**LAB 4 – Raja Aadhithan**

Design – D Flip Flop:

Code:

module dfflop(input clock,reset,d\_in, output Q\_out,Qb\_out);

reg x;

   always@(posedge clock) begin

    if(reset) x <= 1'b0;

    else x <= d\_in;

   end

   assign Q\_out = x;

   assign Qb\_out = !x;

endmodule

Testbench:

module count\_tb();

reg clk,reset,load;

reg [3:0]i;

wire [3:0]q;

count4 dut(clk,reset,load,i,q);

initial begin

    clk = 1;

    forever #5 clk = ~clk;

end

initial begin

    $monitor("@time : %3d - for load = %b output is %b",$time,load,q);

    reset <= 1;

    #15;

    reset <= 0;

    load <= 1;

    i <= 4'b1101;

    #15;

    load <=0;

    #40;

    reset <= 1;

    #15;

    reset <= 0;

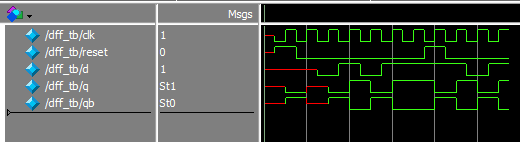
    #300;

    $finish;

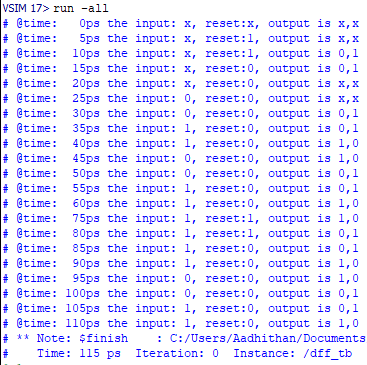
end

endmodule

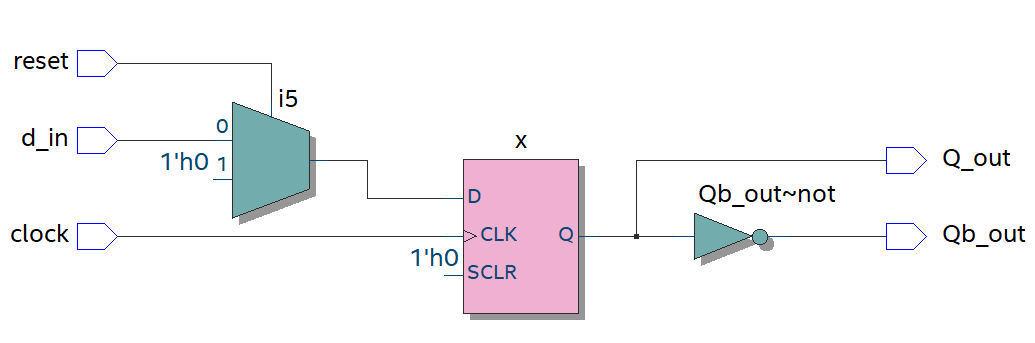
Wave:



Output:



RTL:



Exercises:

Design : SR latch:

Code:

module sr(input s,r,output q,qb);

nor(q,s,qb);

nor(qb,r,q);

endmodule

Testbench:

module sr\_tb();

reg a,b;

wire q,qb;

integer i,j;

sr dut(a,b,q,qb);

initial begin

    $monitor("@time %2d input is %b,%b  - output is %b,%b",$time,a,b,q,qb);

    for (i=0;i<4;i = i+1)

    begin

        {a,b}=i;

        #10;

    end

    for (j=1;j<4;j = j+1)

    begin

        {a,b}=j;

        #10;

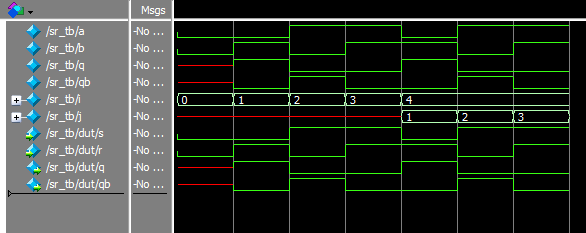
    end

    $finish;

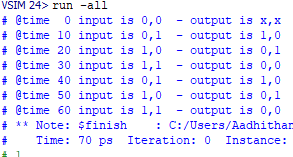
end

endmodule

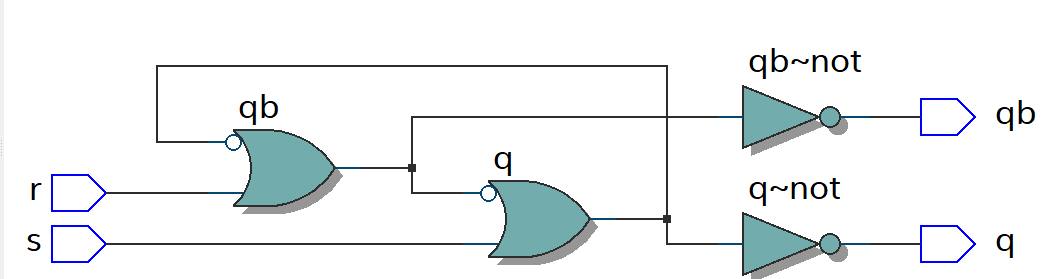
Wave:



Output:



RTL:



Design: JK Flip Flop:

Code:

module jkfflop(input j,k,clk,output reg q,wire qb);

parameter HOLD=2'b00, RESET=2'b01, SET=2'b10, TOGGLE=2'b11;

always@(posedge clk)begin

    case({j,k})

    RESET : q <= 0;

    SET   : q <= 1;

    TOGGLE: q <= qb;

    endcase

end

assign qb = ~q;

endmodule

Testbench:

module jk\_tb();

reg j,k,clk;

wire q,qb;

integer i;

jkfflop dut(j,k,clk,q,qb);

initial begin

    clk = 1'b1;

    forever #5 clk = ~clk;

end

initial begin

    $monitor("@time : %3d: j is %b, k is %b, output is %b-%b",$time,j,k,q,qb);

    for(i=0;i<8;i=i+1)

    begin

        {j,k}=i[1:0];

        #10;

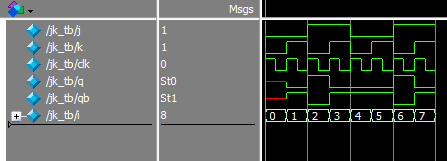
    end

    $finish;

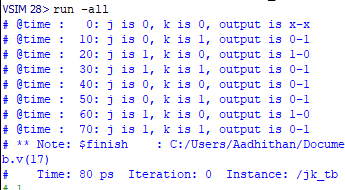
end

endmodule

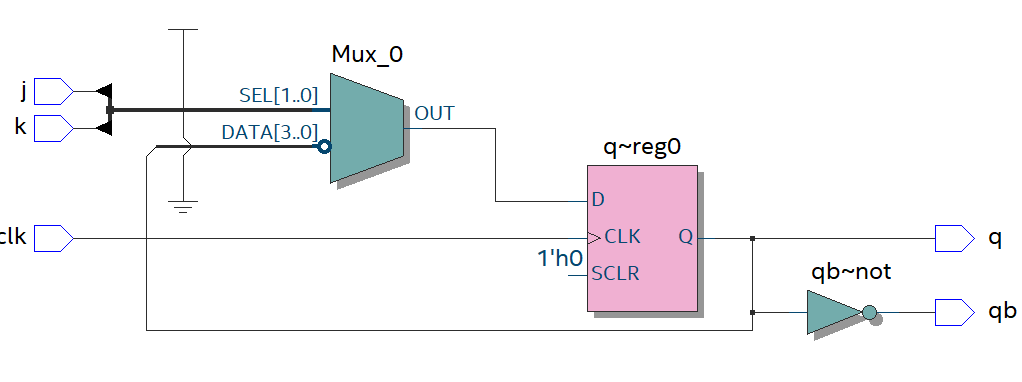
Wave:



Output:



RTL:



Design : T Flip Flop:

Code:

module tfflop(input t,clk,reset,output reg q, wire qb);

wire d;

assign d = t^q;

assign qb = !q;

always@(posedge clk)begin

    if (reset) q<=0;

    else q<=d;

end

endmodule

Testbench:

module tff\_tb();

reg t,clk,reset;

wire q,qb;

integer i;

tfflop dut(t,clk,reset,q,qb);

initial begin

    clk = 1;

    forever #5 clk = ~clk;

end

initial begin

    $monitor("@time = %3d: T is %b and outputs are %b - %b",$time,t,q,qb);

    reset = 1'b1;

    #5;

    reset = 1'b0;

    for(i=0;i<8;i=i+1)

    begin

        t = i[0];

        #10;

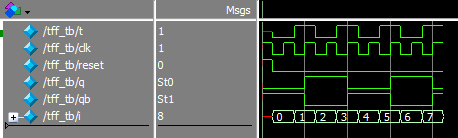
    end

    $finish;

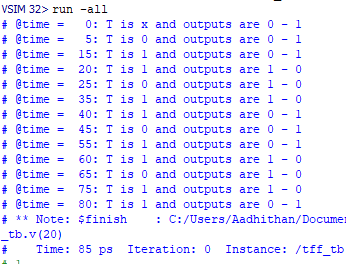
end

endmodule

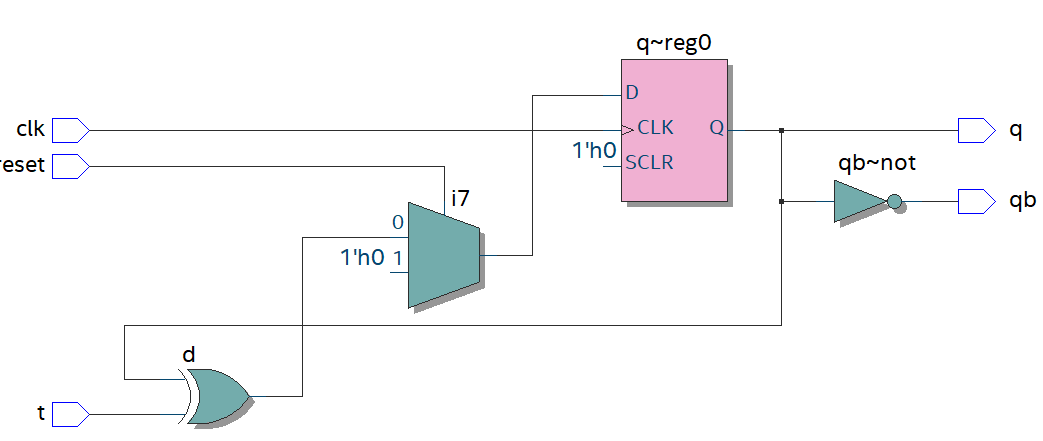
Wave:



Output:



RTL:



Design: 4 bit synchronous up counter:

Code:

module count4(input clk,reset,load,[3:0]i,output reg [3:0]q);

always@(posedge clk)begin

    if(reset) q <= 4'd0;

    else if (load) q <= i;

    else q <= q+1;

end

endmodule

Testbench:

module count\_tb();

reg clk,reset,load;

reg [3:0]i;

wire [3:0]q;

count4 dut(clk,reset,load,i,q);

initial begin

    clk = 1;

    forever #5 clk = ~clk;

end

initial begin

    $monitor("@time : %3d - for load = %b output is %b",$time,load,q);

    reset <= 1;

    #15;

    reset <= 0;

    load <= 1;

    i <= 4'b1101;

    #15;

    load <=0;

    #40;

    reset <= 1;

    #15;

    reset <= 0;

    #300;

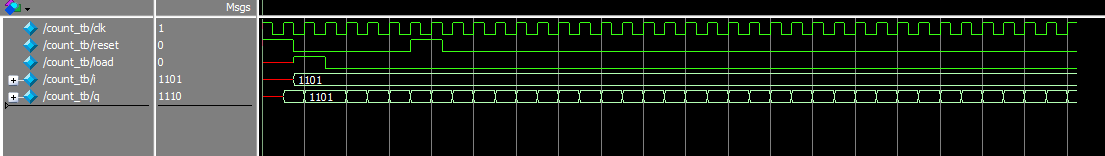
    $finish;

end

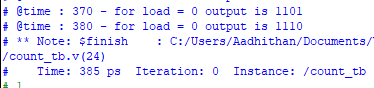
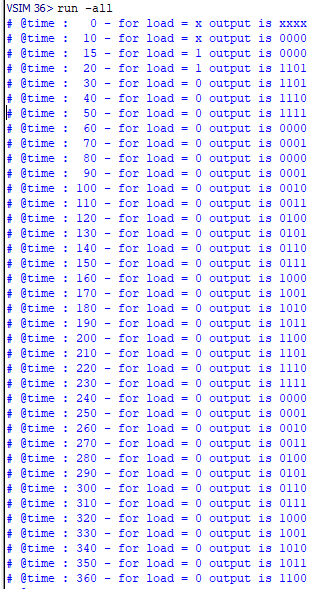
endmodule

Wave:

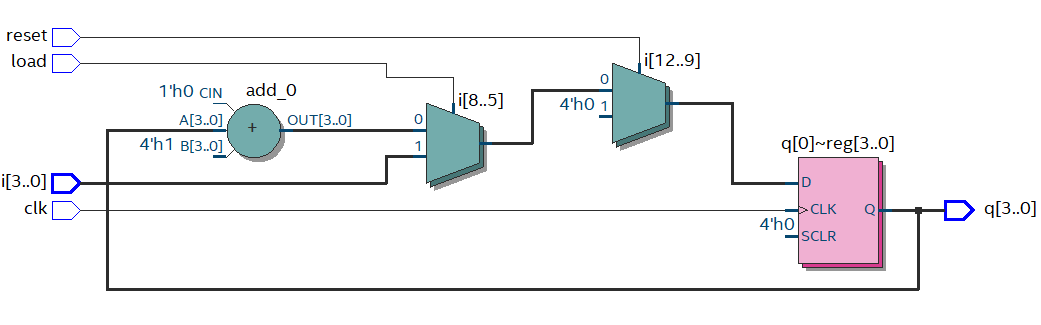
Zoomed out version:



Output:



RTL:



Design: Mod 12 counter:

Code:

module count4(input clk,reset,load,[3:0]i,output reg [3:0]q);

always@(posedge clk)begin

    if(reset||q==4'b1011) q <= 4'd0;

    else if (load) q <= i;

    else q <= q+1;

end

endmodule

Testbench:

module count\_tb();

reg clk,reset,load;

reg [3:0]i;

wire [3:0]q;

count4 dut(clk,reset,load,i,q);

initial begin

    clk = 1;

    forever #5 clk = ~clk;

end

initial begin

    $monitor("@time : %3d - for load = %b output is %b",$time,load,q);

    reset <= 1;

    #15;

    reset <= 0;

    load <= 1;

    i <= 4'b1101;

    #15;

    load <=0;

    #40;

    reset <= 1;

    #15;

    reset <= 0;

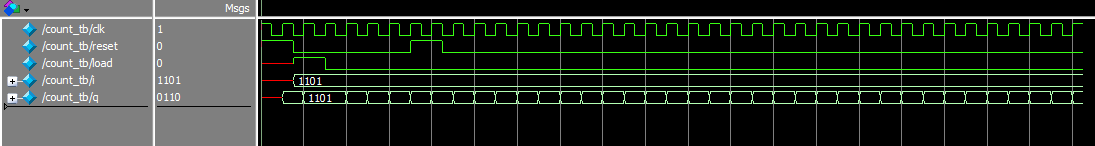
    #300;

    $finish;

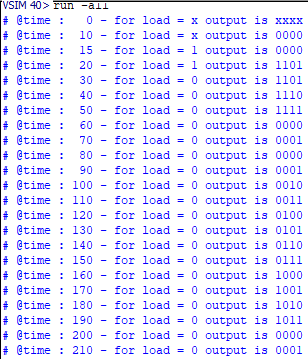
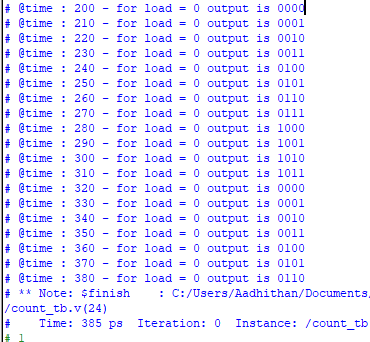
end

endmodule

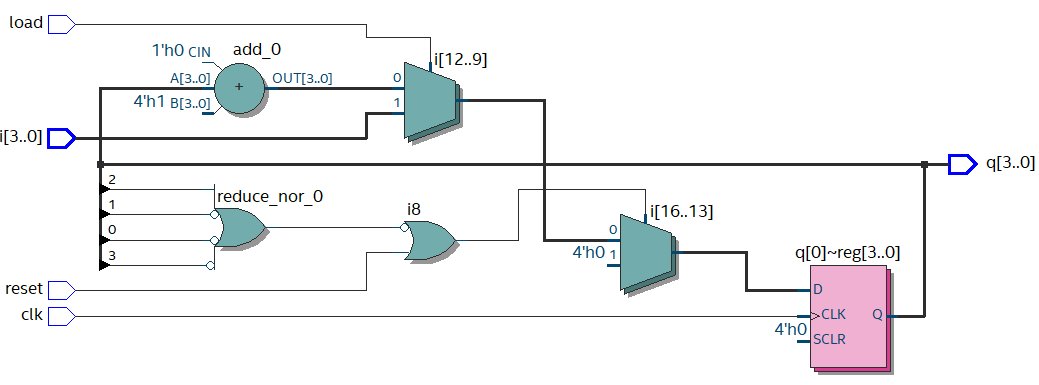
Wave:



Output:

RTL:



Design : up/down counter:

Code:

module count4(input clk,reset,load,up,[3:0]i,output reg [3:0]q);

always@(posedge clk)begin

    if(reset) q <= 4'd0;

    else if (load) q <= i;

    else if (up) q <= q+1;

    else if (!up) q<=q-1;

end

endmodule

Testbench:

module count\_tb();

reg clk,reset,load,up;

reg [3:0]i;

wire [3:0]q;

count4 dut(clk,reset,load,up,i,q);

initial begin

    clk = 1;

    forever #5 clk = ~clk;

end

initial begin

    $monitor("@time : %3d - for load = %b for operation up =%b  output is %b",$time,load,up,q);

    reset <= 1;

    #15;

    reset <= 0;

    up <=1;

    load <= 1;

    i <= 4'b1101;

    #15;

    load <=0;

    #40;

    up <=0;

    #40;

    reset <= 1;

    #15;

    reset <= 0;

    #300;

    up<=1;

    #300;

    up<=0;

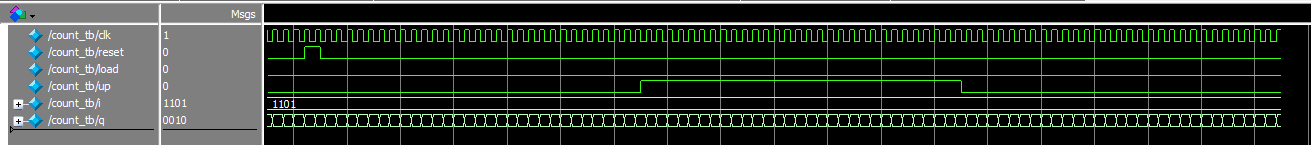
    #300;

    $finish;

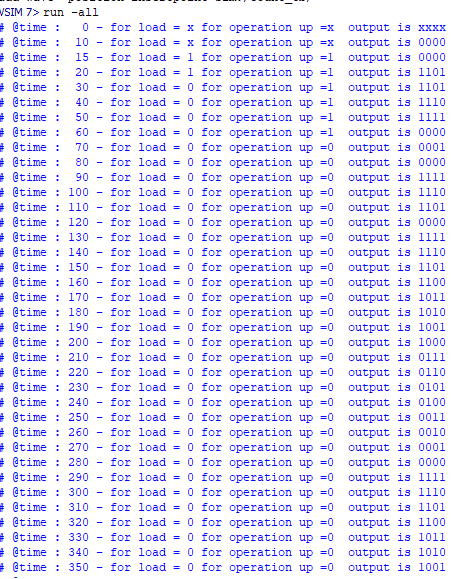
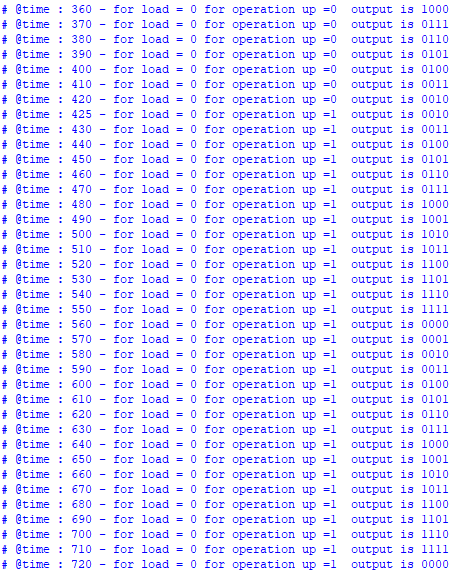
end

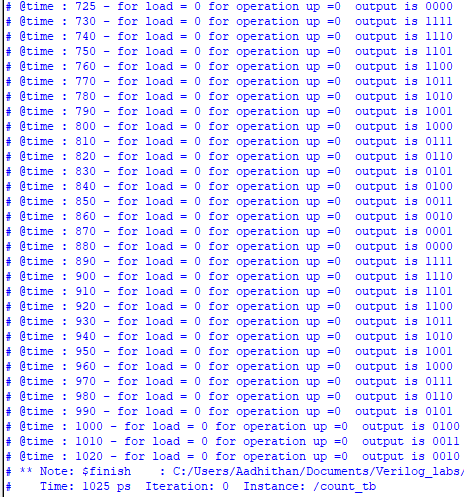
endmodule

Wave:



Output:



RTL:

