DDCO Mini Project ISA Submission

Project Title: 4-bit Up Counter, Down Counter

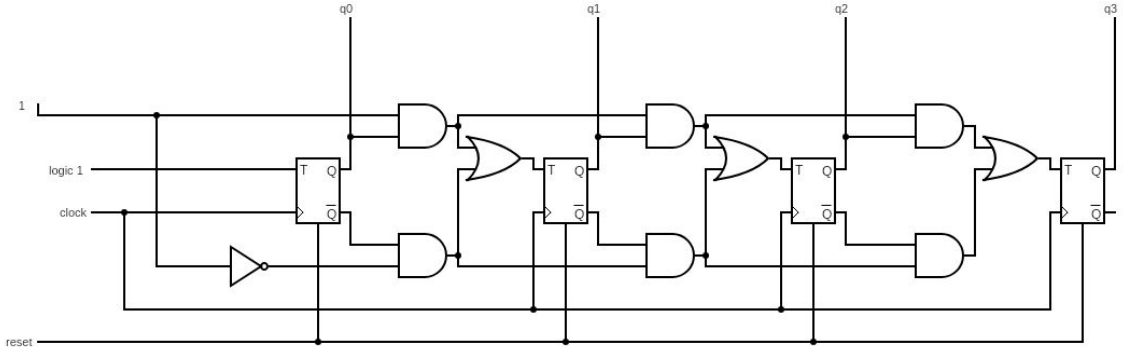
Section : E

Batch Details

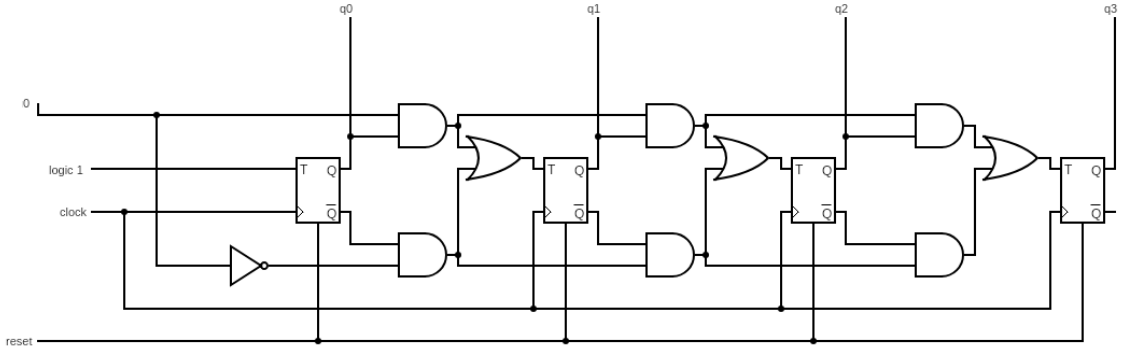
Student Name	SRN
Nemalli Vishnu Vardhan Reddy	PES1UG19CS295
Nerella Venkata Tarunika	PES1UG19CS296
Nihal Ramaswamy	PES1UG19CS297
Nihal Shetty	PES1UG19CS298

Circuit Diagram:

1. Up Counter



2. Down Counter



Implementation (Code):

//Implementation file

```
module and2 (input wire input_wire0, input_wire1, output wire output_wire);

    assign output_wire = input_wire0 & input_wire1;

endmodule

module or2 (input wire input_wire0, input_wire1, output wire output_wire);

    assign output_wire = input_wire0 | input_wire1;

endmodule
//This is the module for the T flip flop
module toggle_flip_flop (input wire clock_signal,reset_wire,toggle_wire, output
data_wire);

    reg data_wire;
    always @ (posedge clock_signal)
    begin
    if(reset_wire)
        data_wire=4'b0000;
    else
        if(toggle_wire) data_wire=~data_wire;
        else data_wire=data_wire;
```

```
end


endmodule
//This module is for the down counter
module down_counter (input wire  clock_signal, reset_signal, output wire[3:0]q);

        wire [5:0]a;
        wire [2:0]or_out;

        toggle_flip_flop t1 ( clock_signal, reset_signal, 1'b1, q[0]);
        and2 and2_1 (q[0], 1'b0, a[0]);
        and2 and2_2 (~q[0], 1'b1, a[1]);
        or2 or2_1 (a[0], a[1], or_out[0]);

        toggle_flip_flop t2 ( clock_signal, reset_signal, or_out[0], q[1]);
        and2 and2_3 (q[1], a[0], a[2]);
        and2 and2_4 (~q[1], a[1], a[3]);
        or2 or2_2 (a[2], a[3], or_out[1]);

        toggle_flip_flop t3 ( clock_signal, reset_signal, or_out[1], q[2]);
        and2 and2_5 (q[2], a[2], a[4]);
        and2 and2_6 (~q[2], a[3], a[5]);
        or2 or2_3 (a[4], a[5], or_out[2]);
```

```
        toggle_flip_flop t4 ( clock_signal, reset_signal, or_out[2], q[3]);

endmodule

//This module is for the up counter
module up_counter (input wire  clock_signal, reset_signal, output wire[3:0]q);

        wire [5:0]a;
        wire [2:0]or_out;

        toggle_flip_flop t1 ( clock_signal, reset_signal, 1'b1, q[0]);
        and2 and2_1 (q[0], 1'b1, a[0]);
        and2 and2_2 (~q[0], 1'b0, a[1]);
        or2 or2_1 (a[0], a[1], or_out[0]);

        toggle_flip_flop t2 ( clock_signal, reset_signal, or_out[0], q[1]);
        and2 and2_3 (q[1], a[0], a[2]);
        and2 and2_4 (~q[1], a[1], a[3]);
        or2 or2_2 (a[2], a[3], or_out[1]);

        toggle_flip_flop t3 ( clock_signal, reset_signal, or_out[1], q[2]);
        and2 and2_5 (q[2], a[2], a[4]);
        and2 and2_6 (~q[2], a[3], a[5]);
        or2 or2_3 (a[4], a[5], or_out[2]);
```

```
        toggle_flip_flop t4 ( clock_signal, reset_signal, or_out[2], q[3]);

endmodule

//Testbench
module counter_tb();

reg clock,reset;
wire [3:0]q_up,q_down;
//This calls up counter
up_counter uc(clock,reset,q_up);
//This calls down counter
down_counter dc(clock,reset,q_down);

initial begin
        $dumpfile("counter4_tb.vcd");
        $dumpvars(0,counter_tb);
end

initial
begin
clock=0; repeat(70) #10 clock=~clock;
end
```

```
//This is where the reset wire becomes 1 and 0 accordingly
initial
begin
reset=1'b1;
#30
reset=1'b0;
#160
reset=1'b1;
#20
reset=1'b0;
end
endmodule
//Link to project: https://github.com/nihal-ramaswamy/DDCO-project
```

Output: