1 0 1	X X	1 1 0
1 1 0	X X	0 0 0

$$S_2' = \overline{S_2}\,\overline{S_1}\,\overline{S_0} + S_2\overline{S_1}\,\overline{S_0} + S_2\overline{S_1}S_0$$
$$S_1' = \overline{S_2}S_1\overline{S_0} + \overline{S_2}S_1S_0 + S_2\overline{S_1}\,\overline{S_0}$$
$$S_0' = \overline{S_2}\,\overline{S_1}S_0 + \overline{S_2}S_1\overline{S_0} + S_2\overline{S_1}\,\overline{S_0}$$

Output Logic

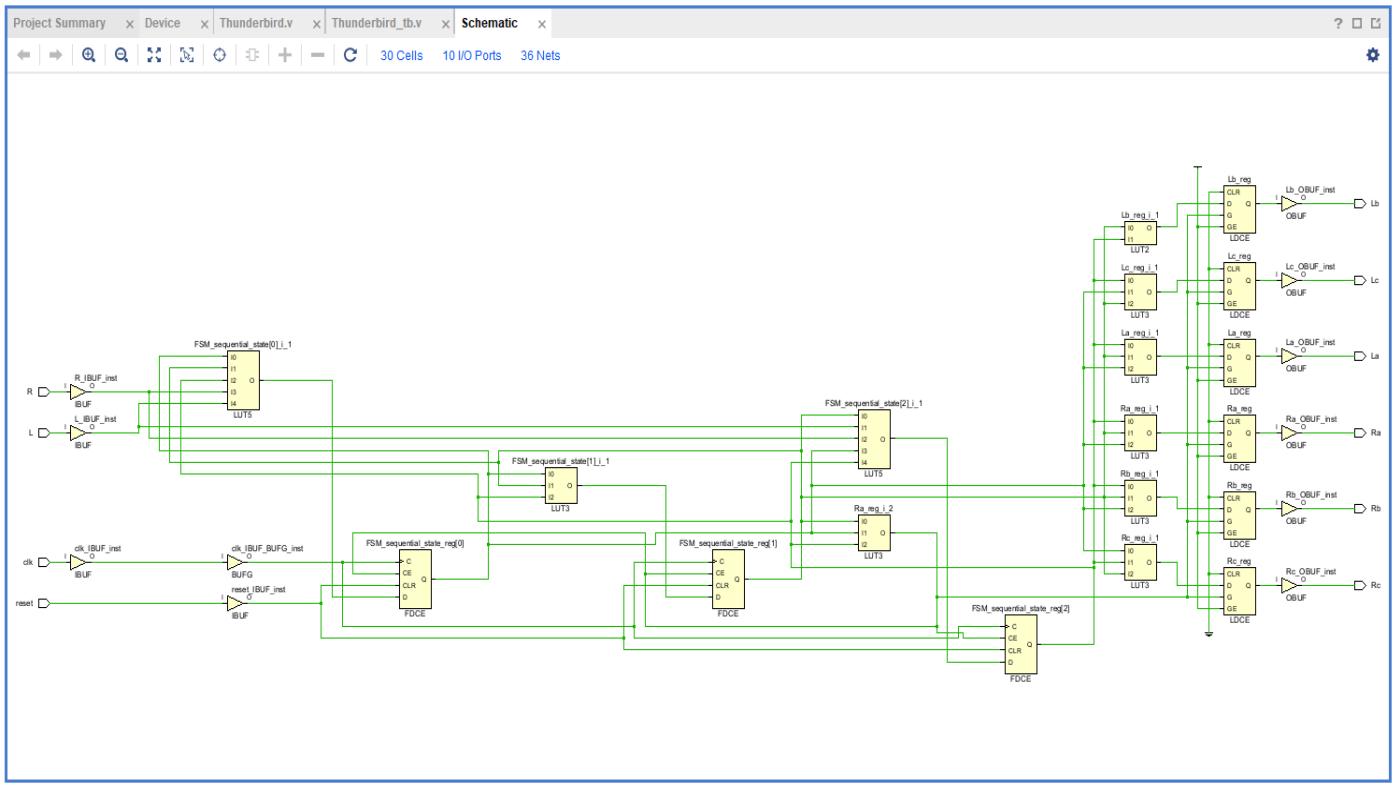
State $S_2 S_1 S_0$	Output LC	LB	LA	RA	RB	RC
0 0 0	0	0	0	0	0	0
0 0 1	0	0	1	0	0	0
0 1 0	0	1	1	0	0	0
0 1 1	1	1	1	0	0	0
1 0 0	0	0	0	1	0	0
1 0 1	0	0	0	1	1	0
1 1 0	0	0	0	1	1	1

$$L_C = \overline{S_2}S_1S_0$$
$$L_B = \overline{S_2}S_1\overline{S_0} + \overline{S_2}S_1S_0$$
$$L_A = \overline{S_2}\,\overline{S_1}S_0 + \overline{S_2}S_1\overline{S_0} + \overline{S_2}S_1S_0$$

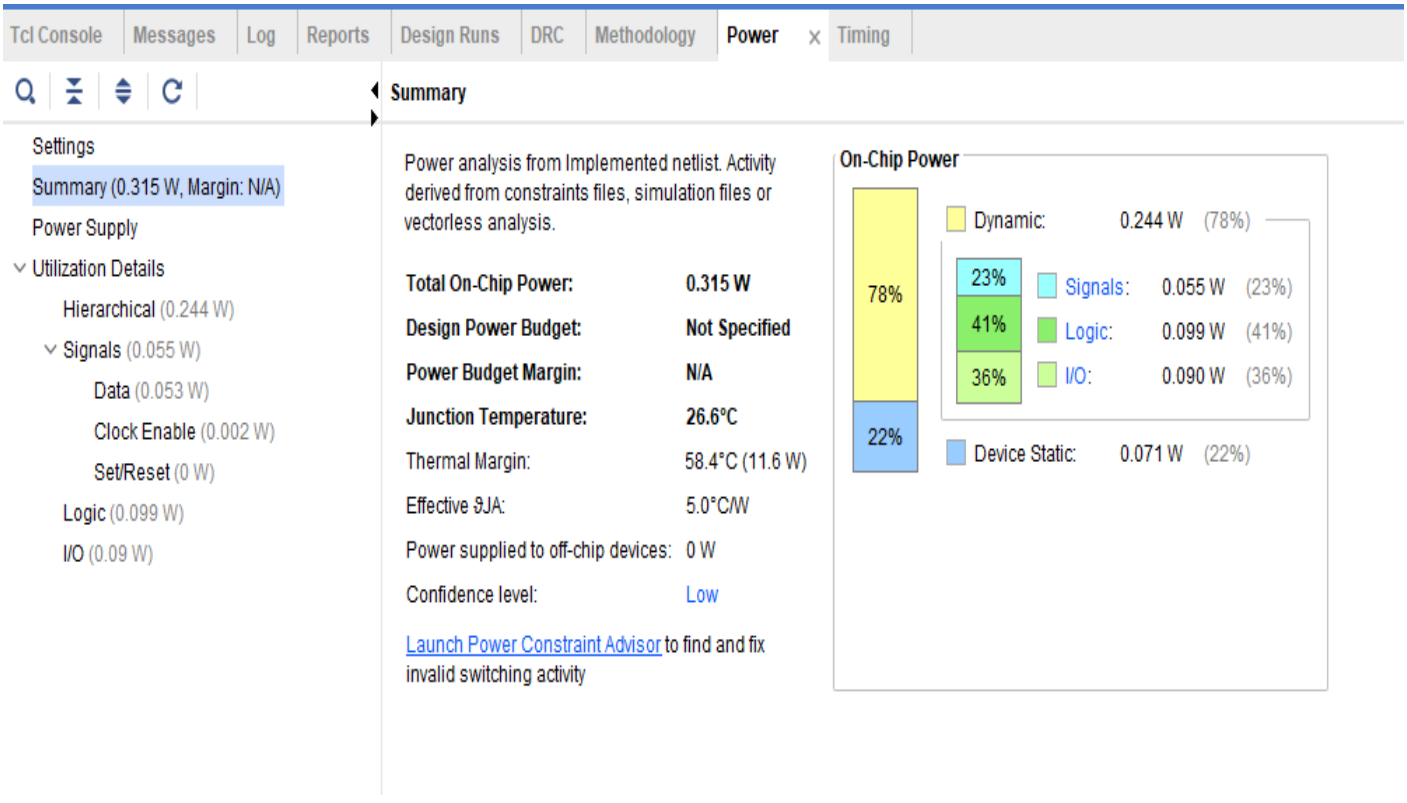
$$R_A = S_2\overline{S_1}\,\overline{S_0} + S_2\overline{S_1}S_0 + S_2S_1\overline{S_0}$$
$$R_B = S_2\overline{S_1}S_0 + S_2S_1\overline{S_0}$$
$$R_C = S_2S_1\overline{S_0}$$

