Ho Chi Minh University of Technology

Faculty of Computer Science and Engineering
Course: Logic Design with HDL (CO1025)

Logic Design Lab:

Weekly report - 3

Lecturer: Pham Quoc Cuong
Student: Phan Minh Toan - 1852798
Tran Nguyen Khoi - 1952797
Hua Vu Minh Hieu - 2052990
Nguyen Quang Khoi - 2153485

Ho Chi Minh, May 2022

Contents

List of Figures

Listings

1 Exercise 1

Clock Frequency Divider implementation:

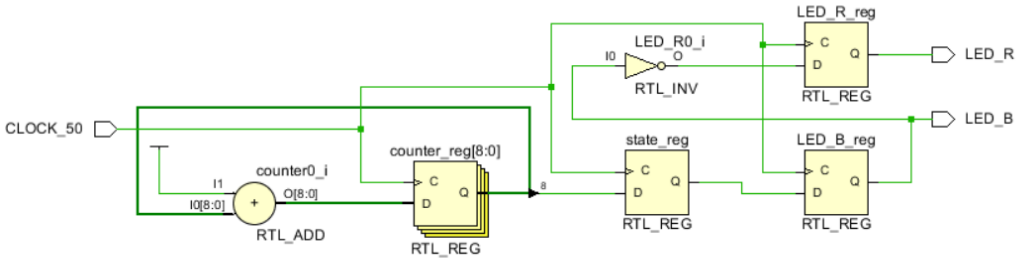

Figure 1: Clock Frequency Divider RTL Design

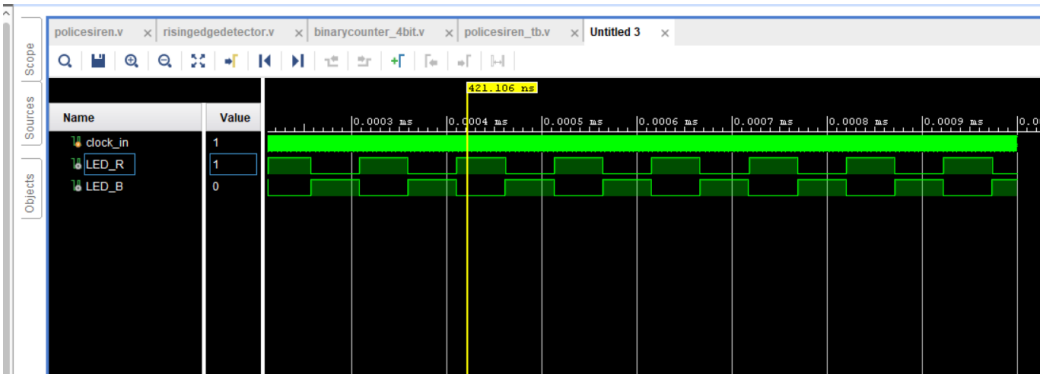

Figure 2: Clock Frequency Divider waveform - 1

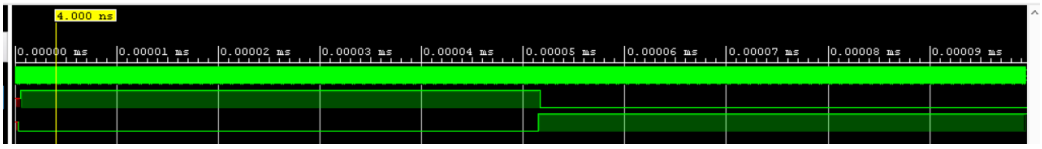

Figure 3: Clock Frequency Divider waveform - 2

```
module policesiren (
    input CLOCK_50,
    output reg LED_B,
    output reg LED_R
```

```
);

    /* reg */
    reg [8:0] counter;
    reg state;
    initial
    counter = 4'b0000;

    always @ (posedge CLOCK_50) begin
        LED_R <= ~LED_B;
        LED_B <= state;
        counter <= counter + 1;
        state <= counter[8];
         // <------- data to change
    end

endmodule
```

