Bryan Alexander
Kendall Won

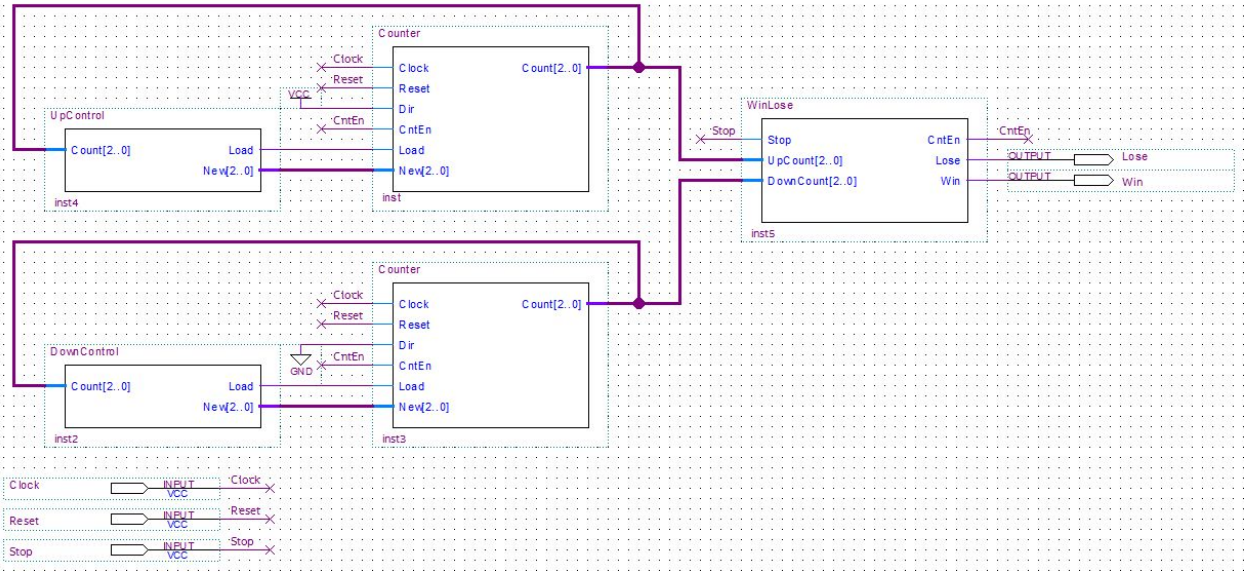
Introduction

In this week's lab we created a circuit that uses counters, one up counter and one down counter. The up counter goes from 1 to 5 and the down counter goes from 5 to 1. When the up counter reaches 5, the counter is reset back to 1 on the next count. Vice versa for the down counter. The circuit stops when the up counter and down counter have the same value. The stop and win signal is changed to 1.

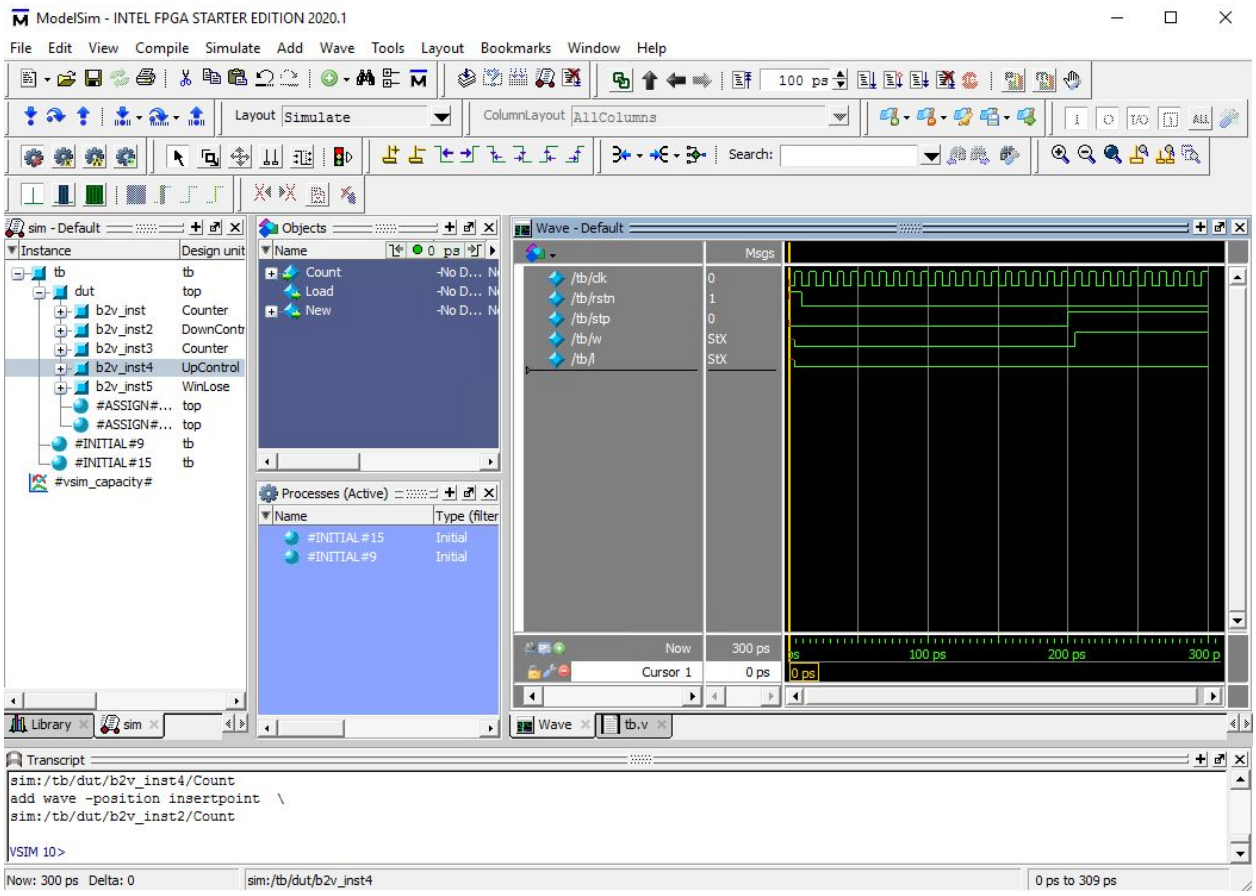
If you were to change your design to have the counters count from 2 to 6 (or 6 to 2) instead of only 1 to 5, list all the things you would need to change.

To change the counter's counter range from 1 to 5 into becoming 2 to 6, we would need to change the values in the UpControl and DownControl modules inside Lab_7.v to reflect this new behavior. In the UpControl module, we would need to now load the value 2 into the counter when the value reaches 6. Similarly in the DownControl module, we would now need to load the value 6 into the counter when the current value reaches 2. After doing this, we would need to recompile the design files before redoing the simulation.