Desing, Code, Verilog Simulation and Synthesis

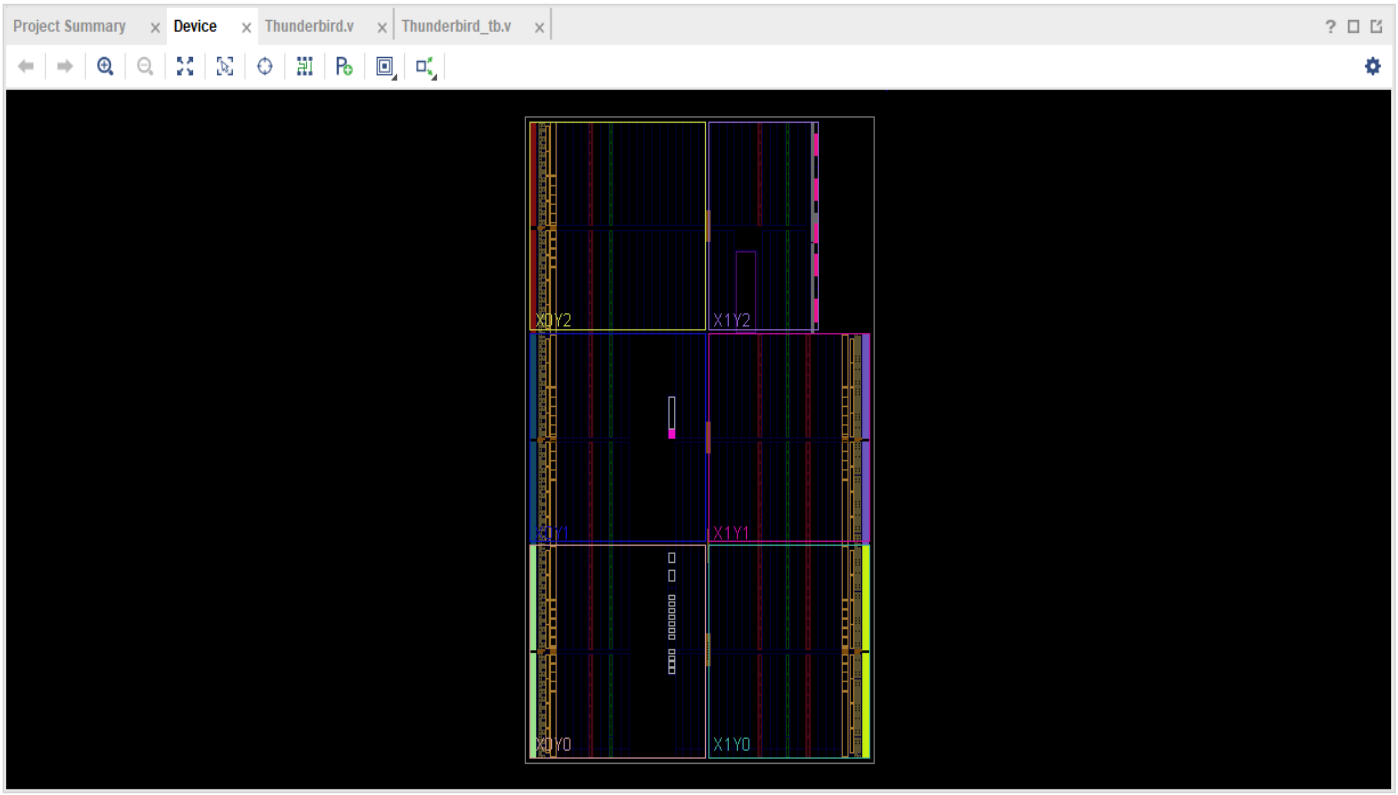
Schematic



Power Report



Device



Code

```
module Thunderbird(
    input clk, reset,
    input wire L,R,
    output reg Ra,Rb,Rc,La,Lb,Lc );


    wire clk_en;
    clk_divider dv(.clk(clk), .reset(reset), .clk_en(clk_en));

    reg [2:0] state, nexstate;
    parameter S0 = 3'b000;
    parameter S1 = 3'b001;
    parameter S2 = 3'b010;
    parameter S3 = 3'b011;
    parameter S4 = 3'b100;
    parameter S5 = 3'b101;
    parameter S6 = 3'b110;

    always @(posedge clk_en, posedge reset) begin


        if (reset)
            state <= S0;
```

```
    else
        state <= nexstate;
end

always @(L,R, state)
    begin
    nexstate = state;
    case (state )
        S0 : begin
            if (L == R )
                nexstate  = S0;
            else
            if (L)
                nexstate  = S1;
            else
                nexstate = S4;
        end
        S1 : nexstate =S2;
        S2 : nexstate =S3;
        S3 : nexstate =S0;
        S4 : nexstate =S5;
        S5 : nexstate =S6;
        S6 : nexstate =S0;
    endcase
end

always @(L,R, state) begin
    case (state)
        S0 : begin
            Ra=0;Rb=0;Rc=0;La=0;Lb=0;Lc=0;
        end
        S1 : begin
            Ra=0;Rb=0;Rc=0;La=1;Lb=0;Lc=0;
        end
        S2 : begin
            Ra=0;Rb=0;Rc=0;La=1;Lb=1;Lc=0;
        end
        S3 : begin
            Ra=0;Rb=0;Rc=0;La=1;Lb=1;Lc=1;
        end
        S4 : begin
            Ra=1;Rb=0;Rc=0;La=0;Lb=0;Lc=0;
        end
        S5 : begin
            Ra=1;Rb=1;Rc=0;La=0;Lb=0;Lc=0;
        end
        S6 : begin
            Ra=1;Rb=1;Rc=1;La=0;Lb=0;Lc=0;
        end
```

```
        endcase
    end
endmodule



`timescale 1ns / 1ps

module Thunderbird(
    input clk, reset,
    input wire L,R,
    output reg Ra,Rb,Rc,La,Lb,Lc );

    reg [2:0] state, nexstate;
    parameter S0 = 3'b000;
    parameter S1 = 3'b001;
    parameter S2 = 3'b010;
    parameter S3 = 3'b011;
    parameter S4 = 3'b100;
    parameter S5 = 3'b101;
    parameter S6 = 3'b110;

    always @(posedge clk, posedge reset) begin
        if (reset)
            state <= S0;
        else
            state <= nexstate;
    end

    always @(L,R, state)
        begin
        nexstate = state;
        case (state )
            S0 : begin
                if (L == R )
                    nexstate  = S0;
                else
                if (L)
                    nexstate  = S1;
                else
                    nexstate = S4;
            end
            S1 : nexstate =S2;
            S2 : nexstate =S3;
            S3 : nexstate =S0;
            S4 : nexstate =S5;
            S5 : nexstate =S6;
            S6 : nexstate =S0;
        endcase
    end
```

```
    always @(L,R, state) begin
        case (state)
            S0 : begin
                Ra=0;Rb=0;Rc=0;La=0;Lb=0;Lc=0;
            end
            S1 : begin
                Ra=0;Rb=0;Rc=0;La=1;Lb=0;Lc=0;
            end
            S2 : begin
                Ra=0;Rb=0;Rc=0;La=1;Lb=1;Lc=0;
            end
            S3 : begin
                Ra=0;Rb=0;Rc=0;La=1;Lb=1;Lc=1;
            end
            S4 : begin
                Ra=1;Rb=0;Rc=0;La=0;Lb=0;Lc=0;
            end
            S5 : begin
                Ra=1;Rb=1;Rc=0;La=0;Lb=0;Lc=0;
            end
            S6 : begin
                Ra=1;Rb=1;Rc=1;La=0;Lb=0;Lc=0;
            end
        endcase
    end
endmodule

`timescale 1ns / 1ps

module Thunderbird_tb;
    reg clk, reset;
    reg L;
    reg R;
    wire Ra,Rb,Rc,La,Lb,Lc;

Thunderbird Thunderbird_tb(clk,reset,L,R,Ra,Rb,Rc,La,Lb,Lc);

always begin
clk = 1'b1;
#10; clk = 1'b0;
#10;
end

initial begin
    L = "0"; R = "0";reset = "1";
    #30; reset = "0";
    #40; L = "1"; R = "0";
    #40; L = "0"; R = "1";
```

```
    #40; L = "1"; R = "1";

    #40; L = "0"; R = "1";
    #40; L = "1"; R = "0";
    #40; L = "1"; R = "1";
    end
endmodule
```

Contrasts

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports)
##according to the top level signal names in the project

## Clock signal
set_property -dict { PACKAGE_PIN W5 IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock-add-name sys_clk_pin-period 10.00-waveform{0 5}[get_ports clk]

## LEDs
set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports {La}]
set_property -dict { PACKAGE_PIN E19 IOSTANDARD LVCMOS33 } [get_ports {Lb}]
set_property -dict { PACKAGE_PIN U19 IOSTANDARD LVCMOS33 } [get_ports {Lc}]
set_property -dict { PACKAGE_PIN V19 IOSTANDARD LVCMOS33 } [get_ports {Ra}]
set_property -dict { PACKAGE_PIN W18 IOSTANDARD LVCMOS33 } [get_ports {Rb}]
set_property -dict { PACKAGE_PIN U15 IOSTANDARD LVCMOS33 } [get_ports {Rc}]

##Buttons
set_property -dict{PACKAGE_PIN U18 IOSTANDARD LVCMOS33 } [get_ports reset]
#set_property -dict{PACKAGE_PIN T18 IOSTANDARD LVCMOS33 } [get_ports btnU]
set_property -dict { PACKAGE_PIN W19 IOSTANDARD LVCMOS33 } [get_ports L]
set_property -dict { PACKAGE_PIN T17 IOSTANDARD LVCMOS33 } [get_ports R]
#set_property -dict{PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports btnD]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be
## used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

Authors: Carlos Sobenes, Angello Soldi and Wanly Obregón