**Lab 5**

Ashmitha Murali

Prof. Alex Yang

Verilog HDL Lab EE461L

28th October 2025

Q1.

Waveform:  
A black rectangular object with green and blue lines

AI-generated content may be incorrect.

Result:

A screenshot of a computer code

AI-generated content may be incorrect.

**Code:**

module counter\_with\_overflow (

input wire clk,

input wire start,

input wire [3:0] NumberIn,

output reg [7:0] Count,

output reg Overflow

);

wire [8:0] sum;

wire cout;

assign sum = {1'b0, Count} + {5'b00000, NumberIn};

assign cout = sum[8];

//first mux and flip flop

always @(posedge clk) begin

if (start) begin

Count <= 8'd0;

end else begin

Count <= sum[7:0];

end

end

//second mux and flip flop

always @(posedge clk) begin

casez ({start, Overflow})

2'b00: Overflow <= cout;

2'b01: Overflow <= 1'b1;

2'b1?: Overflow <= 1'b0;

2'bx?: Overflow <= 1'bx;

endcase

end

endmodule

**Testbench:**

`timescale 1ns/1ns

module tb\_counter;

reg clk;

reg start;

reg [3:0] NumberIn;

wire [7:0] Count;

wire Overflow;

// DUT instance

counter\_with\_overflow dut (

.clk(clk),

.start(start),

.NumberIn(NumberIn),

.Count(Count),

.Overflow(Overflow)

);

initial begin

clk = 0;

forever #5 clk = ~clk;

end

initial begin

$dumpfile("counter.vcd");

$dumpvars(0, tb\_counter);

End

initial begin

$monitor("Time=%0t | clk=%b | start=%b | NumberIn=%0d | Count=%0d | Overflow=%b",

$time, clk, start, NumberIn, Count, Overflow);

start = 1;

#10 NumberIn = 4'd3;

#10 start = 0;

#20 NumberIn = 4'd3;

#20 NumberIn = 4'd5;

#20 NumberIn = 4'd10;

#20 NumberIn = 4'd10;;

#20 NumberIn = 4'd15;

#20 NumberIn = 4'd15;

$finish;

end

endmodule

Q2.

*`timescale 1ns/100ps*

*module xor3 (input B\_i, D\_i, sel\_i, clock, output E\_o);*

*reg E\_o;*

*always@(posedge clock) begin*

*case(sel\_i)*

*0: E\_o <= D\_i + B\_i;*

*1: E\_o <= B\_i;*

*endcase*

*end*

*endmodule*

Since we are using “always @(posedge clock)” this would implement a D flip flop.

The case statement would implement a MUX with ‘E\_o’ as output. This mux will have select input from ‘sel\_i’, and the 0-input would be “*D\_i + B\_i”* and1-inputwould be *“B\_i”.*

The output of the Mux would go into the D pin of the flip flop and the output of D flip flop will be E\_o.

A diagram of a circuit

AI-generated content may be incorrect.

Q3.

Waveform:  
A black rectangular object with green lines

AI-generated content may be incorrect.

Result:

A screenshot of a computer program

AI-generated content may be incorrect.

**Modified Design Code:**

module counter (clock, in, latch, dec, zero);

input clock; /\* clock \*/

input [3:0] in; /\* starting count \*/

input latch; /\* latch `in’ when high \*/

input dec; /\* decrement count when dec high \*/

output zero; /\* high when count down to zero \*/

reg [3:0] value; /\* current count value \*/

always@(posedge clock) begin

if (latch) value <= in;

else if (dec && !zero && (value > 4'd1)) value <= value - 4'd2;

end

assign zero = (value == 4'b0) || (value == 4'd1);

endmodule /\* counter \*/

**Testbench:**

module tb\_counter;

reg clock;

reg [3:0] in;

reg latch;

reg dec;

wire zero;

// Instantiate the DUT

counter dut (

.clock(clock),

.in(in),

.latch(latch),

.dec(dec),

.zero(zero)

);

initial begin

clock = 0;

forever #5 clock = ~clock;

end

initial begin

$dumpfile("counter\_wave.vcd");

$dumpvars(0, tb\_counter);

end

// Stimulus

initial begin

$display("---- Simulation Start ----");

$monitor("T=%0t | latch=%b | dec=%b | in=%b | value=%b | zero=%b",

$time, latch, dec, in, dut.value, zero);

latch = 1; dec = 0; in = 4'b1010; // Load 10

#10 latch = 0; dec = 1;

#100 latch = 1; in = 4'b0111; // Load 7;

#10 latch = 0; dec = 1;

#60;

$finish;

end

endmodule