Listing 1: Clock Frequency Divider Implemetation

```
`timescale 10ps / 1ps
//
    //////////////////////////////////////////////////////////////////////////////////

// Company:
// Engineer:
//
// Create Date: 04/28/2022 01:19:35 PM
// Design Name:
// Module Name: clock_divider_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////


module policesiren_tb(

    );
```

```
26 reg clock_in;
27 wire LED_R;
28 wire LED_B;
29
30 policesiren uut(.CLOCK_50(clock_in), .LED_B(LED_B), .LED_R(LED_R
       ));
31
32 initial begin
33     clock_in = 0;
34     forever #10 clock_in = ~clock_in;
35 end
36 endmodule
```

Listing 2: Clock Frequency Divider Testbench

2 Exercise 2.1

Rising Edge Detection Circuit implementation:

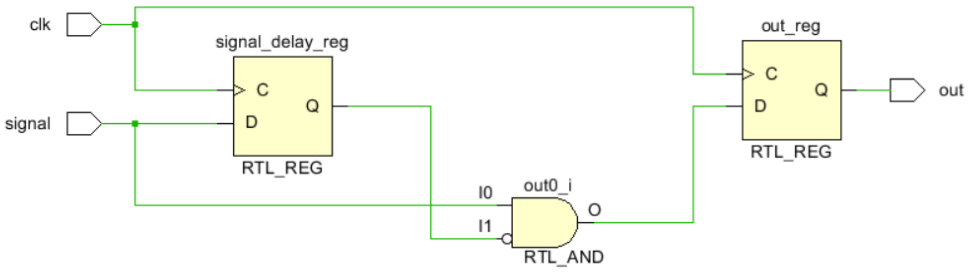

Figure 4: Rising Edge Detection Circuit RTL Design

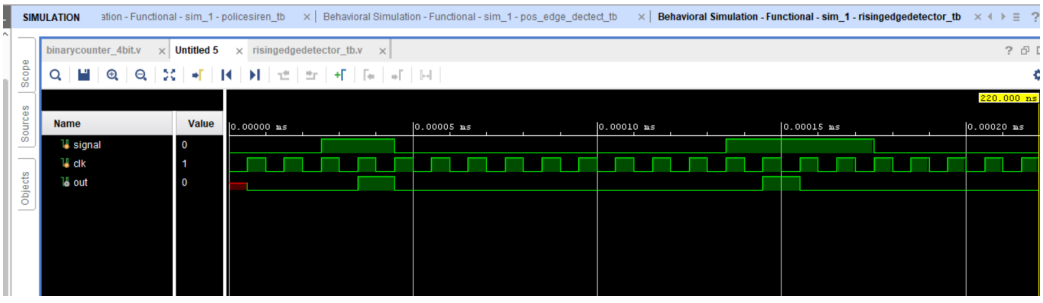

Figure 5: Rising Edge Detection Circuit Waveform

```
module risingedgedetector (
    input signal,
    input clk,
    output reg out
);
    reg signal_delay;
    always @(posedge clk) begin
        signal_delay <= signal;
        out = signal & ~signal_delay;
    end
endmodule
```

Listing 3: Rising Edge Detection Circuit Implemetation

```
`timescale 1ns / 1ps
module risingedgedetector_tb();

    reg signal;
    reg clk;
    wire out;

    risingedgedetector UUT(.signal(signal), .clk(clk), .out(out)
    );

    always #5 clk = ~clk;

    initial begin
        clk <= 0;
        signal <=0;
        #10 signal <= 0;
        #15 signal <= 1;
        #20 signal <= 0;
        #25 signal <= 0;
        #30 signal <= 0;
        #35 signal <= 1;
        #40 signal <= 0;
        #45 signal <= 0;
        $stop;
    end
endmodule
```

Listing 4: Rising Edge Detection Circuit Testbench

3 Exercise 2.2

4 bit binary counter implementation:

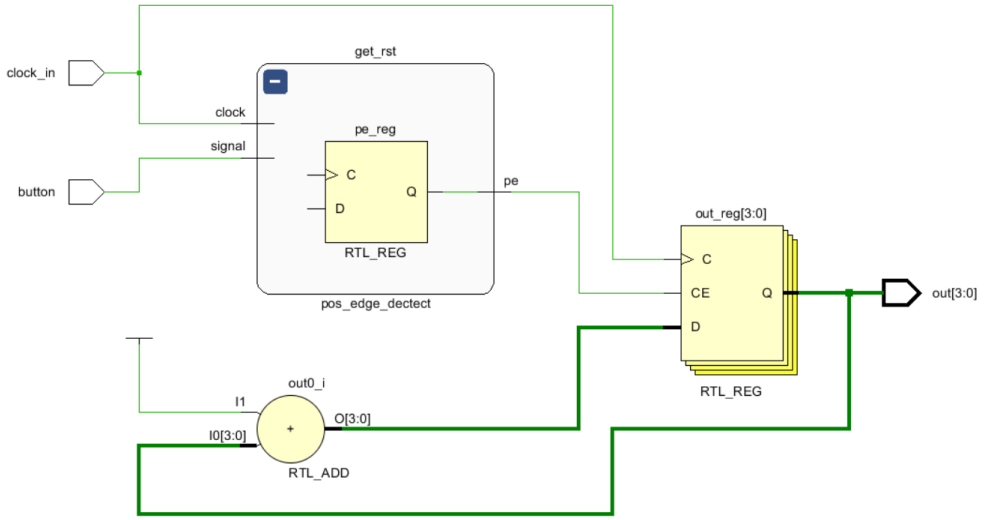

Figure 6: 4 bit binary counter RTL Design

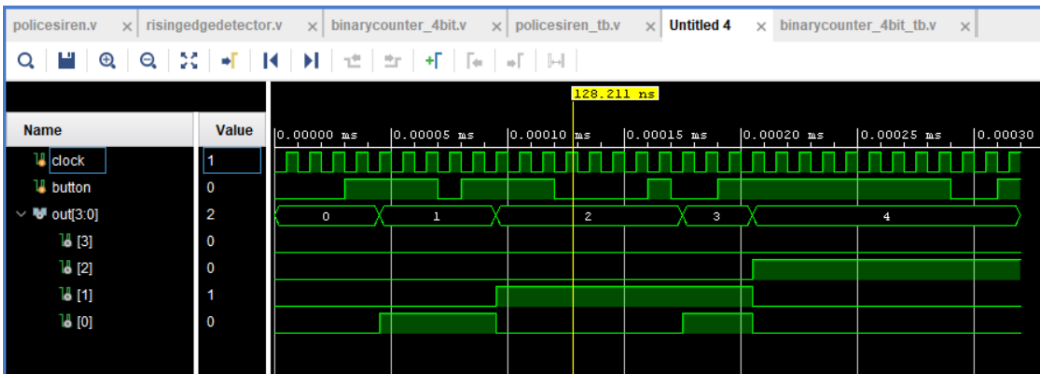

Figure 7: 4 bit binary counter Waveform

```
module pos_edge_dectect(
    input signal,
    input clock,
    output reg pe
    );

    reg sig_delay;

    always @(posedge clock) begin
        sig_delay <= signal;
        pe = signal & ~sig_delay;
    end
endmodule
```

```

module counter(
    input clock_in,
    input button,
    output reg [3:0] out
    );

    pos_edge_dectect get_rst(button, clock_in, clock_out);
    initial begin out = 4'b0000; end
    always @(posedge clock_in) begin
        if (clock_out == 1) out = out + 1;
    end
endmodule
```

Listing 5: 4 bit binary counter Implemetation

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/28/2022 02:22:56 PM
// Design Name:
// Module Name: pos_edge_dectect_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module pos_edge_dectect_tb(

    );
    reg clock;
    reg button;
    wire [3:0] out;

```

```
30
31      counter ped0(.clock_in(clock), .button(button), .out(out));
32
33      always #5 clock = ~clock;
34
35      initial begin
36      clock <= 0;
37      button <= 0;
38      #20;
39      #10 button <= 1;
40      #40 button <= 0;
41      #10 button <= 1;
42      #40 button <= 0;
43      #20
44      #20 button <= 1;
45      #10 button <= 0;
46      #20 button <= 1;
47      #100 button <= 0;
48      #20 button <= 1;
49      #10 button <= 0;
50      $stop;
51      end
52  endmodule
```

Listing 6: 4 bit binary counter Testbench