# Lab Experiment #2

**Realization of Digital Circuits Using Behavioral Level Modeling**

* 1. **Objective:** To realize the design of digital circuits in Verilog using behavioral level modelling then simulating and synthesizing using EDA tools.

## Software tools Requirement

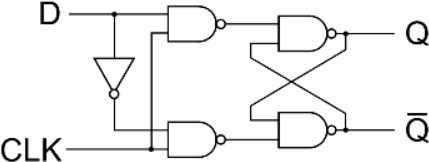
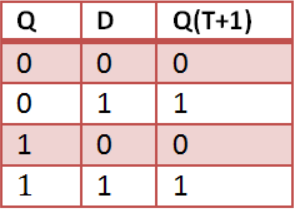
Synthesis tool: Xilinx VIVADO

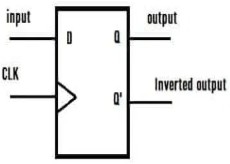
## Prelab Questions

*(write pre lab Q & A in an A4 sheet)*

* + 1. Write the difference between initial and always block.
    2. List the Reduction and Logical Operators.
    3. Give the use of Blocking and Nonblocking statments.
    4. Differentiate case, casex and casez statements.
    5. **Problem 1:** Write a Verilog code to implement Flip flops. The following points should be taken care of:
       1. Use If statement to design positive edge triggered D flip
       2. Use case statement to design negative edge triggered T flip flop

## Logic Diagram

D flip flop



T flip flop

## Verilog

**Code - Problem 1**

## 1.(a) POSITIVE EDGE TRIGGERED D FLIPFLOP USING IF STATEMENT

Module dff (q,qbar d, clk, clear);

output q; output qbar; input d; input clk; input clear; reg q, qbar;

always@(posedge clk or posedge clear) begin

if(clear== 1) begin

q <= 0;

qbar <= 1; end

else begin q <= d;

qbar = !d; end

end endmodule

// **TEST BENCH**

module dff\_tb\_v;

// Inputs reg d; reg clk; reg clear;

// Outputs wire q; wire qbar;

// Instantiate the Unit Under Test (UUT) dff uut (

.q(q),

.qbar(qbar),

.d(d),

.clk(clk),

.clear(clear)

);

initial begin

// Initialize Inputs

d = 0;clk = 0;clear = 1;

// Wait 100 ns for global reset to finish #100 d = 0;clear = 0;

#100 d = 1;clear = 0;

#100 d = 1;clear = 1;

// Add stimulus here end

always #50 clk=~clk; endmodule

## 1.2) NEGATIVE EDGE TRIGGERED T FLIPFLOP USING CASE STATEMENT

module tffcase(q, clk, clr, t); output q;

input clk; input clr;

input t; reg q; initial q=0;

always@(negedge clk) begin

case({clr,t}) 2'b10: q=0; 2'b00: q=q; 2'b01: q=~q; endcase

end endmodule

## // TEST BENCH

module tffcase\_tb\_v;

// Inputs

reg clk; reg clr; reg t;

//Outputs wire q;

//Instantiate the Unit Under Test (UUT) tffcase uut (.q(q),.clk(clk),.clr(clr),.t(t)); initial begin

//Initialize Inputs clk = 0; clr = 1; t = 0;

//Wait 100 ns for global reset to finish #100; clr=0; #100; clr=0; t=1;

