ECE 451-LAB6

Fall 2019

Ram Rohit Gannavarapu

Contents

List of Figures

1. List of items completed

a) Design of Pulse counter on verilog
b) Design of Pyramid Counter on verilog

2. Lab Procedures

Truth table of Pulse counter based on the waveform provided in the question:

Enable	Clear	Clk	Clk2	Clk4	P1	P2	P3	P4	P23
0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	0	1	1	1	0	0	0	0
1	0	1	1	1	0	1	0	0	1
1	0	0	0	1	0	0	1	0	1
1	0	1	0	1	0	0	0	1	0
1	0	0	1	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	1
1	0	0	0	0	0	0	1	0	1
1	0	1	0	0	0	0	0	1	0
1	0	0	1	1	1	0	0	0	0
1	0	1	1	1	0	1	0	0	1
1	0	0	0	1	0	0	1	0	1
1	0	1	0	1	0	0	0	1	0

The above truth table was used to understand the system behavior and design the Pulse Counter on Verilog.

3. Schematics/Verilog models

```verilog
module Pulse_Clock(clock, clk2,clk4,p1,p2,p3,p4,p23,enable,clear);

input clock,enable,clear;
output reg clk2,clk4;
output reg p1, p2, p3, p4, p23;
reg flag=0;

always @(negedge clock)
    begin
    case({enable,clear})
        2'b10:begin clk2=~clk2; flag=1; end
        default: clk2 = 0;
    endcase
    end

always @(posedge clk2)
    begin
    case({enable,clear})
        2'b10:clk4=~clk4;
        default: clk4 = 0;
        endcase
    end

```

```verilog
always @(enable,clear,clock,clk2,clk4)
    begin
        case({flag,enable,clear,clock,clk2})
            5'b11000: begin p1=0;p2=0;p3=1;p4=0;p23=1;  end
            5'b11001: begin p1=1;p2=0;p3=0;p4=0;p23=0;  end
            5'b11010: begin p1=0;p2=0;p3=0;p4=1;p23=0;  end
            5'b11011: begin p1=0;p2=1;p3=0;p4=0;p23=1;  end
            default: begin p1=0;p2=0;p3=0;p4=0;p23=0;  end

        endcase
    end

  endmodule
```

Code 1: Pulse Counter Verilog Code

```verilog
module Pyramid_Counter(clock,enable,clear,pulse1,pulse2,count);


input clock, enable, clear;
output reg [3:0] count;
reg [3:0] max = 4'b1111;
output reg pulse1;
output reg pulse2;
wire max_reached;
wire max_equal_0;

always @(posedge clock) begin
  if(clear) begin
    count <= 4'b0000;
    max   <= 4'b1111;
  end
  else if (enable) begin
    if(pulse2) pulse2 <= 'b0;
    else if (max_equal_0) pulse2 <= 1'b1;

    if(pulse1) pulse1 <= 'b0;
    else if(max_reached) begin
      count <= 4'b0000;
      if(max != 4'b0001) max <= max - 'b1;
      else max <= 4'b1111;
      pulse1 <= 'b1;
    end
    else count <= count + 'b1;
  end
end

assign max_reached = (count == max) ? 'b1 : 'b0;
assign max_equal_0 = ((count == max) && (max == 4'b0001)) ? 'b1 : 'b0;


endmodule
```

Code 2: Pyramid counter Verilog Code

4. Simulation Results

Figure 3: Simulation Results of Pulse Counter

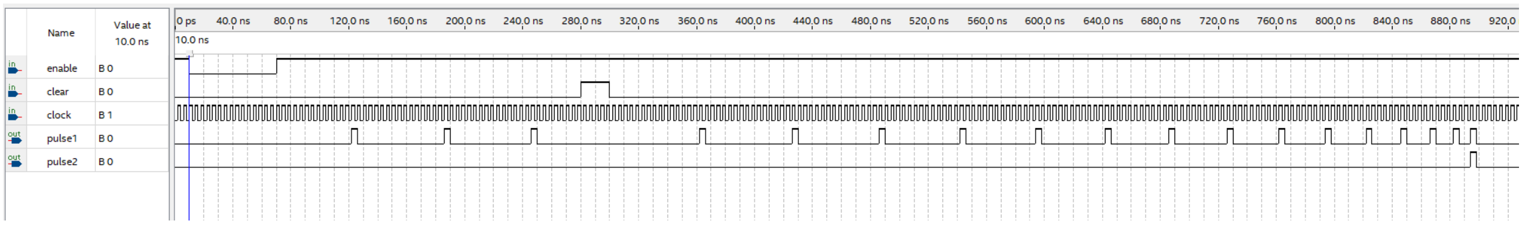

Figure 4: Simulation Results of Pyramid Counter

5. Analysis and Explanations

The Pulse Counter was designed in Verilog based on the truth table. Clk2 and Clk4 signals were generated with periods, 2 times and 4 times of the input clock signal. P1 p2 p3 p4 follow the following pattern:

Clk	Clk2	P1	P2	P3	P4	P23
0	0	0	0	1	0	1
0	1	1	0	0	0	0
1	0	0	0	0	1	0
1	1	0	1	0	0	1

Pyramid Counter on the other hand was designed to output 2 signals:

1) Pulse 1: This signal generates a pulse whenever a counter reaches the max value. This max value keeps reducing when the counter reaches its max value.
2) Pulse2: This Signal generates a pulse whenever the max value of the counter in the pulse1 reaches zero.

6. Conclusion

In this Lab we have successfully designed a pulse Counter and Pyramid Counter in Verilog and verified those using Waveforms and also by implementing them on the Altera Board.

7. Answers to questions posted in the lab

Q) Describe how you would implement a hierarchical design in Verilog?

By creating small functional modules we can re-use them as part of a bigger functional module. A good example would be the inbuilt AND and OR gates in Verilog. These blocks form the basic building blocks of many complex models. Similarly, any design can be broken down to small functional units which can be then modeled in Verilog using a separate module and these modules can be reused.