Tyler Youk HW #10

//up\_down\_counter source code

module up\_down\_counter #(

parameter N = 4

) (

input logic clk,

input logic en\_b,

input logic load\_b,

input logic up,

input logic [N-1:0] load\_in,

output logic [N-1:0] q,

output logic rco\_b

);

always\_ff @(posedge clk) begin

if (~en\_b) begin

if (load\_b) begin

//using non-blocking assignment to assign q value

q <= up ? q+1 : q-1;

else

q <= load\_in;

end

else begin

q <= load\_b;

end

end

//end

assign rco\_b = ~(up& (&q)) | (~up & ~(|q));

endmodule

//testbench source code

module testbench\_hw7 ();

logic clk, en\_b;

logic [3:0] load\_in4, q4;

logic [4:0] load\_in5, q5;

logic rco\_b4, rco\_b5;

up\_down\_counter #(

.N(4)

) counter4 (

.clk (clk),

.en\_b (en\_b),

.load\_b (load4\_b),

.up (up),

.load\_in(load\_in4),

.q (q4),

.rco\_b (rco4\_b)

);

up\_down\_counter #(

.N(5)

) counter5 (

.clk(clk),

.en\_b(en\_b),

.up(up),

.load\_b(load5\_b),

.load\_in(load\_in5),

.q(q5),

.rco\_b(rco\_b5)

.rco

);

initial begin

clk = 1'b0;

en\_b = 1'b1;

load4\_b = 1'b0;

load5\_b = 1'b0;

up = 1'b1;

load\_in4 = 4'b0000;

load\_in5 = 5'b00000;

forever #5 clk = ~clk;

end

initial begin

#10;

load4\_b = 1'b0;

load5\_b = 1'b0;

#10;

en\_b = 1'b0;

#10;

load4\_b = 1'b1;

load5\_b = 1'b1;

#320;

en\_b = 1'b1;

up = 1'b0;

load4\_b = 1'b0;

load5\_b = 1'b0;

load\_in4 = 4'b1111;

load\_in5 = 5'b11111;

#10;

en\_b = 1'b0;

#10;

load4\_b = 1'b1;

load5\_b = 1'b1;

en\_b = 1'b1;

#10;

en\_b = 1'b0;

#320;

load4\_b = 1'b0;

load5\_b = 1'b0;

load\_in4 = 4'b1010;

load\_in5 = 5'b01010;

#10;

load4\_b = 1'b1;

load5\_b = 1'b1;

#10;

up = 1'b1;

load4\_b = 1'b0;

load5\_b = 1'b0;

load\_in4 = 4'b0101;

load\_in5 = 5'b10101;

#10;

load4\_b = 1'b1;

load5\_b = 1'b1;

#10;

$finish();

end

endmodule

Waveform Screenshot

