ECE 451-LAB6

Fall 2019

Ram Rohit Gannavarapu

Contents

[List of Figures 1](#_Toc24407196)

[1. List of items completed 1](#_Toc24407197)

[2. Lab Procedures 2](#_Toc24407198)

[3. Schematics/Verilog models 2](#_Toc24407199)

[4. Simulation Results 4](#_Toc24407200)

[5. Analysis and Explanations 4](#_Toc24407201)

[6. Conclusion 5](#_Toc24407202)

[7. Answers to questions posted in the lab 5](#_Toc24407203)

# List of Figures

[Code 1: Pulse Counter Verilog Code 3](#_Toc24407213)

[Code 2: Pyramid counter Verilog Code 3](#_Toc24407214)

[Figure 3: Simulation Results of Pulse Counter 4](#_Toc24407215)

[Figure 4: Simulation Results of Pyramid Counter 4](#_Toc24407216)

# List of items completed

* 1. Design of Pulse counter on verilog
  2. Design of Pyramid Counter on verilog

# 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.

# Schematics/Verilog models

1. module Pulse\_Clock(clock, clk2,clk4,p1,p2,p3,p4,p23,enable,clear);
3. input clock,enable,clear;
4. output reg clk2,clk4;
5. output reg p1, p2, p3, p4, p23;
6. reg flag=0;
8. always @(negedge clock)
9. begin
10. **case**({enable,clear})
11. 2'b10:begin clk2=~clk2; flag=1; end
12. **default**: clk2 = 0;
13. endcase
14. end
16. always @(posedge clk2)
17. begin
18. **case**({enable,clear})
19. 2'b10:clk4=~clk4;
20. **default**: clk4 = 0;
21. endcase
22. end
24. always @(enable,clear,clock,clk2,clk4)
25. begin
26. **case**({flag,enable,clear,clock,clk2})
27. 5'b11000: begin p1=0;p2=0;p3=1;p4=0;p23=1;  end
28. 5'b11001: begin p1=1;p2=0;p3=0;p4=0;p23=0;  end
29. 5'b11010: begin p1=0;p2=0;p3=0;p4=1;p23=0;  end
30. 5'b11011: begin p1=0;p2=1;p3=0;p4=0;p23=1;  end
31. **default**: begin p1=0;p2=0;p3=0;p4=0;p23=0;  end
33. endcase
34. end
36. endmodule

Code 1: Pulse Counter Verilog Code

1. module Pyramid\_Counter(clock,enable,clear,pulse1,pulse2,count);

4. input clock, enable, clear;
5. output reg [3:0] count;
6. reg [3:0] max = 4'b1111;
7. output reg pulse1;
8. output reg pulse2;
9. wire max\_reached;
10. wire max\_equal\_0;
12. always @(posedge clock) begin
13. **if**(clear) begin
14. count <= 4'b0000;
15. max   <= 4'b1111;
16. end
17. **else** **if** (enable) begin
18. **if**(pulse2) pulse2 <= 'b0;
19. **else** **if** (max\_equal\_0) pulse2 <= 1'b1;
21. **if**(pulse1) pulse1 <= 'b0;
22. **else** **if**(max\_reached) begin
23. count <= 4'b0000;
24. **if**(max != 4'b0001) max <= max - 'b1;
25. **else** max <= 4'b1111;
26. pulse1 <= 'b1;
27. end
28. **else** count <= count + 'b1;
29. end
30. end
32. assign max\_reached = (count == max) ? 'b1 : 'b0;
33. assign max\_equal\_0 = ((count == max) && (max == 4'b0001)) ? 'b1 : 'b0;

36. endmodule

Code 2: Pyramid counter Verilog Code

# Simulation Results

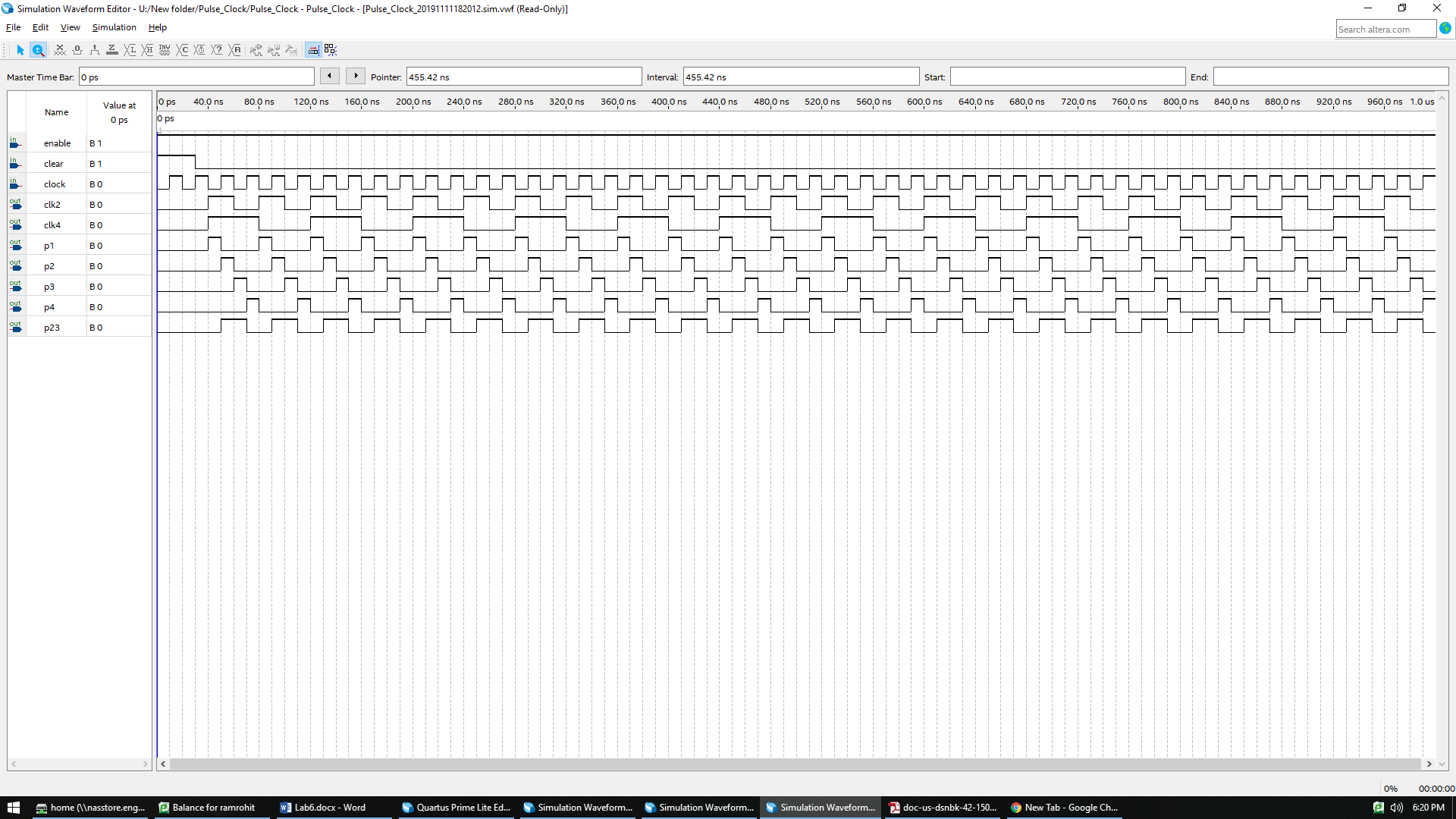


Figure 3: Simulation Results of Pulse Counter

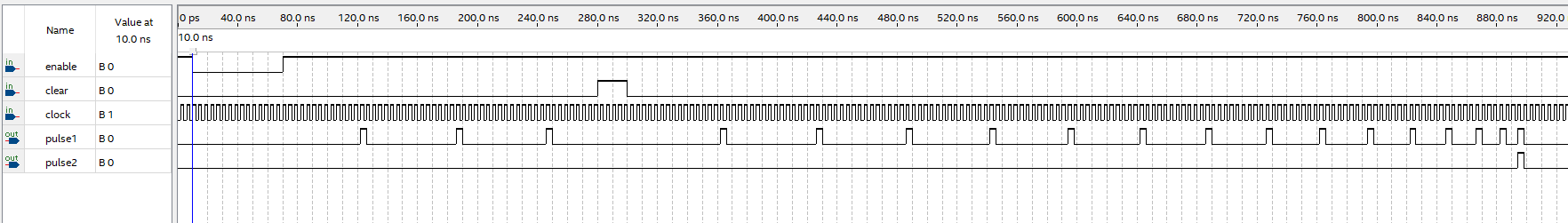


Figure 4: Simulation Results of Pyramid Counter

# 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.

# 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.

# 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.