Circuit Diagram/Schematic



ModelSim Simulation/Waveform



top.v Verilog File

```
// Copyright (C) 2020  Intel Corporation. All rights reserved.
// Your use of Intel Corporation's design tools, logic functions
// and other software and tools, and any partner logic
// functions, and any output files from any of the foregoing
// (including device programming or simulation files), and any
// associated documentation or information are expressly subject
// to the terms and conditions of the Intel Program License
// Subscription Agreement, the Intel Quartus Prime License Agreement,
// the Intel FPGA IP License Agreement, or other applicable license
// agreement, including, without limitation, that your use is for
// the sole purpose of programming logic devices manufactured by
// Intel and sold by Intel or its authorized distributors.  Please
// refer to the applicable agreement for further details, at
// https://fpgasoftware.intel.com/eula.

// PROGRAM     "Quartus Prime"
// VERSION     "Version 20.1.0 Build 711 06/05/2020 SJ Lite Edition"
// CREATED     "Tue Feb 23 17:37:40 2021"

module top(
    Reset,
    Stop,
    Clock,
    Lose,
    Win
);


  input wire  Reset;
  input wire  Stop;
  input wire  Clock;
  output wire Lose;
  output wire Win;

  wire  CntEn;
  wire  SYNTHESIZED_WIRE_0;
  wire  SYNTHESIZED_WIRE_1;
  wire  [2:0] SYNTHESIZED_WIRE_2;
  wire  [2:0] SYNTHESIZED_WIRE_10;
  wire  SYNTHESIZED_WIRE_4;
  wire  SYNTHESIZED_WIRE_5;
  wire  [2:0] SYNTHESIZED_WIRE_6;
  wire  [2:0] SYNTHESIZED_WIRE_11;

  assign   SYNTHESIZED_WIRE_0 = 1;
  assign   SYNTHESIZED_WIRE_4 = 0;



Counter  b2v_inst(
    .Clock(Clock),
    .Reset(Reset),
    .Dir(SYNTHESIZED_WIRE_0),
    .CntEn(CntEn),
    .Load(SYNTHESIZED_WIRE_1),
    .New(SYNTHESIZED_WIRE_2),
    .Count(SYNTHESIZED_WIRE_11));

DownControl b2v_inst2(
    .Count(SYNTHESIZED_WIRE_10),
    .Load(SYNTHESIZED_WIRE_5),
    .New(SYNTHESIZED_WIRE_6));

Counter  b2v_inst3(
    .Clock(Clock),
    .Reset(Reset),
    .Dir(SYNTHESIZED_WIRE_4),
    .CntEn(CntEn),
    .Load(SYNTHESIZED_WIRE_5),
    .New(SYNTHESIZED_WIRE_6),
    .Count(SYNTHESIZED_WIRE_10));

UpControl   b2v_inst4(
    .Count(SYNTHESIZED_WIRE_11),
    .Load(SYNTHESIZED_WIRE_1),
    .New(SYNTHESIZED_WIRE_2));

```

tb.v Verilog File

```
1   module tb();
2
3     reg clk,rstn,stp;
4     wire w,l;
5
6   top dut(.Clock(clk), .Reset(rstn), .Stop(stp), .Win(w), .Lose(l));
7
8
9     initial begin
10      clk = 0;
11
12    forever #5 clk = ~clk;
13    end
14
15  initial begin
16
17  rstn = 1;
18
19  stp = 0;
20  #10;
21
22  rstn = 0;
23
24
25  #100;
26
27
28
29  #90;
30
31  stp = 1;
32
33  #100
34  $finish;
35  end
36  endmodule
37
```

Lab_7.v Verilog File

```
1   module UpControl(
2     input[2:0] Count,
3     output Load,
4     output[2:0] New);
5
6     assign Load=(Count==3'b101)?1'b1:1'b0;
7     assign New=(Count==3'b101)?3'b001:1'b001;
8   endmodule
9
10  module DownControl(
11    input[2:0] Count,
12    output Load,
13    output[2:0] New);
14
15    assign Load=(Count==3'b001)?1'b1:1'b0;
16    assign New=(Count==3'b001)?3'b101:1'b101;
17  endmodule
18
19   module Counter(
20    input Clock,
21    input Reset,
22    input Dir,
23    input CntEn,
24    input Load,
25    input[2:0] New,
26    output reg[2:0] Count);
27
28
29   always @(posedge Clock)
30     begin
31      if(Reset == 1)
32         begin
33         Count <= 3'b001;
34         end
35       else
36         begin
37         if(CntEn == 1)
38             begin
39             if(Load == 1)
40               begin
41                 Count <= New;
42               end
43             else
44               begin
45                if (Dir == 1)
46                  begin
47                    Count <= Count + 1;
48                  end
49                else
50                  begin
51                    Count <= Count - 1;
52                  end
53              end
54           end
55       end
56    end
57
58   endmodule
59
60  module WinLose(
61    input Stop,
62    input[2:0] UpCount,
63    input[2:0] DownCount,
64    output reg CntEn,
65    output reg Lose,
66    output reg Win);
67
68
69
70    always@(UpCount or DownCount)
71      begin
72        Win<=1'b0;
73        Lose<=1'b0;
74        if(Stop)
75          begin
76            CntEn<=1'b0;
77            if(UpCount==DownCount)
78              Win<=1'b1;
79            else
80              Lose<=1'b1;
81          end
82        else
83          begin
84            CntEn<=1'b1;
85          end
86      end
87
88  endmodule
```