//Add stimulus here end

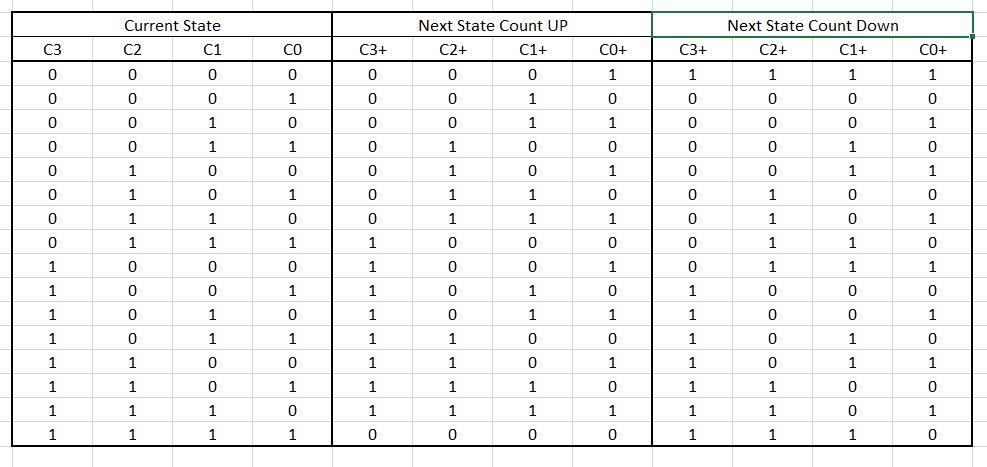
always

#50 clk=~clk; endmodule

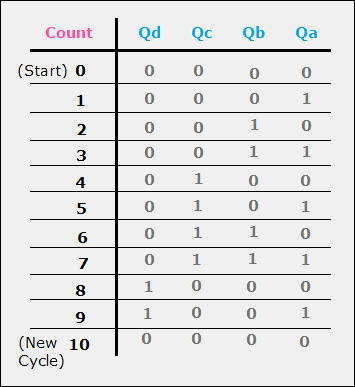
## Waveforms – Problem 1

* + 1. **Problem 2:** Write a Verilog code to implement Counters. The following points should be taken care of:
       1. Use behavioral model to design Up-Down Counter. When mode =’1’ do up counting and for mode=’0’ do down counting.
       2. Use behavioral model to design Mod-N counter. 3.Design SISO registers using behavioral model.

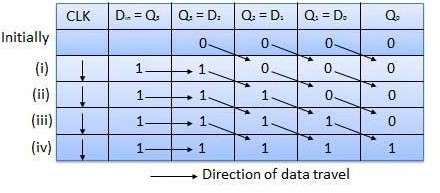
## Logic Diagram – Problem 2 Up-Down Counter:



**Mod-N Counter:**



## SISO Register:



**Verilog Code - Problem 2**

## 2.(1) UP DOWN COUNTER USING BEHAVIOURAL MODEL

module updowncntr(q, clr, clk, mode); output reg [3:0] q;

input clr; input clk; input mod;

always@(posedge clk) begin

case({clr,mod}) 2'b11 : q=0;

2'b10 : q=0;

2'b01 : q=q+1;

2'b00 : q=q-1;

endcase end endmodule

## // TEST BENCH

module updowncntr\_tb\_v;

// Inputs

reg clr; reg clk; reg mod;

//Outputs wire [3:0]q;

//Instantiate the Unit Under Test (UUT) updowncntr uut (.q(q),.clr(clr),.clk(clk),.mod(mod)); initial begin

// Initialize Inputs

clr = 1; clk = 0; mod = 1;

//Wait 100 ns for global reset to finish #100; clr=0; #1000; mod=0;

//Add stimulus here end

always

#50 clk=~clk; endmodule

## 2.(2) Mod N-COUNTER USING BEHAVIOURAL MODEL

module modN\_ctr

# (parameter N = 10, parameter WIDTH = 4) ( input clk,

input rstn,

output reg[WIDTH-1:0] out); always @ (posedge clk) begin if (rstn) begin

out <= 0;

end else begin if (out == N-1) out <= 0;

else

out <= out + 1; end

end endmodule

## // TEST BENCH

module modncounter\_tb\_v;

// Inputs reg clk; reg rstn;

// Outputs wire [3:0] out;

// Instantiate the Unit Under Test (UUT) modN\_ctr uut (

.clk(clk),

.rstn(rstn),

.out(out)

);

initial begin

// Initialize Inputs clk = 1;rstn = 1;

// Wait 100 ns for global reset to finish #100 rstn=0;

// Add stimulus here end

lways #50 clk=~clk; endmodule

## 2.(3) SISO REGISTER USING BEHAVIOURAL MODEL

module siso\_register(shift\_out, shift\_in, clk); input shift\_in;

input clk;

output shift\_out; reg shift\_out;

reg [2:0] data;

always @(negedge clk) begin

data[0] <= shift\_in ; data[1] <= data[0]; data[2] <= data[1]; shift\_out <= data[2]; end

endmodule

## // TEST BENCH

module siso\_tb\_v;

// Inputs reg shift\_in; reg clk;

// Outputs wire shift\_out;

// Instantiate the Unit Under Test (UUT) siso\_register uut (

.shift\_out(shift\_out),

.shift\_in(shift\_in),

.clk(clk)

);

initial begin

// Initialize Inputs shift\_in = 1;

clk = 1;

// Wait 100 ns for global reset to finish #100 shift\_in = 0;

#100 shift\_in = 1;

#100 shift\_in = 0;

// Add stimulus here

end

always #50 clk=~clk; endmodule

## Waveforms – Problem 2

## Post Lab :

Write a Verilog HDL Code to implement a SIPO and PIPO shift registers.

## Result: