2\_4 decoder

module decoder\_2\_4(a,b,w,x,y,z);

output w,x,y,z;

input a,b;

assign w = (~a) & (~b);

assign x = (~a) & b;

assign y = a & (~b);

assign z = a & b;

endmodule

testbench

module dec\_tb;

// Inputs

reg a;

reg b;

// Outputs

wire w;

wire x;

wire y;

wire z;

// Instantiate the Unit Under Test (UUT)

decoder\_2\_4 uut (

.a(a),

.b(b),

.w(w),

.x(x),

.y(y),

.z(z)

);

initial begin

// Initialize Inputs

// Wait 100 ns for global reset to finish

#000 a=0; b=0;

#100 a=0; b=1;

#100 a=1; b=0;

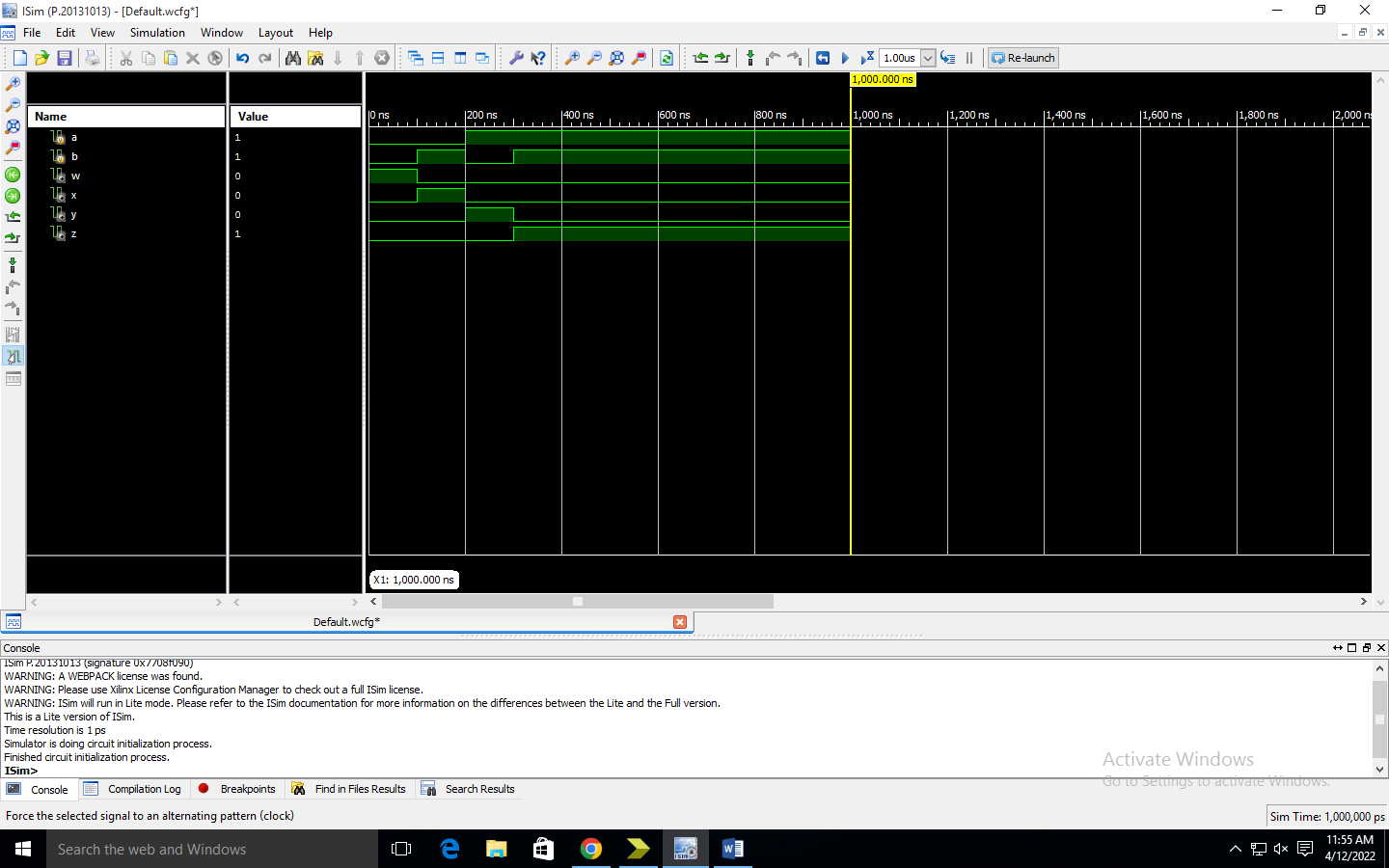
#100 a=1; b=1;

// Add stimulus here

end

endmodule

simulation



Synthesis report

=========================================================================

\* Final Report \*

=========================================================================

Final Results

RTL Top Level Output File Name : decoder\_2\_4.ngr

Top Level Output File Name : decoder\_2\_4

Output Format : NGC

Optimization Goal : Speed

Keep Hierarchy : No

Design Statistics

# IOs : 6

Cell Usage :

# BELS : 4

# LUT2 : 4

# IO Buffers : 6

# IBUF : 2

# OBUF : 4

=========================================================================

Device utilization summary:

---------------------------

Selected Device : 3s400pq208-5

Number of Slices: 2 out of 3584 0%

Number of 4 input LUTs: 4 out of 7168 0%

Number of IOs: 6

Number of bonded IOBs: 6 out of 141 4%

---------------------------

Partition Resource Summary:

---------------------------

No Partitions were found in this design.

---------------------------

=========================================================================

TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.

FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT

GENERATED AFTER PLACE-and-ROUTE.

Clock Information:

------------------

No clock signals found in this design

Asynchronous Control Signals Information:

----------------------------------------

No asynchronous control signals found in this design

Timing Summary:

---------------

Speed Grade: -5

Minimum period: No path found

Minimum input arrival time before clock: No path found

Maximum output required time after clock: No path found

Maximum combinational path delay: 7.858ns

Timing Detail:

--------------

All values displayed in nanoseconds (ns)

=========================================================================

Timing constraint: Default path analysis

Total number of paths / destination ports: 8 / 4

-------------------------------------------------------------------------

Delay: 7.858ns (Levels of Logic = 3)

Source: a (PAD)

Destination: w (PAD)

Data Path: a to w

Gate Net

Cell:in->out fanout Delay Delay Logical Name (Net Name)

---------------------------------------- ------------

IBUF:I->O 4 0.715 1.074 a\_IBUF (a\_IBUF)

LUT2:I0->O 1 0.479 0.681 z1 (z\_OBUF)

OBUF:I->O 4.909 z\_OBUF (z)

----------------------------------------

Total 7.858ns (6.103ns logic, 1.755ns route)

(77.7% logic, 22.3% route)

=========================================================================

Total REAL time to Xst completion: 4.00 secs

Total CPU time to Xst completion: 3.81 secs

-->

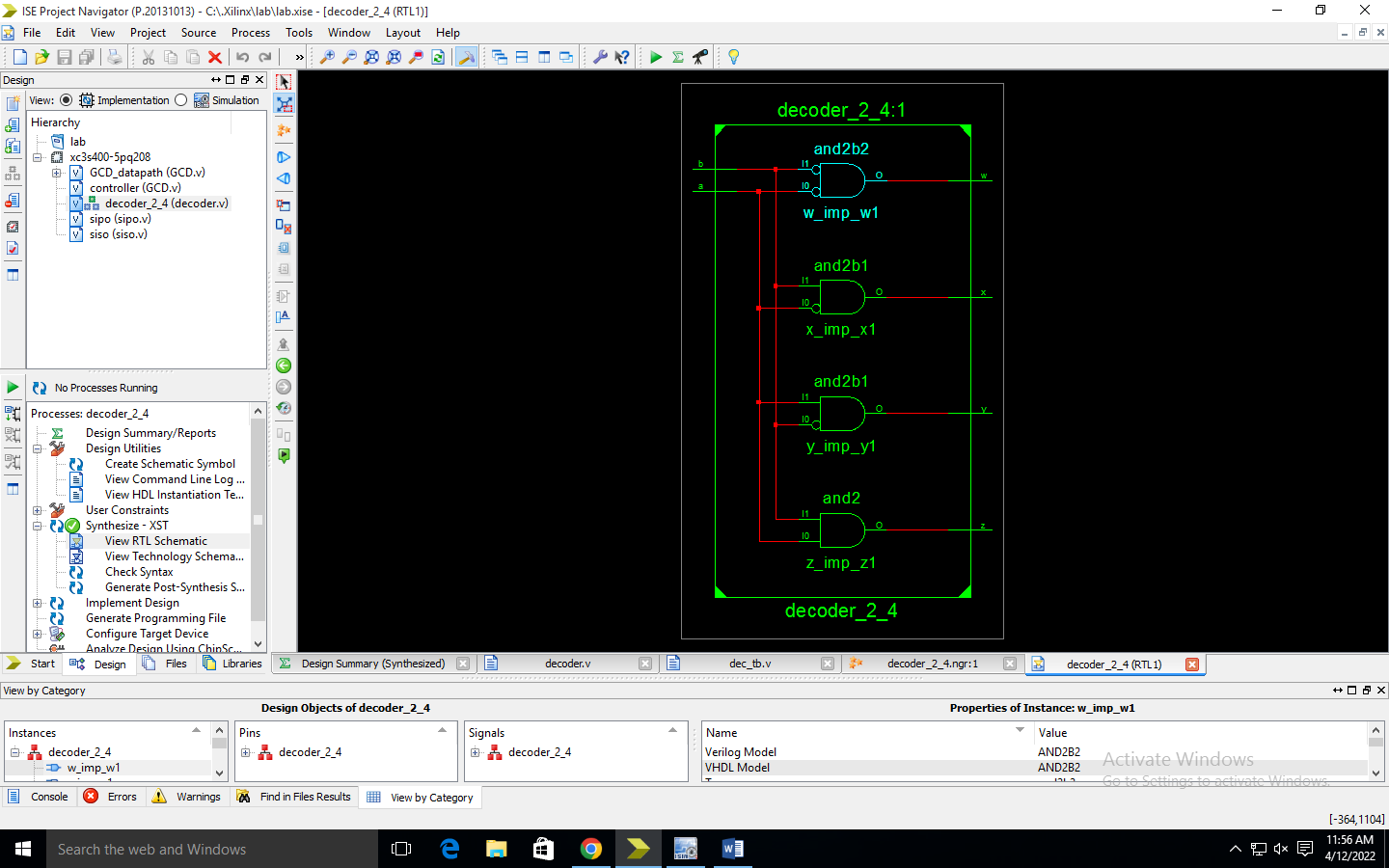
Total memory usage is 289464 kilobytes

Number of errors : 0 ( 0 filtered)

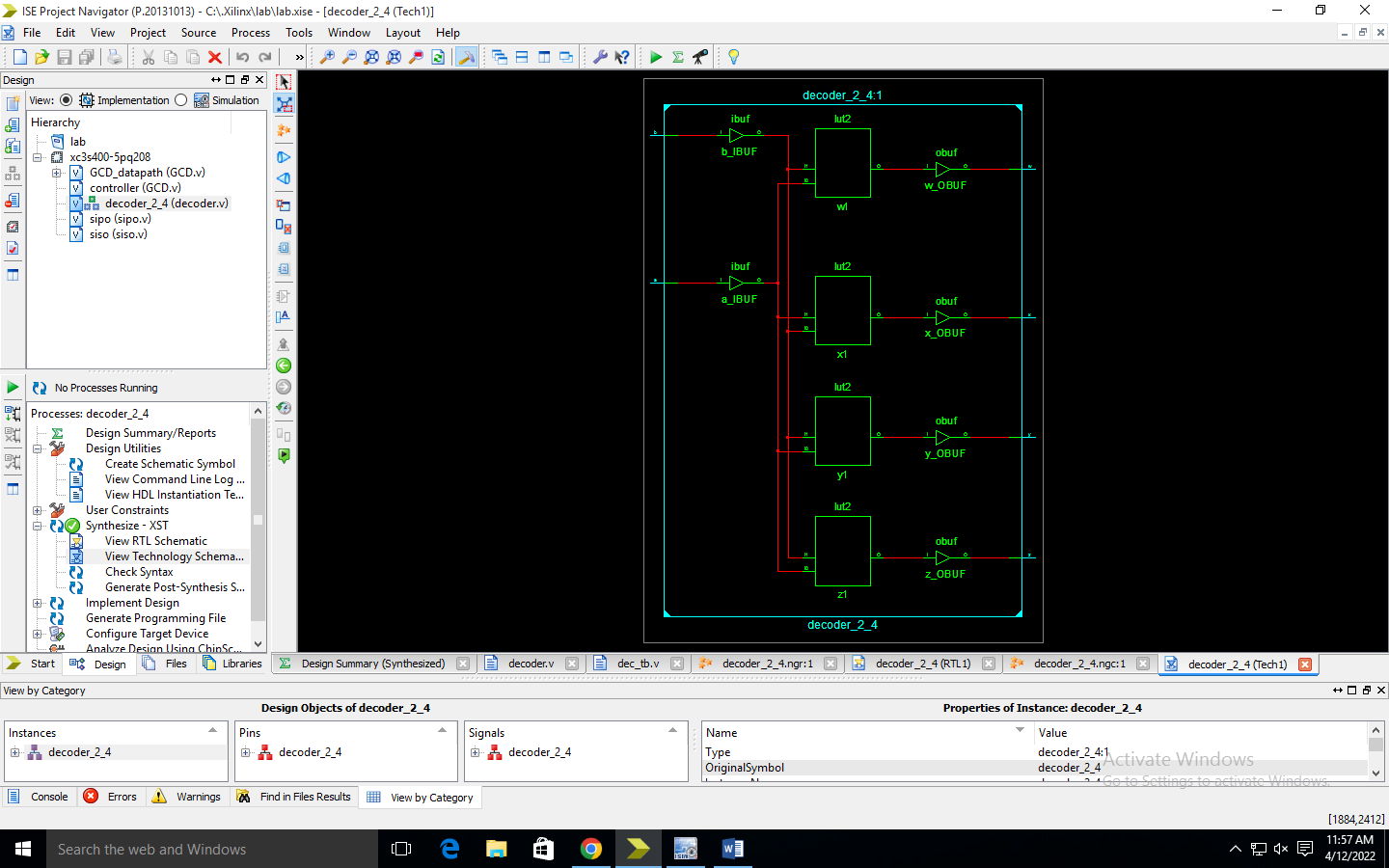
Number of warnings : 0 ( 0 filtered)

Number of infos : 0 ( 0 filtered)

Rtl schematic



Technology schematic



8\_3 encoder without priority

module encoder (din, dout);

input [7:0] din;

output [2:0] dout;

reg [2:0] dout;

always @(din)

begin

if (din ==8'b00000001) dout=3'b000;

else if (din==8'b00000010) dout=3'b001;

else if (din==8'b00000100) dout=3'b010;

else if (din==8'b00001000) dout=3'b011;

else if (din==8'b00010000) dout=3'b100;

else if (din ==8'b00100000) dout=3'b101;

else if (din==8'b01000000) dout=3'b110;

else if (din==8'b10000000) dout=3'b111;

else dout=3'bX;

end

endmodule

testbench

module enco\_tb;

// Inputs

reg [7:0] din;

// Outputs

wire [2:0] dout;

// Instantiate the Unit Under Test (UUT)

encoder uut (

.din(din),

.dout(dout)

);

initial begin

#10 din=8'b10000000;

#10 din=8'b01000000;

#10 din=8'b00100000;

#10 din=8'b00010000;

#10 din=8'b00001000;

#10 din=8'b00000100;

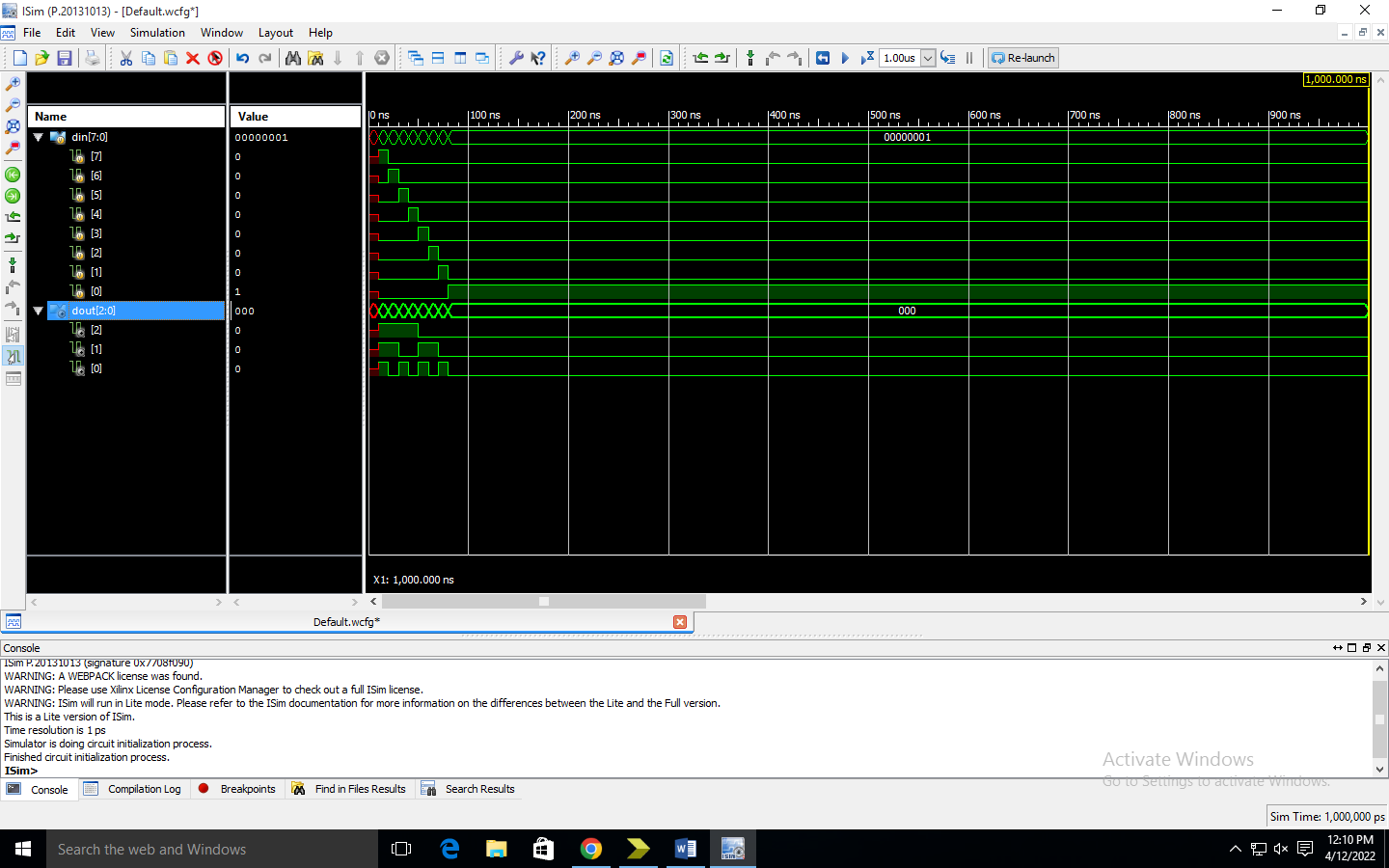
#10 din=8'b00000010;

#10 din=8'b00000001;

end

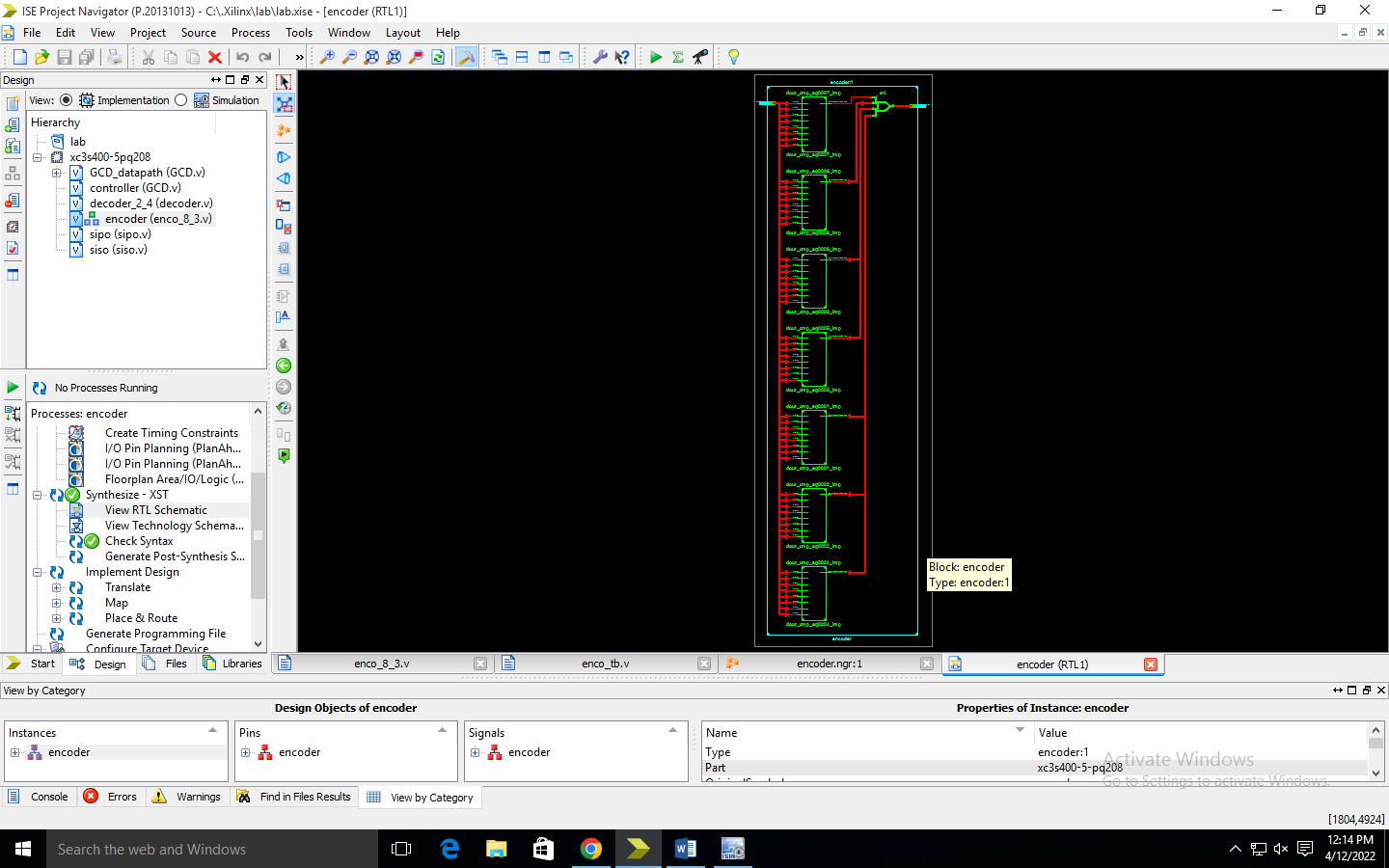
endmodule

simulation

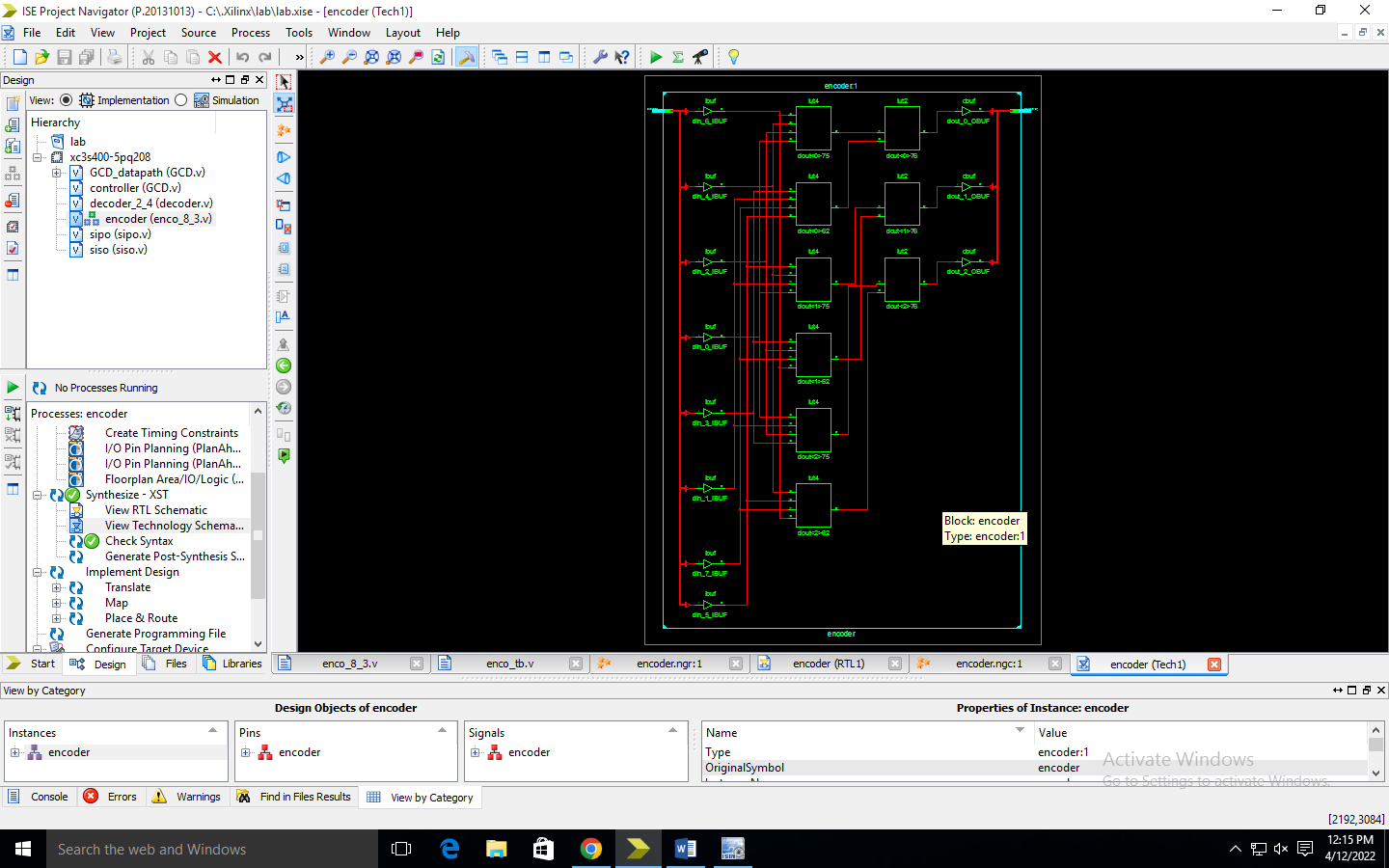


Synthesis report

Rtl schematic



Technology schematic



8\_3 with priority encoder

module prio\_enco\_8x3(dout, din);

output [2:0] dout;

input [7:0] din ;

assign dout = (din[7] ==1'b1 ) ? 3'b111:

(din[6] ==1'b1 ) ? 3'b110:

(din[5] ==1'b1 ) ? 3'b101:

(din[4] ==1'b1) ? 3'b100:

(din[3] ==1'b1) ? 3'b011:

(din[2] ==1'b1) ? 3'b010:

(din[1] ==1'b1) ? 3'b001:

(din[0] ==1'b1) ? 3'b000: 3'bxxx;

Endmodule

Testbench

module prio\_tb;

// Inputs

reg [7:0] din;

// Outputs

wire [2:0] dout;

// Instantiate the Unit Under Test (UUT)

prio\_enco\_8x3 uut (

.dout(dout),

.din(din)

);

initial begin

d\_in=8'b11001100; #10;

d\_in=8'b01100110; #10;

d\_in=8'b00110011; #10;

d\_in=8'b00010010; #10;

d\_in=8'b00001001; #10;

d\_in=8'b00000100; #10;

d\_in=8'b00000011; #10;

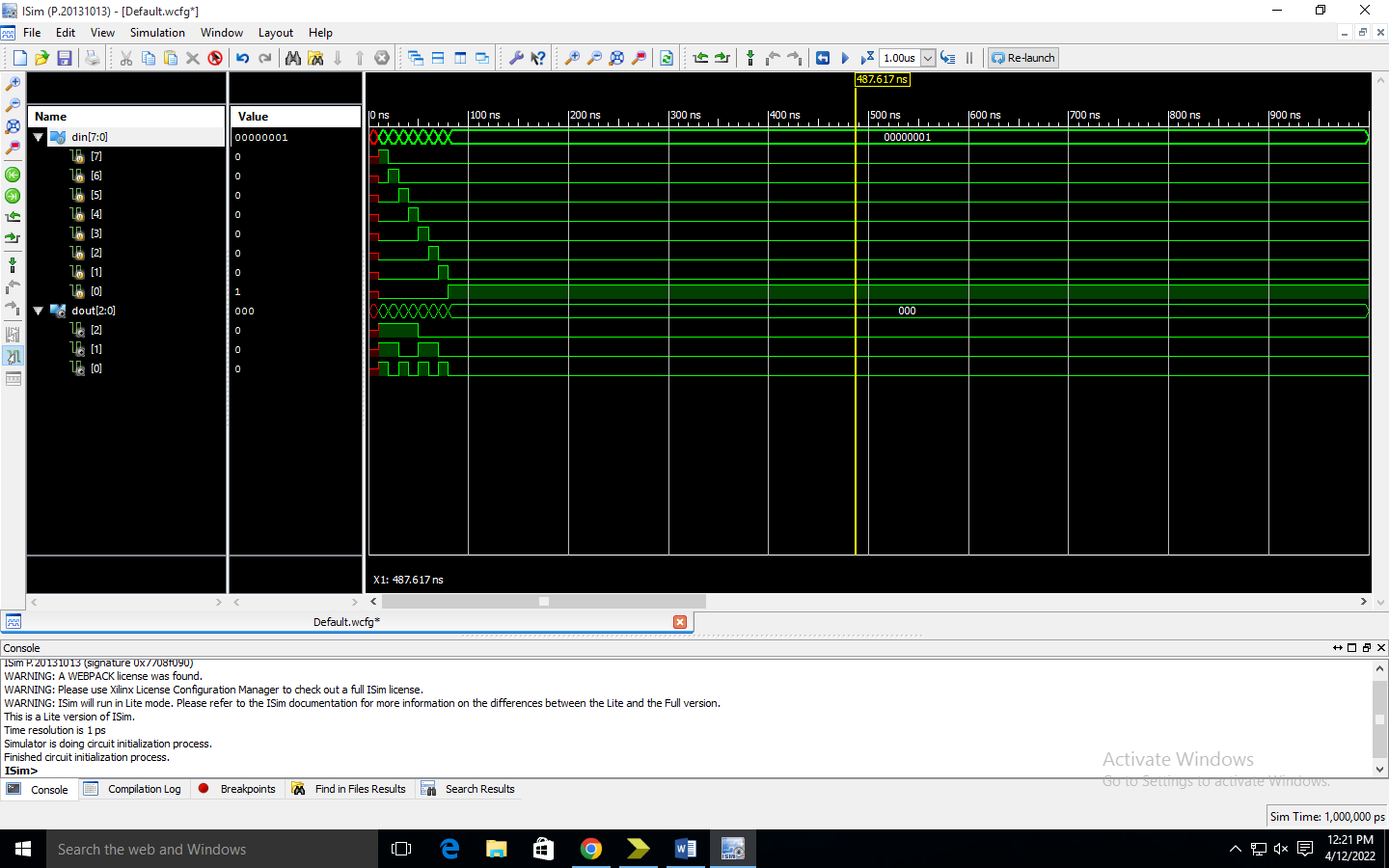
d\_in=8'b00000001; #10;

d\_in=8'b00000000; # 10;

end

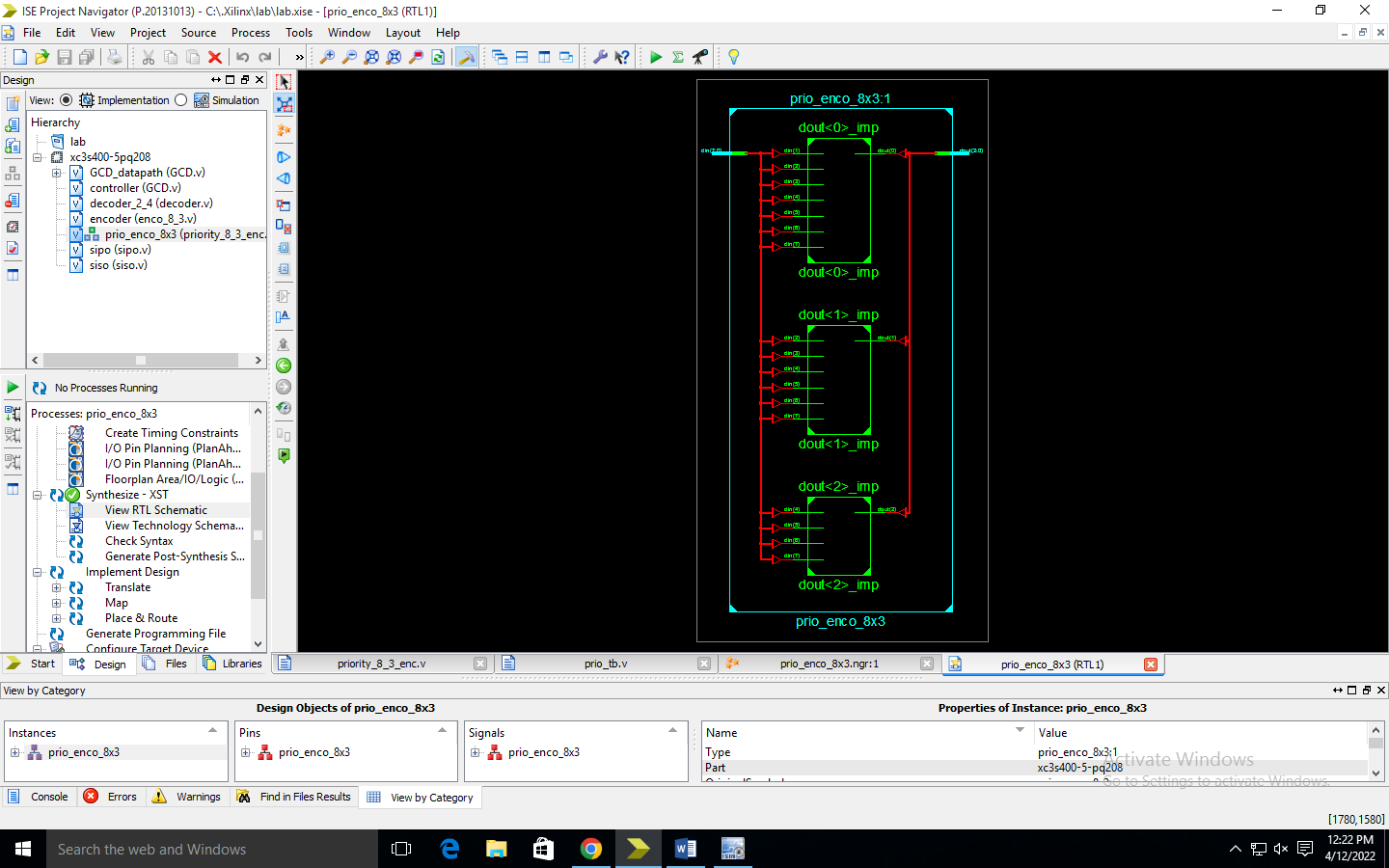
endmodule

simulation

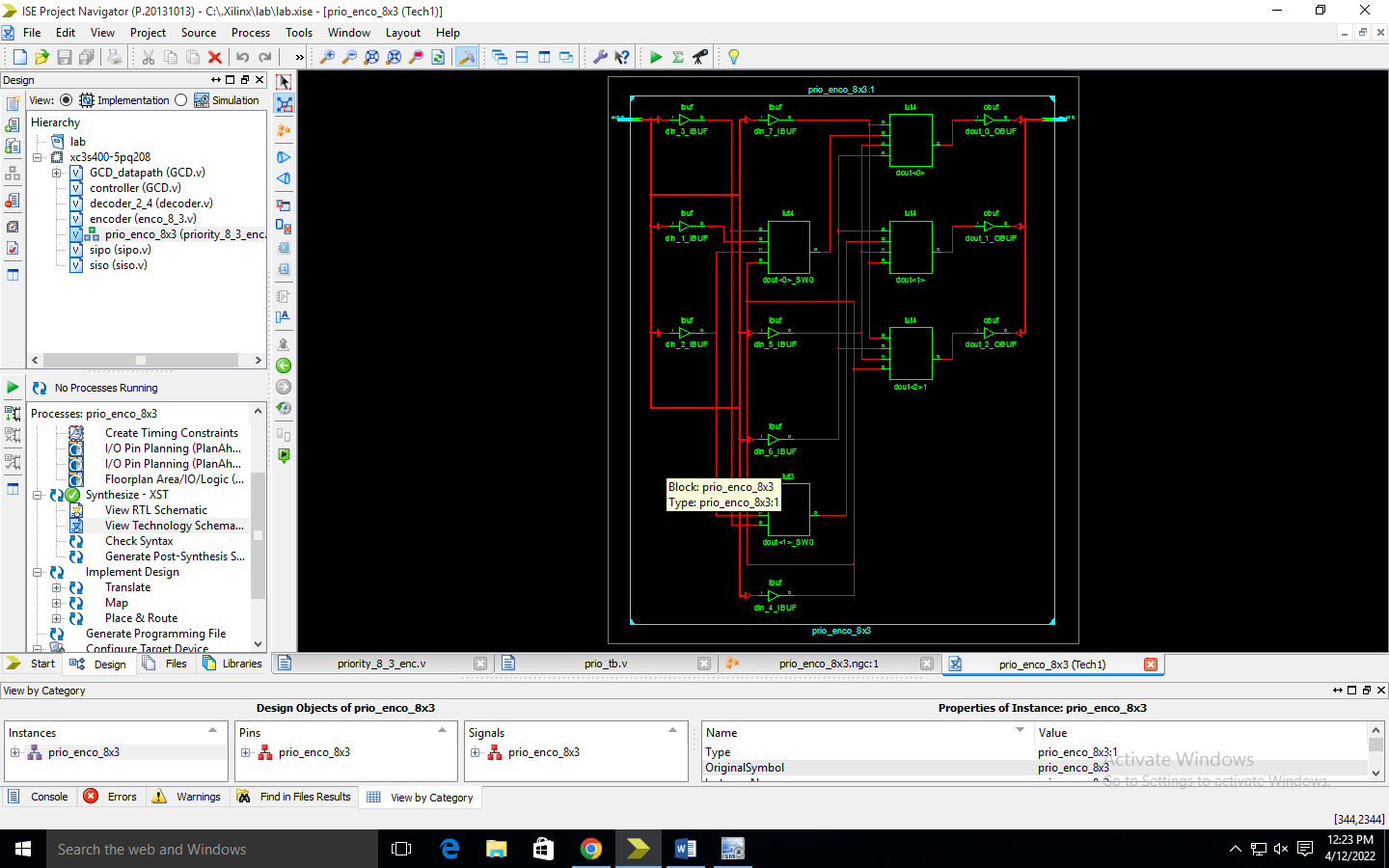


Synthesis report

Rtl schematic



Technology schematic



8:1 multiplexer

module mux\_3x8(in,out,sel);

input [7:0]in;

input [2:0]sel;

output out;

reg out;

wire [2:0]sel;

wire [7:0]in;

always @(sel or in)

begin

if (sel==0)

out = in[0];

if (sel==1)

out = in[1];

if (sel==2)

out = in[2];

if (sel==3)

out = in[3];

if (sel==4)

out = in[4];

if (sel==5)

out = in[5];

if (sel==6)

out = in[6];

if (sel==7)

out = in[7];

end

endmodule

testbench

module mux\_3x8\_tb;

wire out;

reg [2:0]sel;

reg [7:0]in;

mux\_3x8 mux( .out(out), .in(in), .sel(sel) );

initial begin

$monitor(sel,in,out);

sel=3'b000;

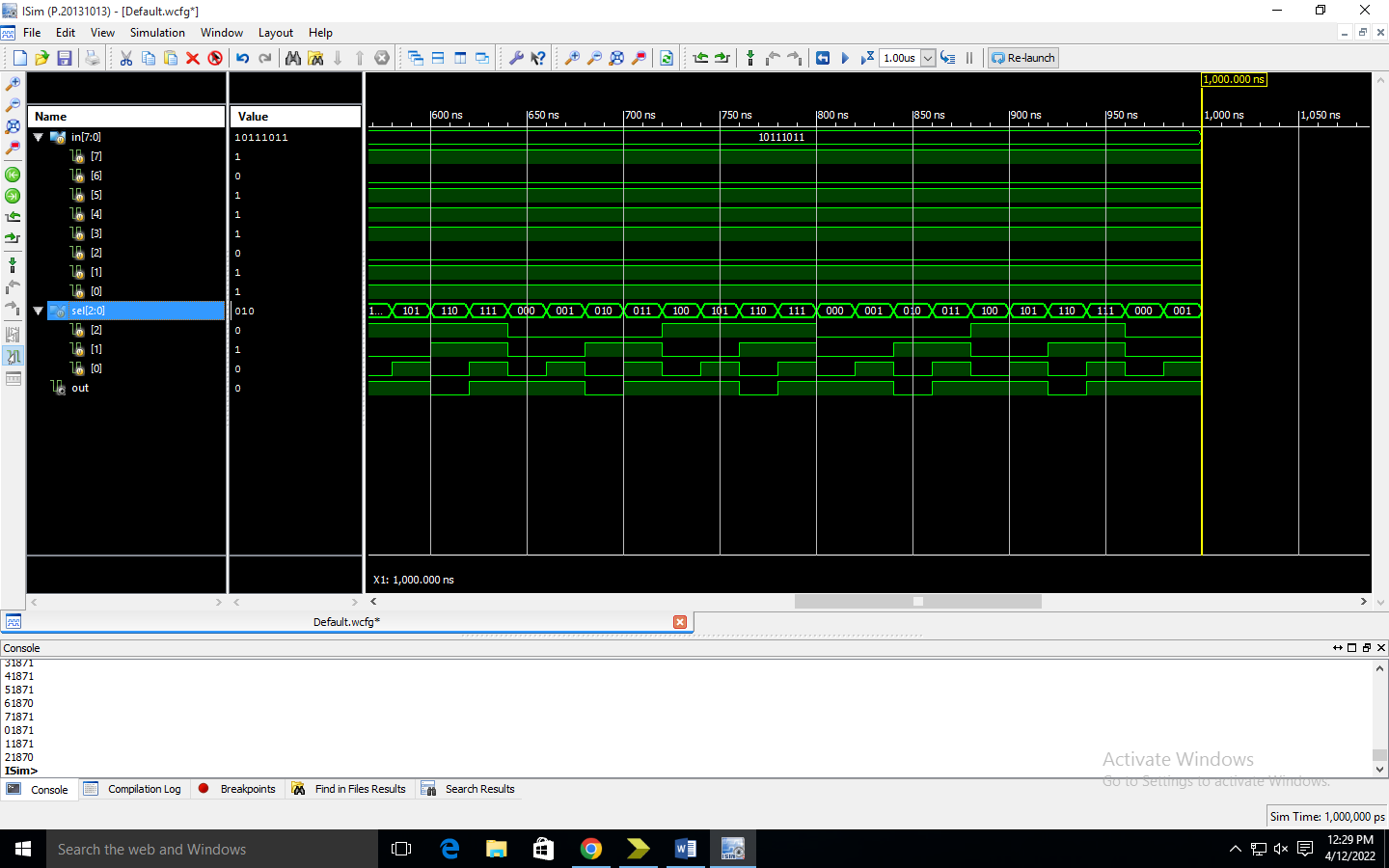
in=8'b10111011;

end

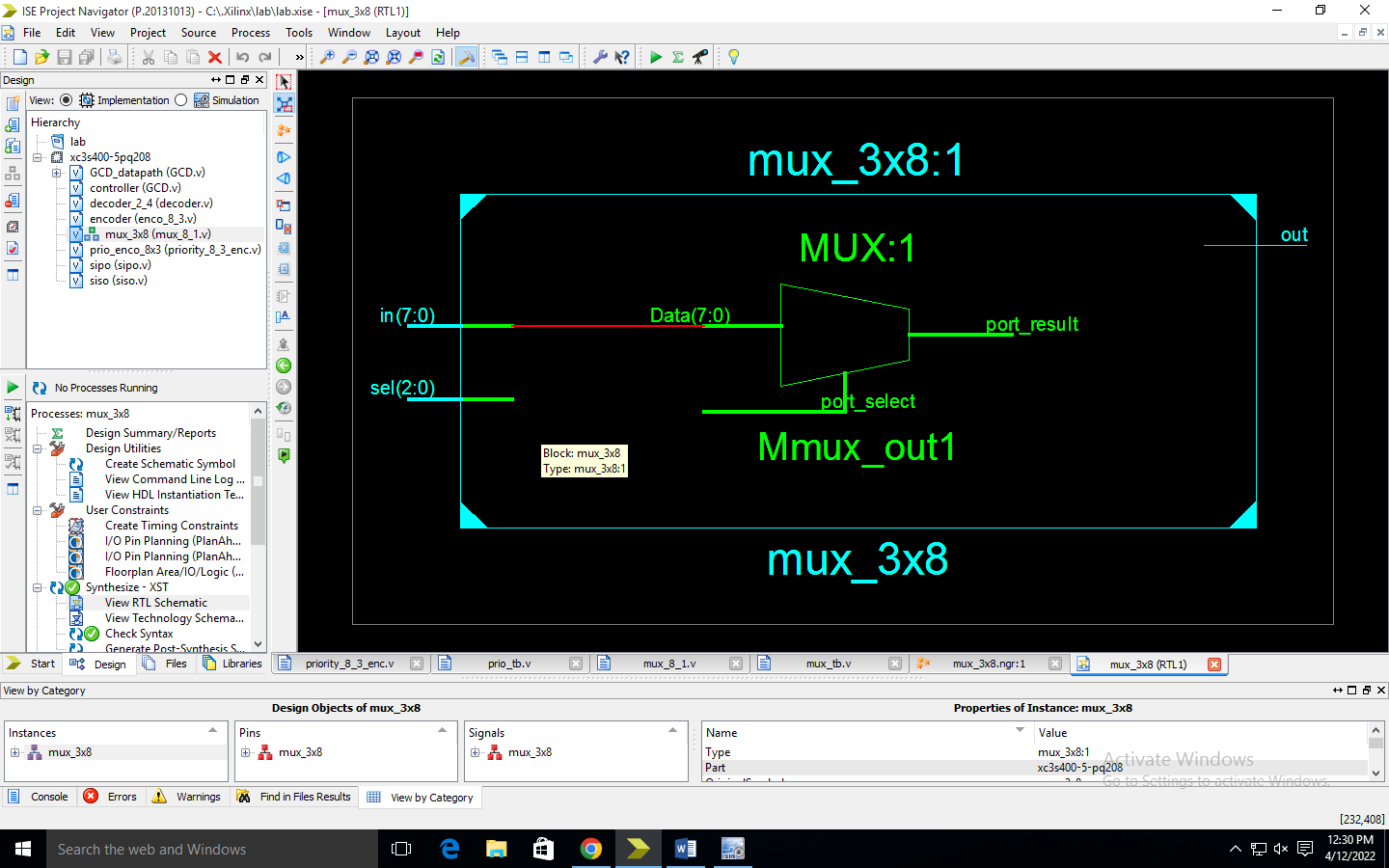
always #20 sel=sel+3'b001;

endmodule

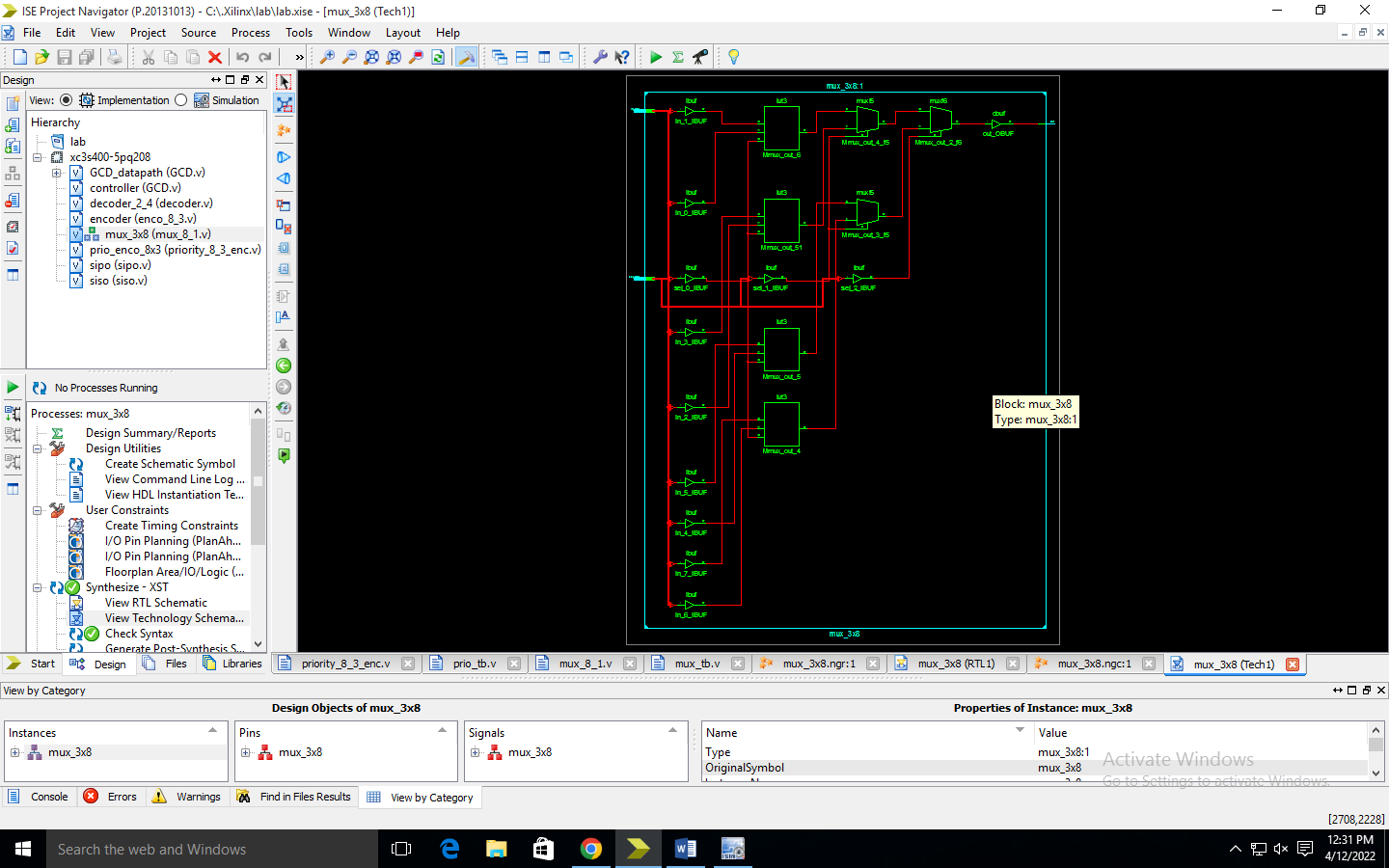
simulation



**Rtl schematic**



**Technologu schematic**



**4 bit binary to gray converter**

**module bin2gray**

**(input [3:0] bin, //binary input**

**output [3:0] G //gray code output**

**);**

**//xor gates.**

**assign G[3] = bin[3];**

**assign G[2] = bin[3] ^ bin[2];**

**assign G[1] = bin[2] ^ bin[1];**

**assign G[0] = bin[1] ^ bin[0];**

**endmodule**

testbench

module b\_g\_tb;

// Inputs

reg [3:0] bin;

// Outputs

wire [3:0] G;

// Instantiate the Unit Under Test (UUT)

bin2gray uut (

.bin(bin),

.G(G)

);

initial begin

bin <= 0; #10;

bin <= 1; #10;

bin <= 2; #10;

bin <= 3; #10;

bin <= 4; #10;

bin <= 5; #10;

bin <= 6; #10;

bin <= 7; #10;

bin <= 8; #10;

bin <= 9; #10;

bin <= 10; #10;

bin <= 11; #10;

bin <= 12; #10;

bin <= 13; #10;

bin <= 14; #10;

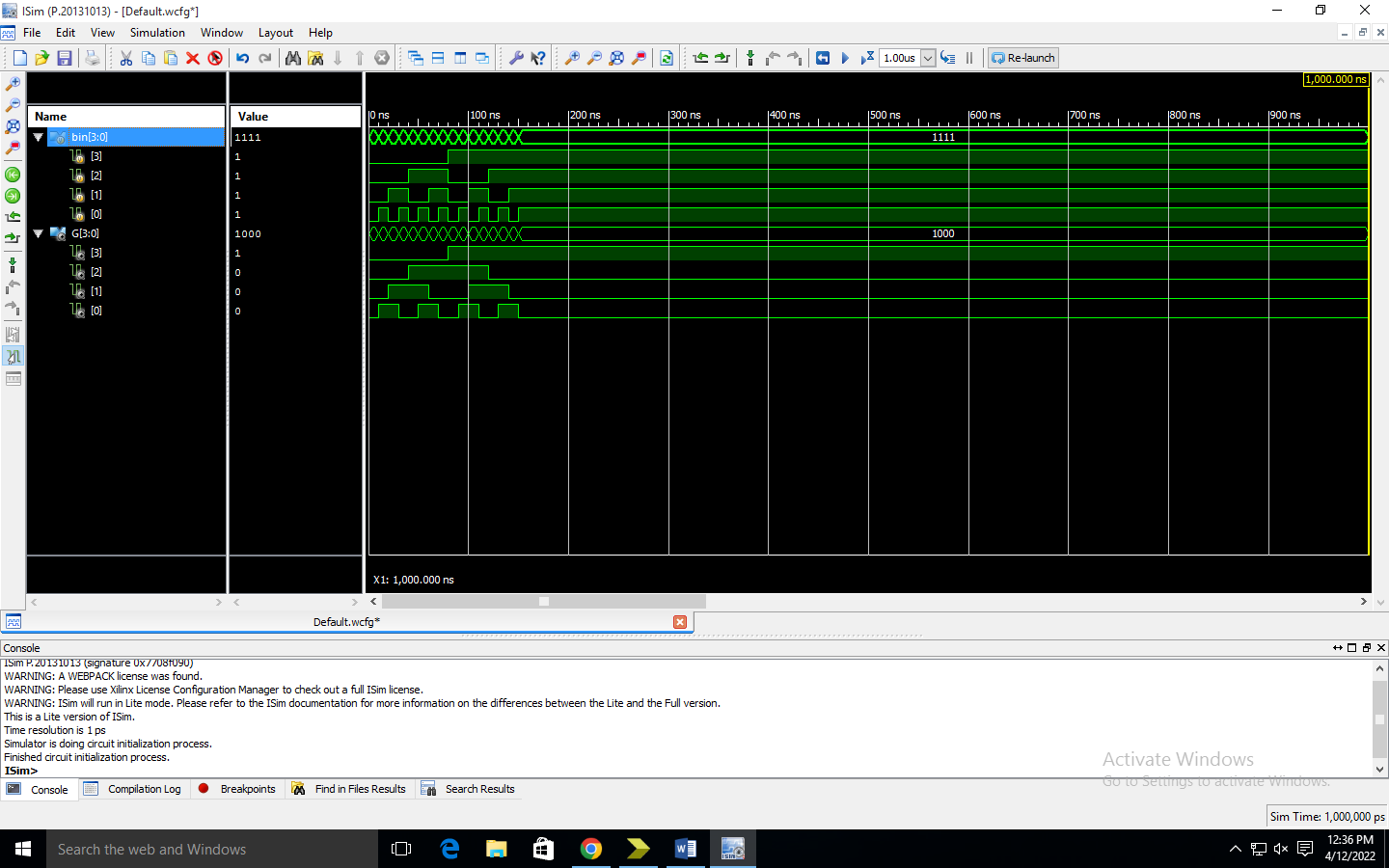
bin <= 15; #10;

#100;

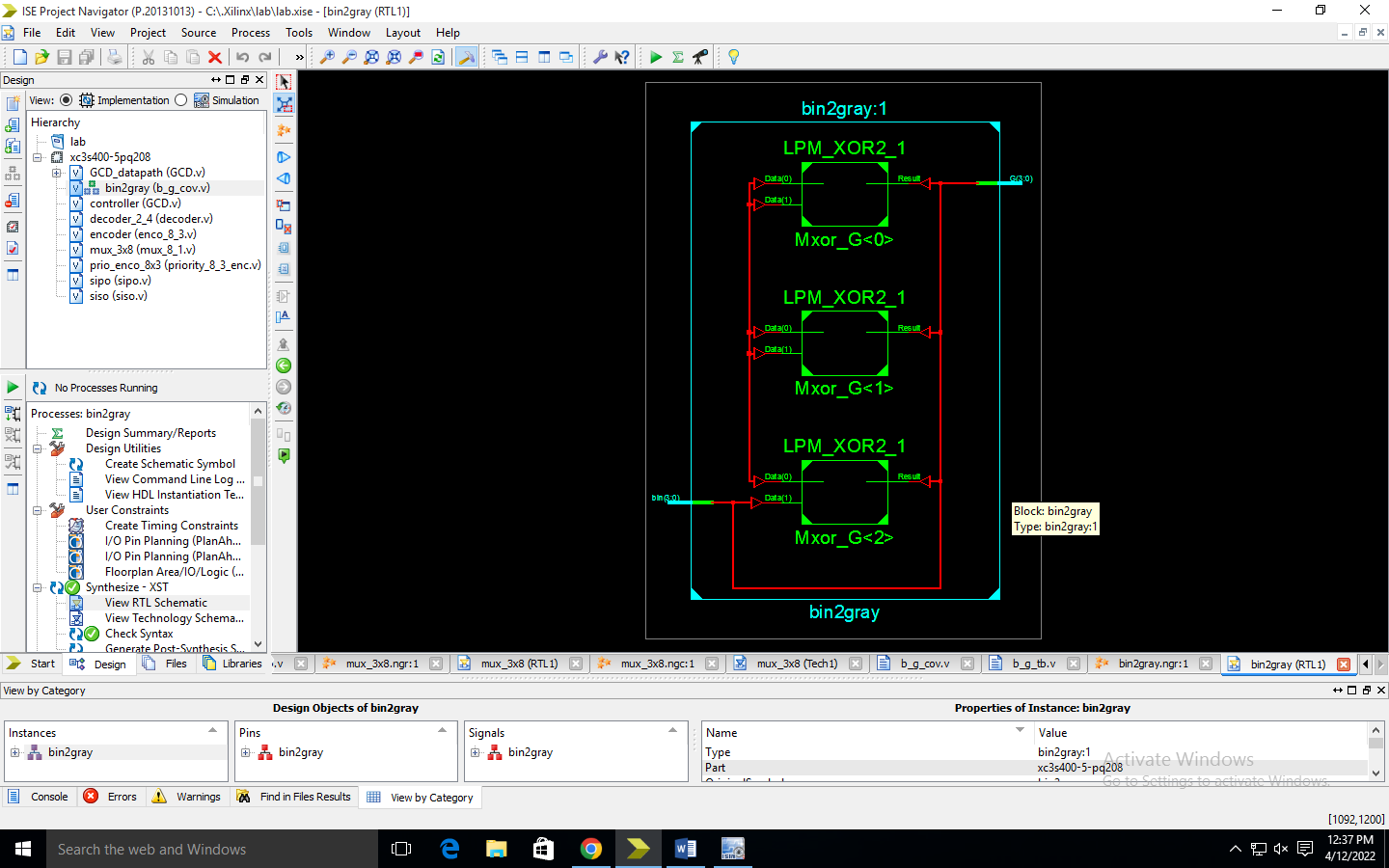
end

endmodule

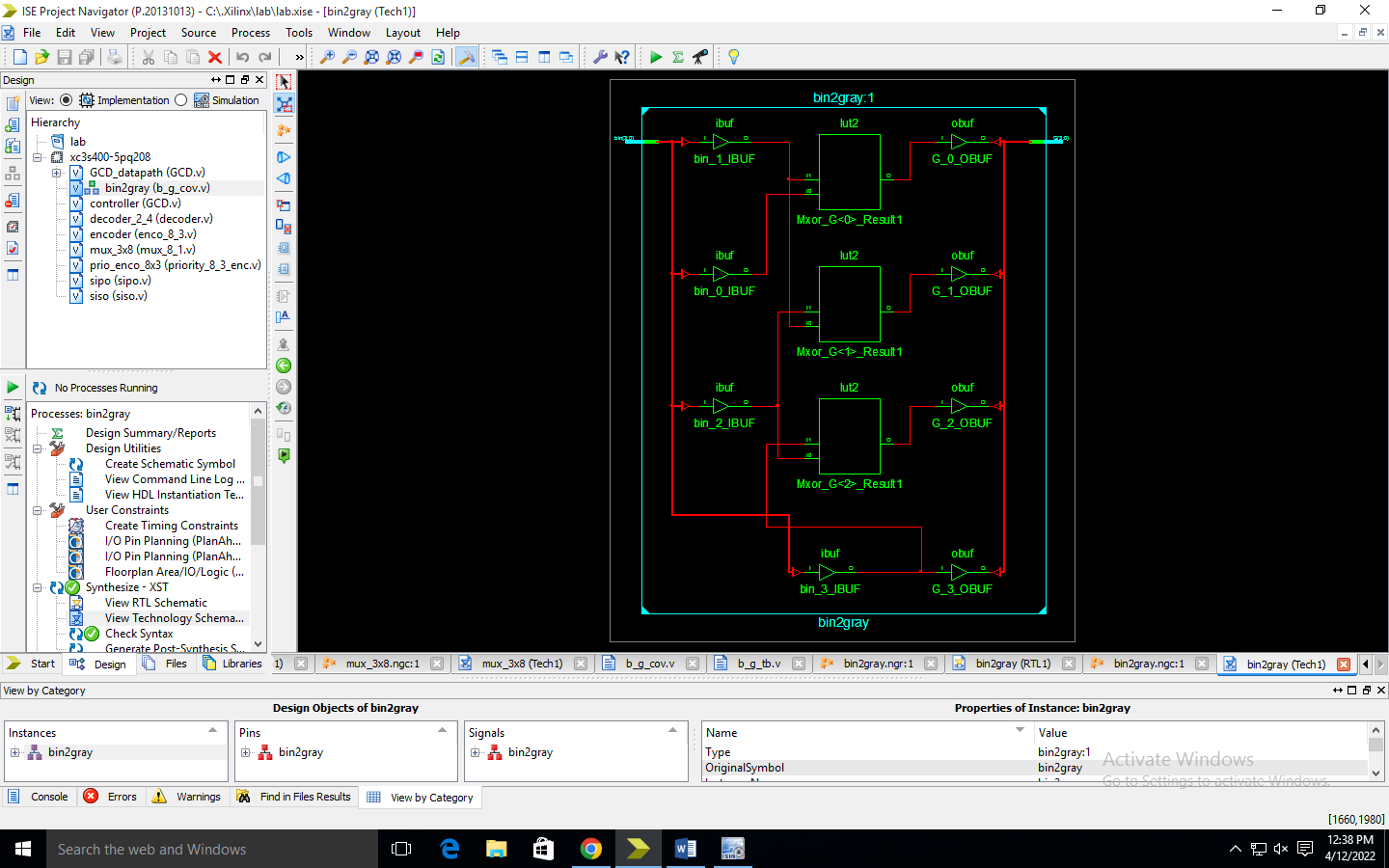
simulation



Rtl schematic



Technology schematic



2x1 mux

module mux2\_1(in1, in2, select, out);

// define input port

input in1, in2, select;

// define the output port

output out;

// assign one of the inputs to the output based upon select line input

assign out = select ? in2 : in1;

endmodule

testbench

module mux\_tb;

// Inputs

reg in1;

reg in2;

reg select;

// Outputs

wire out;

// Instantiate the Unit Under Test (UUT)

mux2\_1 uut (

.in1(in1),

.in2(in2),

.select(select),

.out(out)

);

initial begin

in1=1'b0;in2=1'b0;select=1'b0;

#10 in1=1'b1;

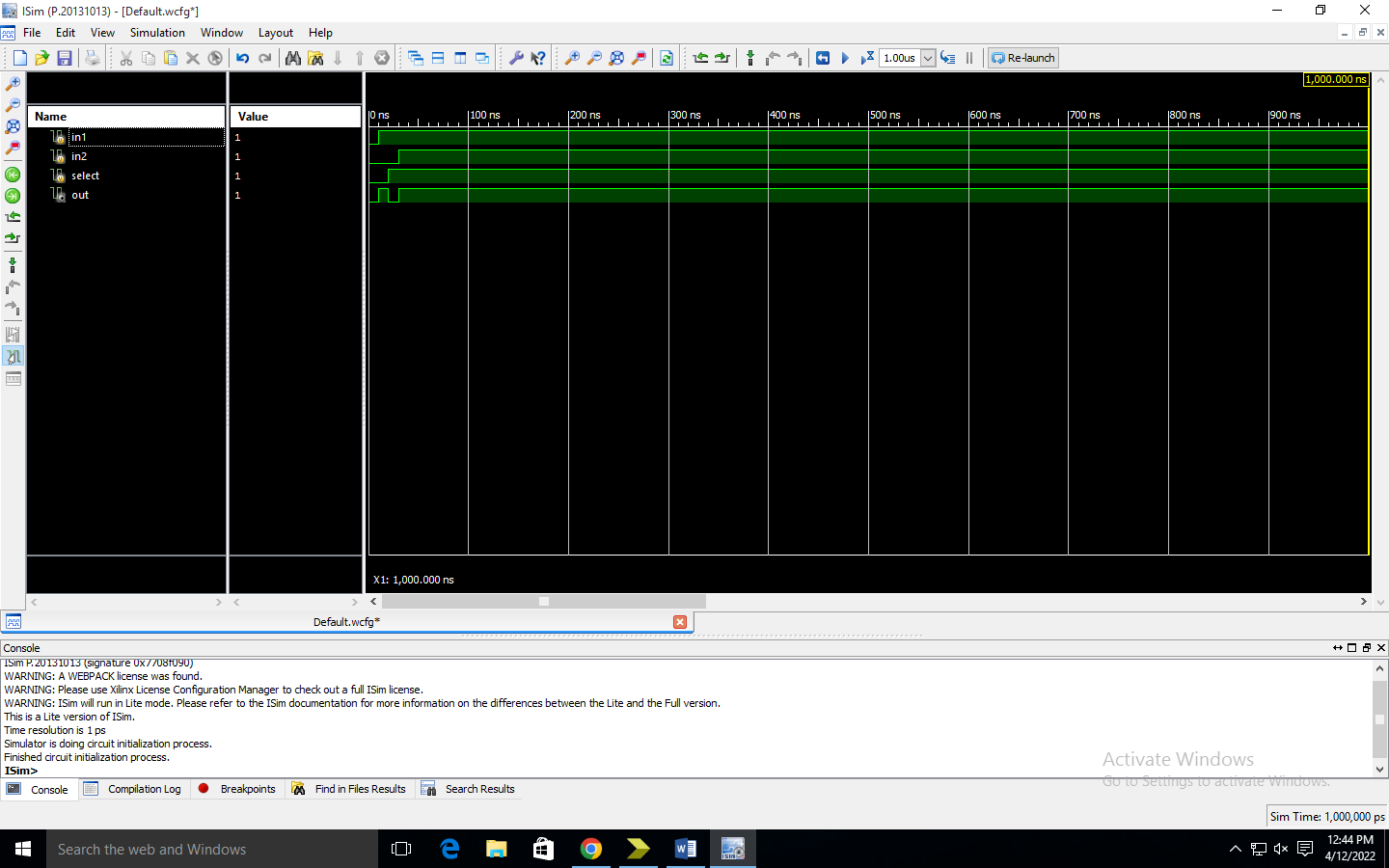
#10 select=1'b1;

#10 in2=1'b1;

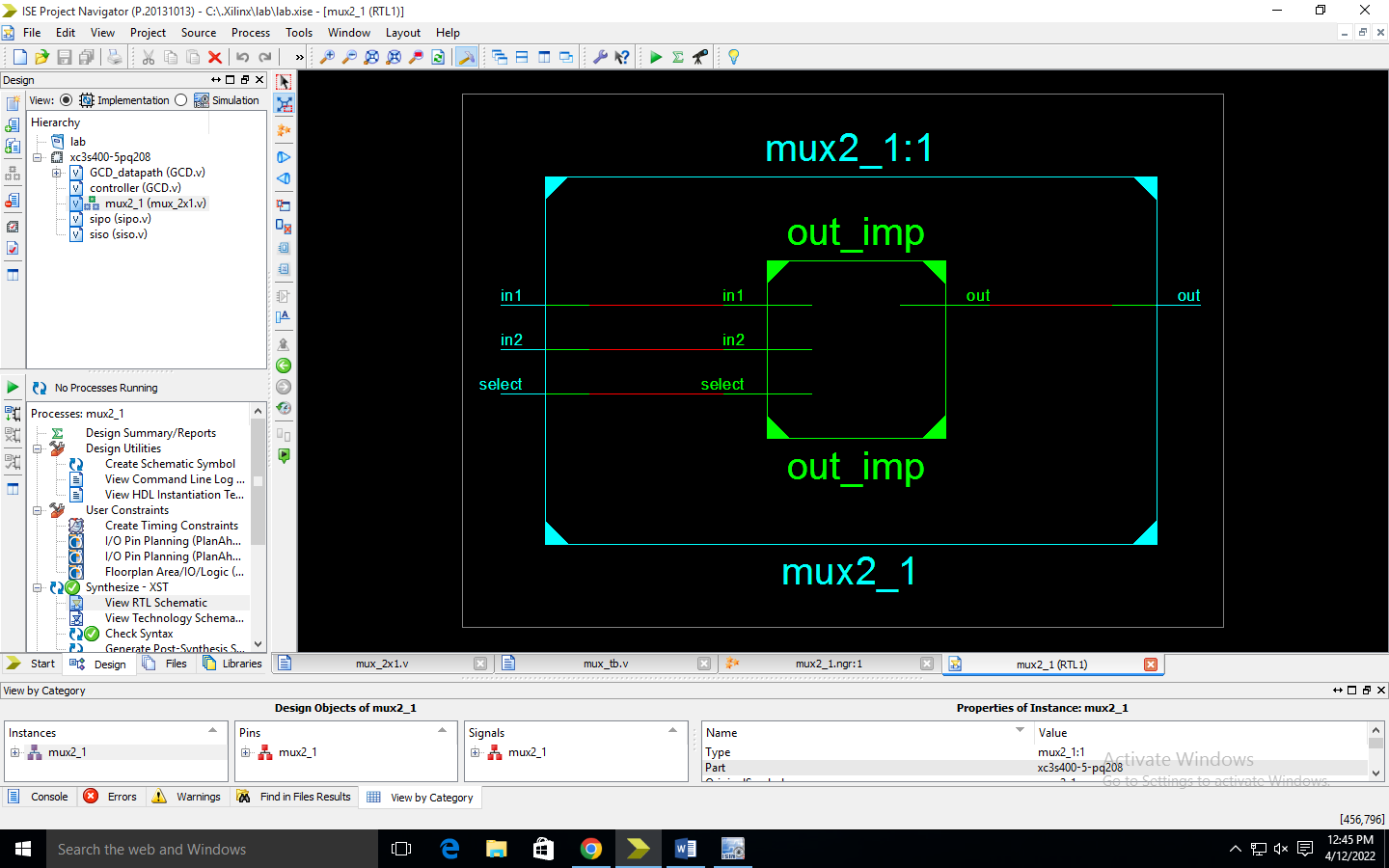
end

endmodule

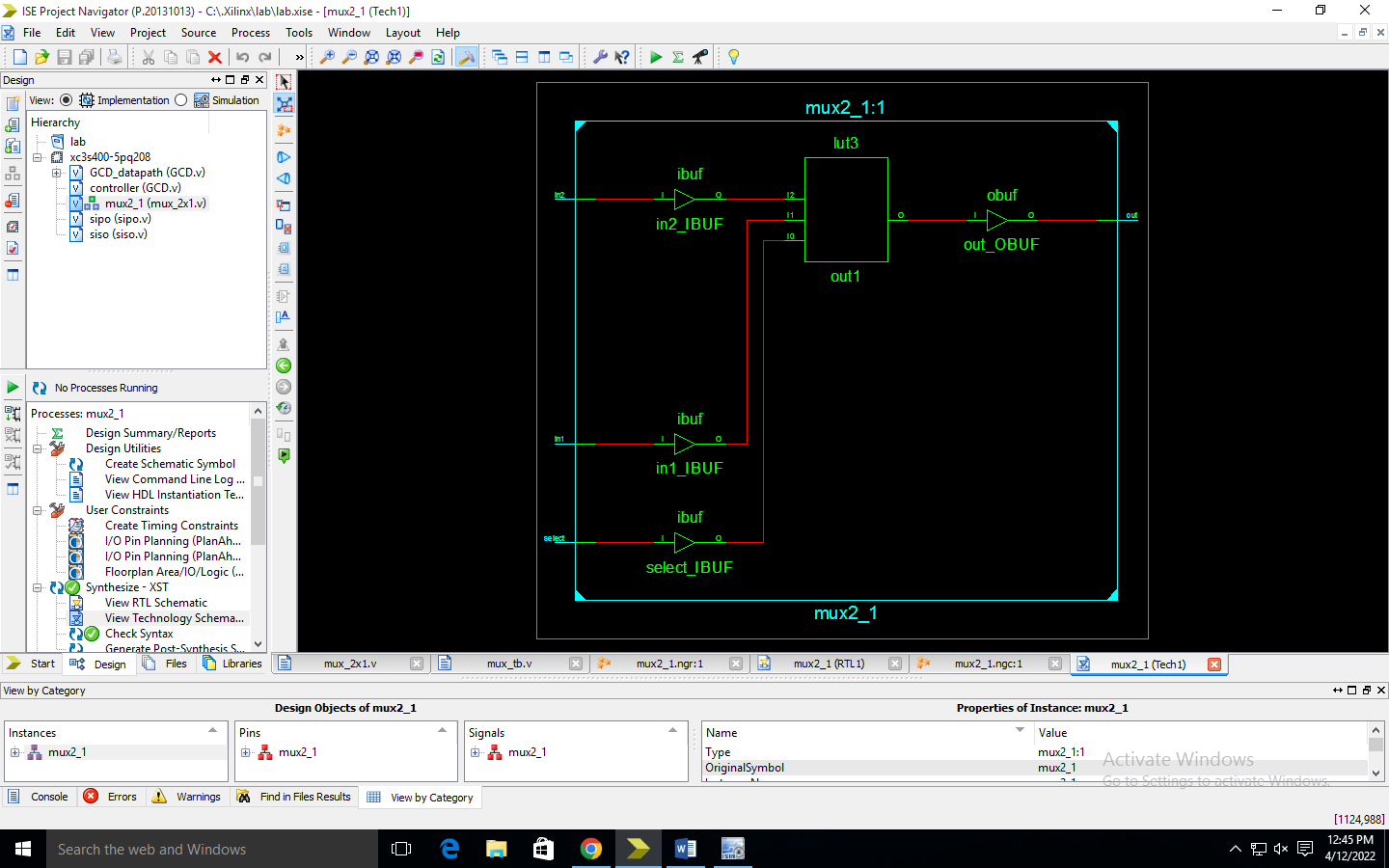
simulation



Rtl schematic



Technology



4- siso

module siso(input clk,clr,d, output v);

reg [3:0]q;

always@(posedge clk)

if(clr==1)

begin

q<=4'b0000;

end

else

begin

q<=q>>1;

q[3]<=d;

end

assign v=q[0];

endmodule

testbench

module siso\_tb;

// Inputs

reg clk;

reg clr;

reg d;

// Outputs

wire v;

// Instantiate the Unit Under Test (UUT)

siso uut (

.clk(clk),

.clr(clr),

.d(d),

.v(v)

);

initial begin

// Initialize Inputs

clk = 0;

clr = 0;

d = 0;

#10 clr=0;

end

always #1 clk=~clk;

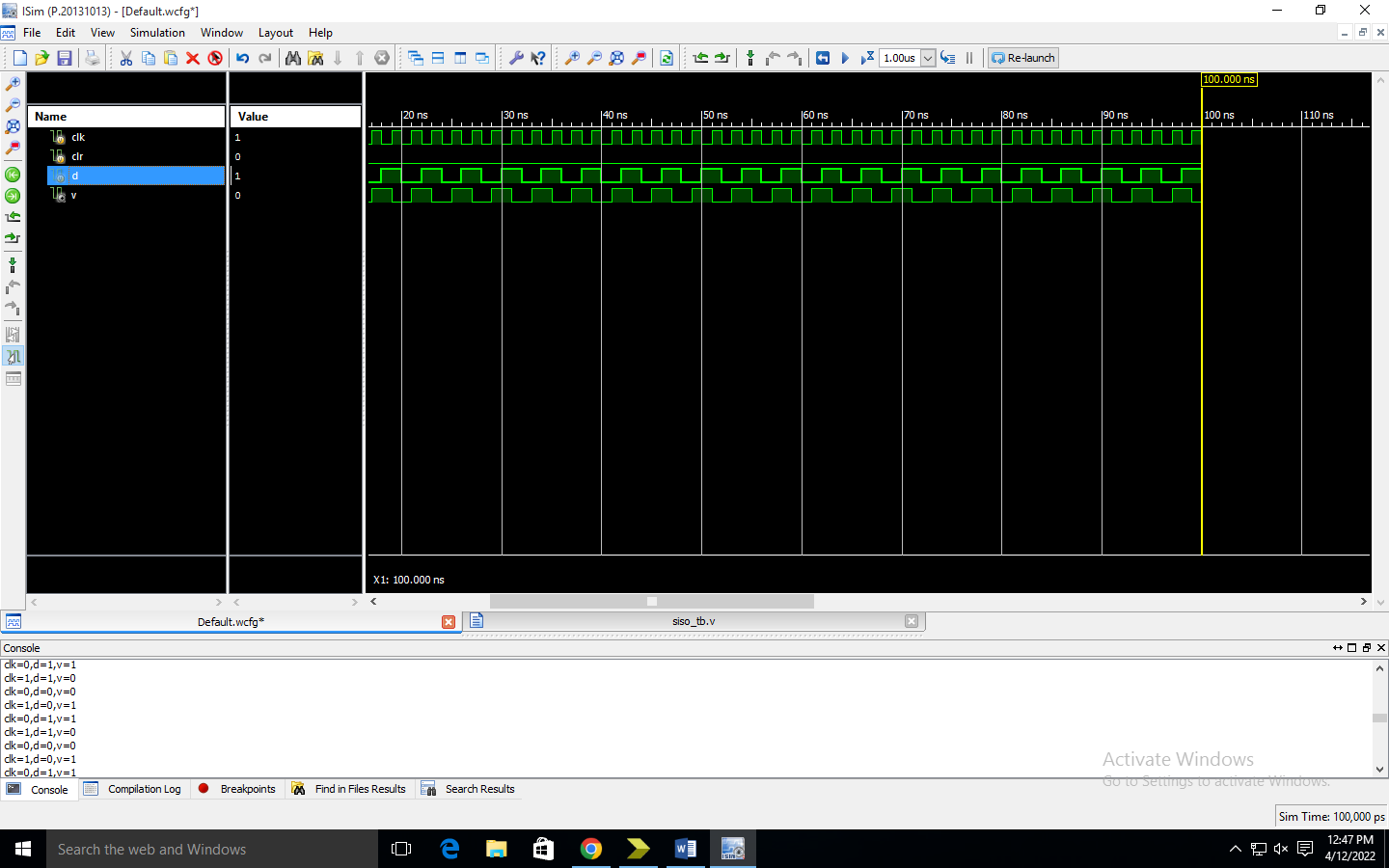
always #2 d=~d;

initial $monitor("clk=%b,d=%b,v=%b",clk,d,v);

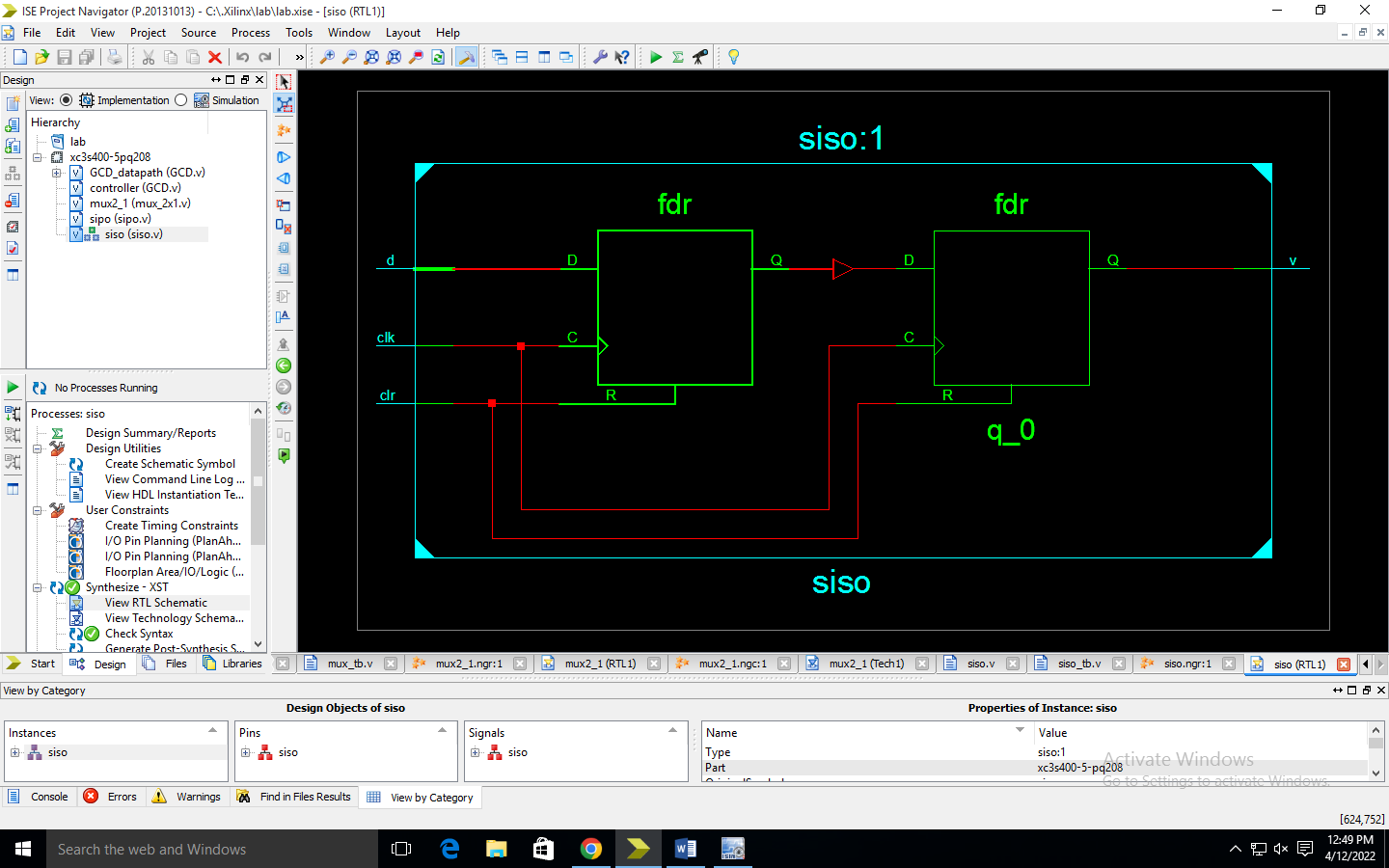
initial #100 $finish;

endmodule

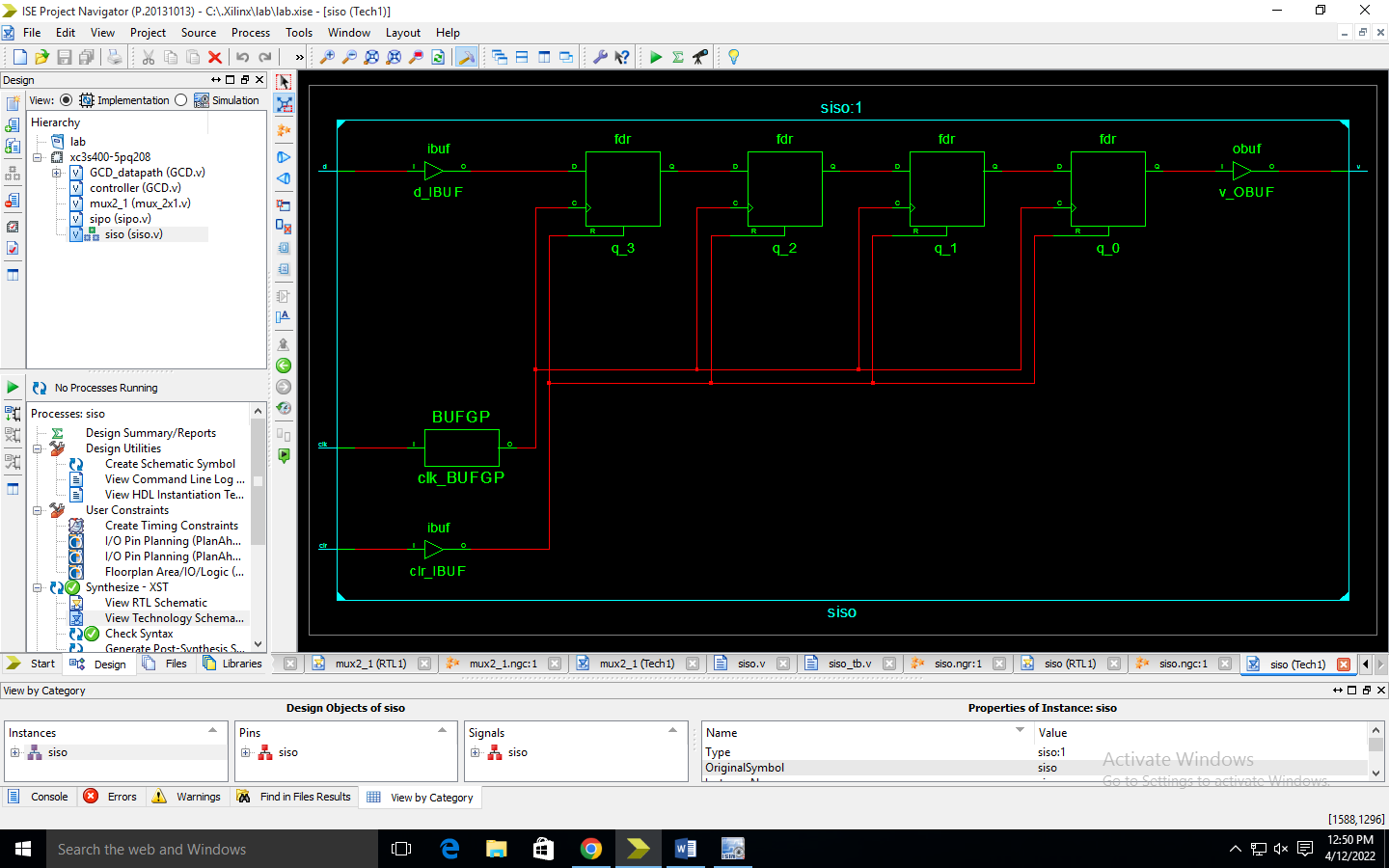
simulation



Rtl



Tech



4-sipo

module sipomod(clk,clear, si, po);

input clk, si,clear;

output [3:0] po;

reg [3:0] tmp;

reg [3:0] po;

always @(posedge clk)

begin

if (clear)

tmp <= 4'b0000;

else

tmp <= tmp << 1;

tmp[0] <= si;

po = tmp;

end

endmodule

testbench

module sipot\_b;

reg clk;

reg clear;

reg si;

wire [3:0] po;

sipomod uut (.clk(clk),.clear(clear), .si(si),.po(po) );

initial begin

clk = 0;

clear = 0;

si = 0;

#5 clear=1'b1;

#5 clear=1'b0;

#10 si=1'b1;

#10 si=1'b0;

#10 si=1'b0;

#10 si=1'b1;

#10 si=1'b0;

#10 si=1'bx;

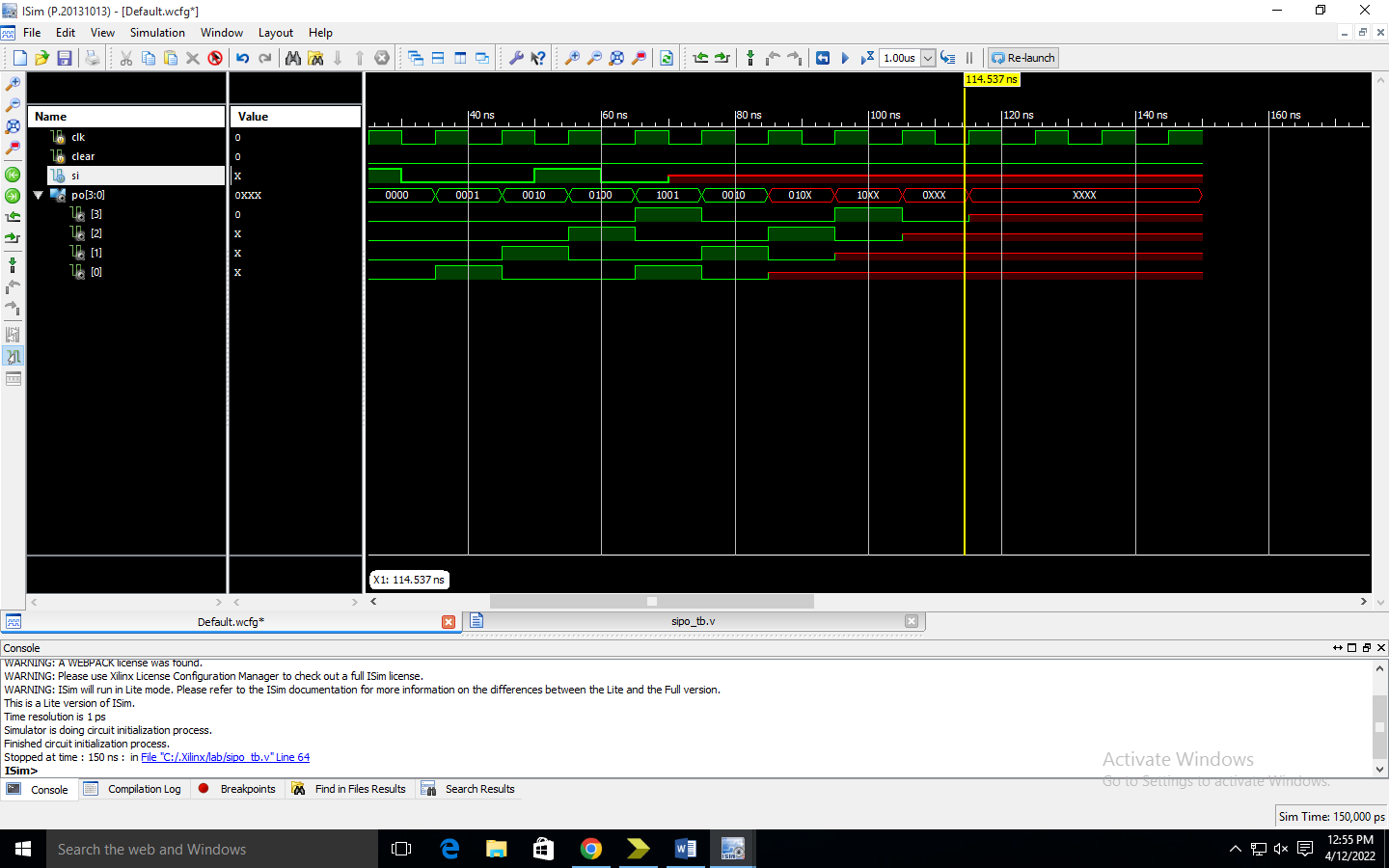
end

always #5 clk = ~clk;

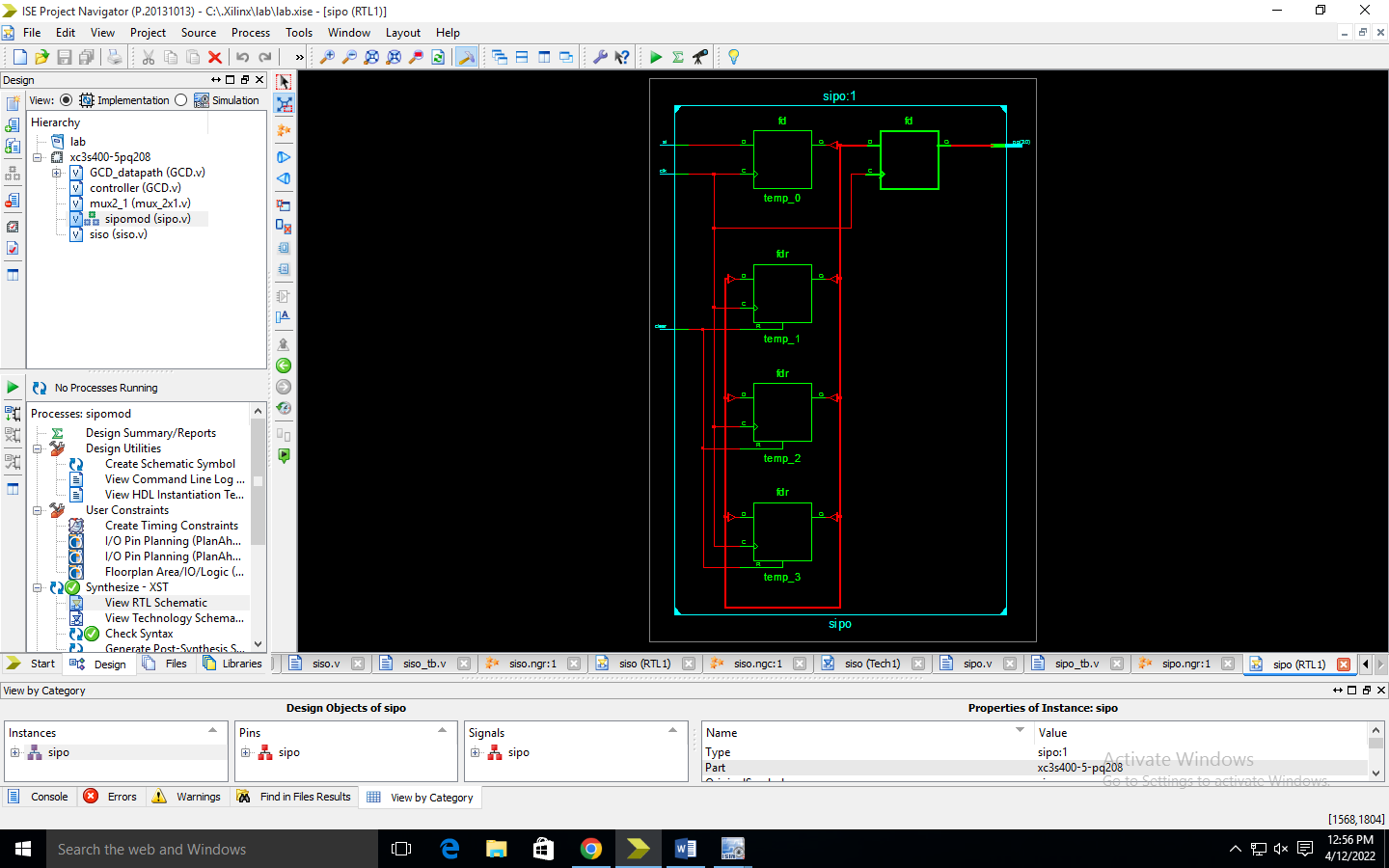
initial #150 $stop;

endmodule

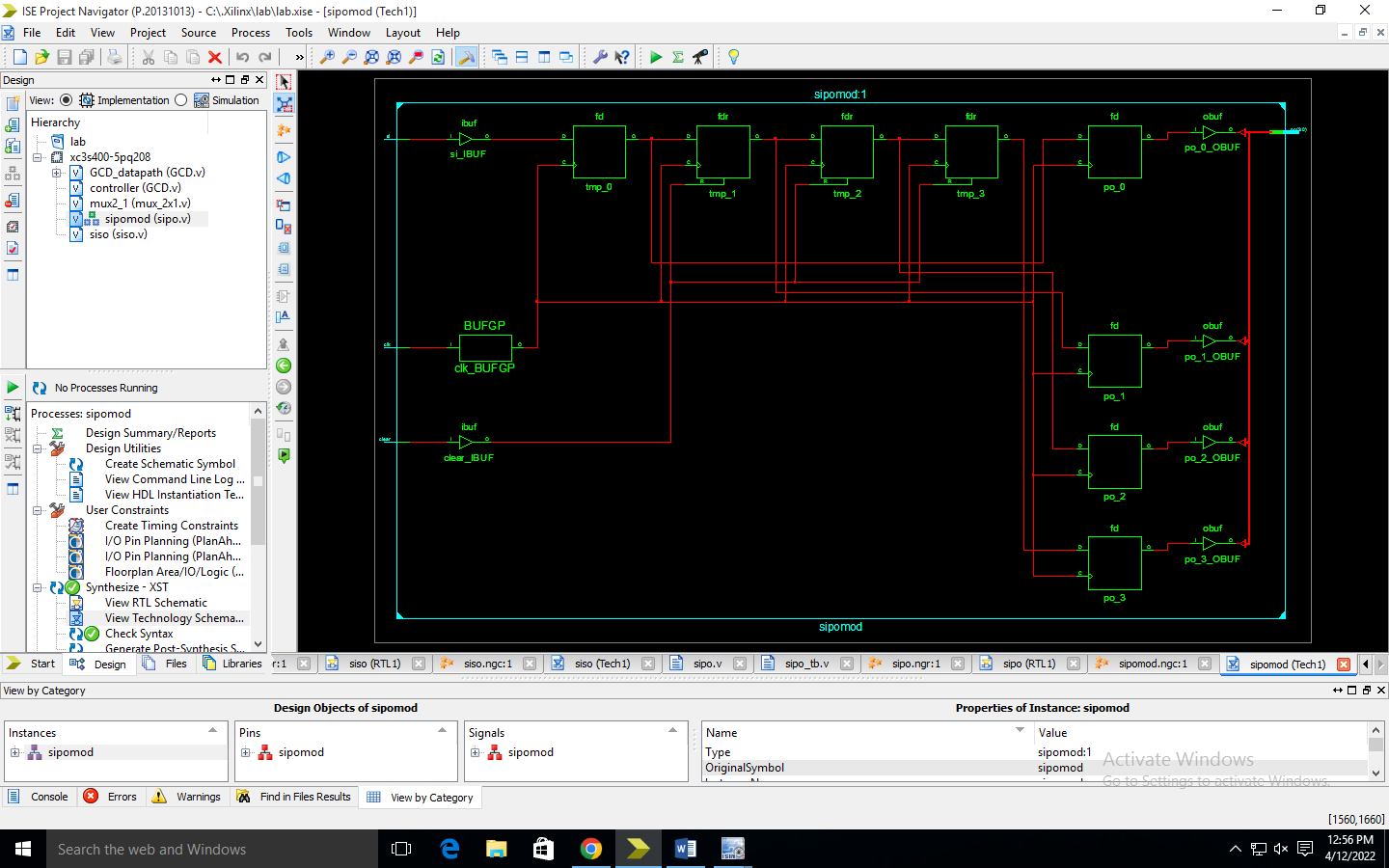
simulation



Rtl



Tech



4-piso

module piso(clk,rst,a,q);

input clk,rst;

input [3:0]a;

output q;

reg q;

reg [3:0]temp;

always@(posedge clk,posedge rst)

begin

if(rst==1'b1)

begin

q<=1'b0;

temp<=a;

end

else

begin

q<=temp[0];

temp <= temp>>1'b1;

end

end

endmodule

testbench

module piso\_tb;

// Inputs

reg clk;

reg rst;

reg [3:0] a;

// Outputs

wire q;

// Instantiate the Unit Under Test (UUT)

piso uut (

.clk(clk),

.rst(rst),

.a(a),

.q(q)

);

initial begin

rst=1'b1; a=4'b1101;

#300 rst=1'b0;

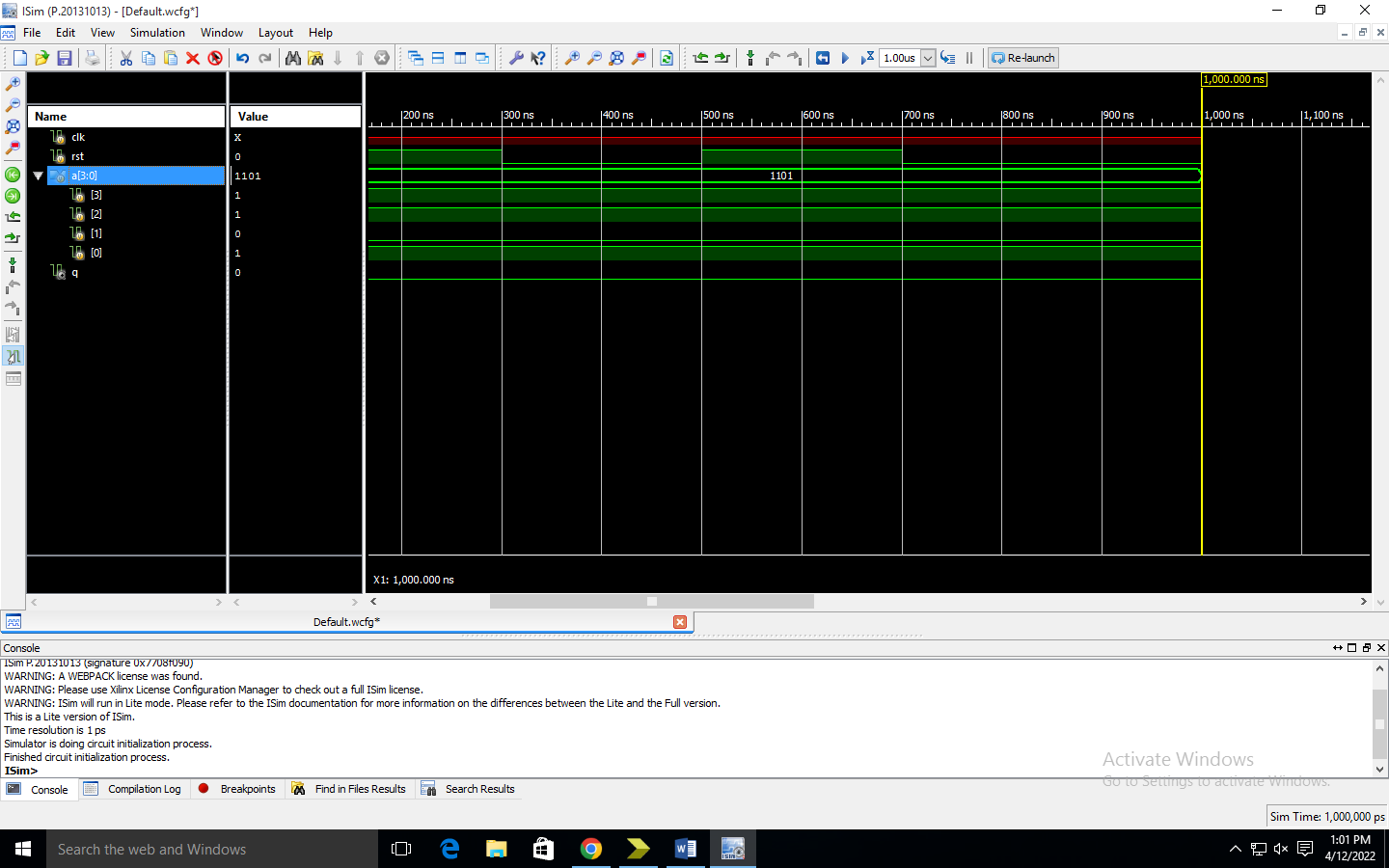
#200 rst=1'b1;

#200 rst=1'b0;

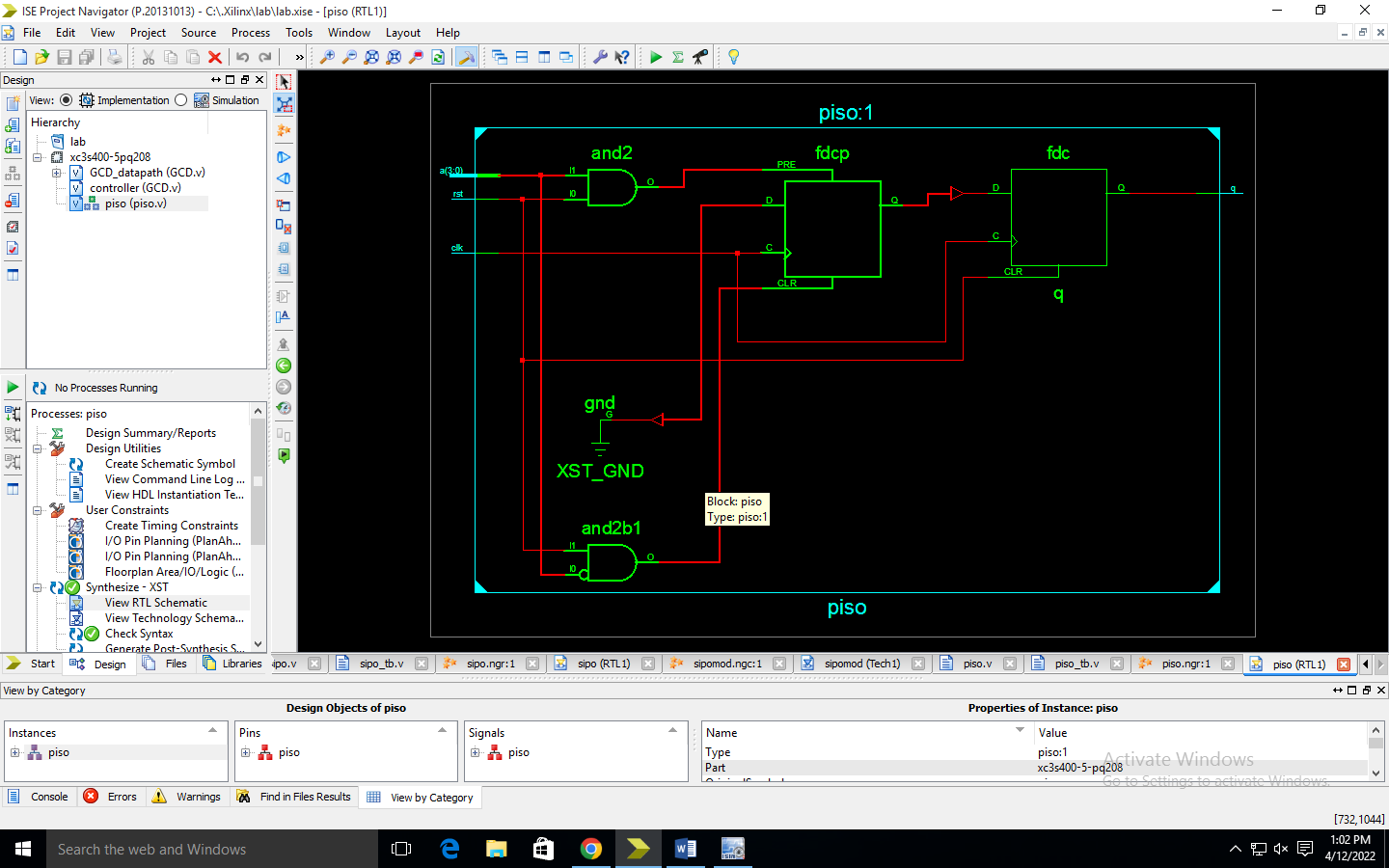
end

endmodule

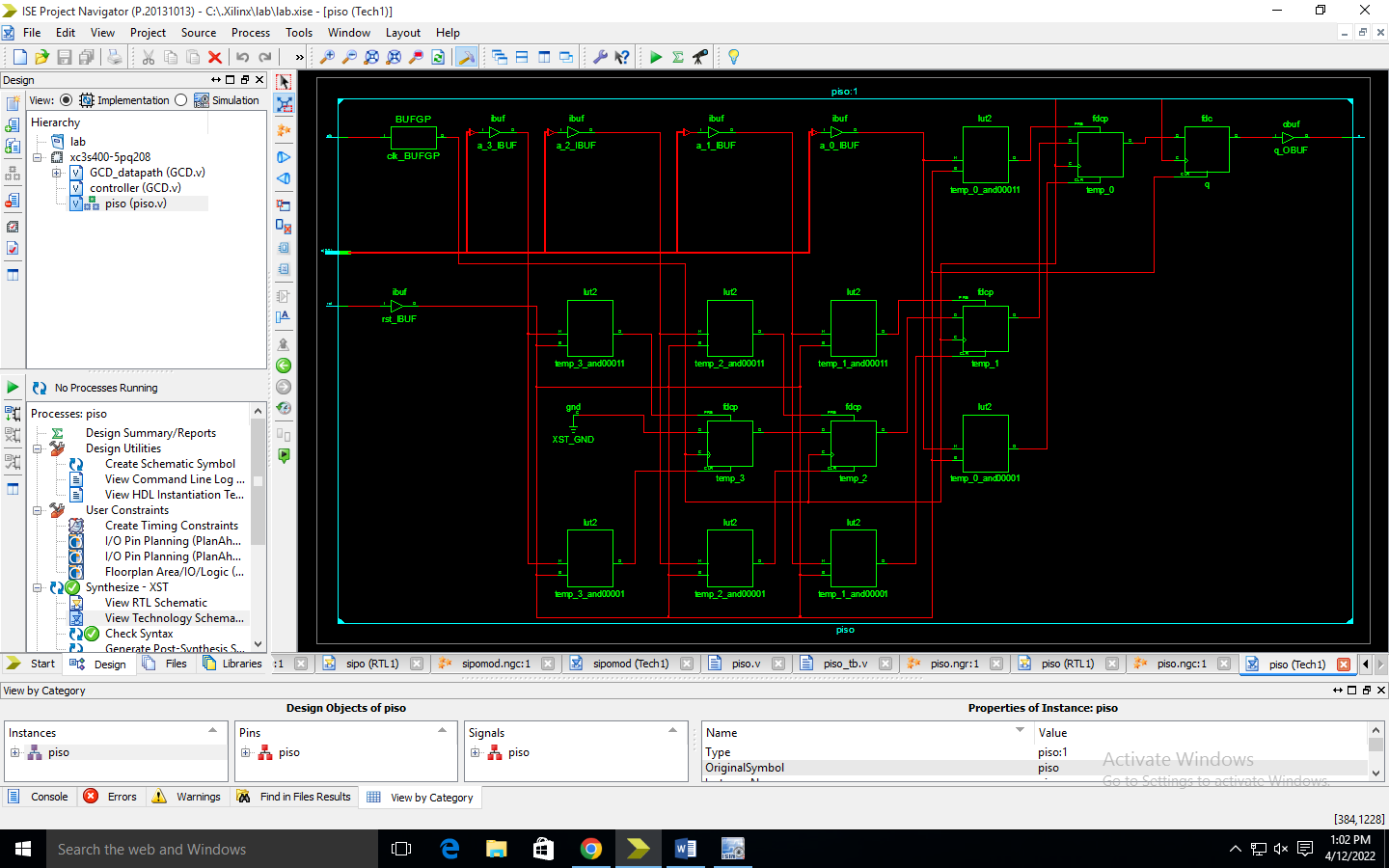
simulation



Rtl



Tech



Pipo

module pipo(clk,rst,a,q);

input clk,rst;

input[3:0]a;

output[3:0]q;

reg[3:0]q;

always@(posedge clk,posedge rst)

begin

if (rst==1'b1)

q<=4'b0000;

else

q<=a;

end

endmodule

testbench

module pipo\_tb;

// Inputs

reg clk;

reg rst;

reg [3:0] a;

// Outputs

wire [3:0] q;

// Instantiate the Unit Under Test (UUT)

pipo uut (

.clk(clk),

.rst(rst),

.a(a),

.q(q)

);

initial begin

a=4'b1101;rst=1'b1;

#100 rst=1'b0;

#100 a=4'b1000;

#100 rst=1'b1;

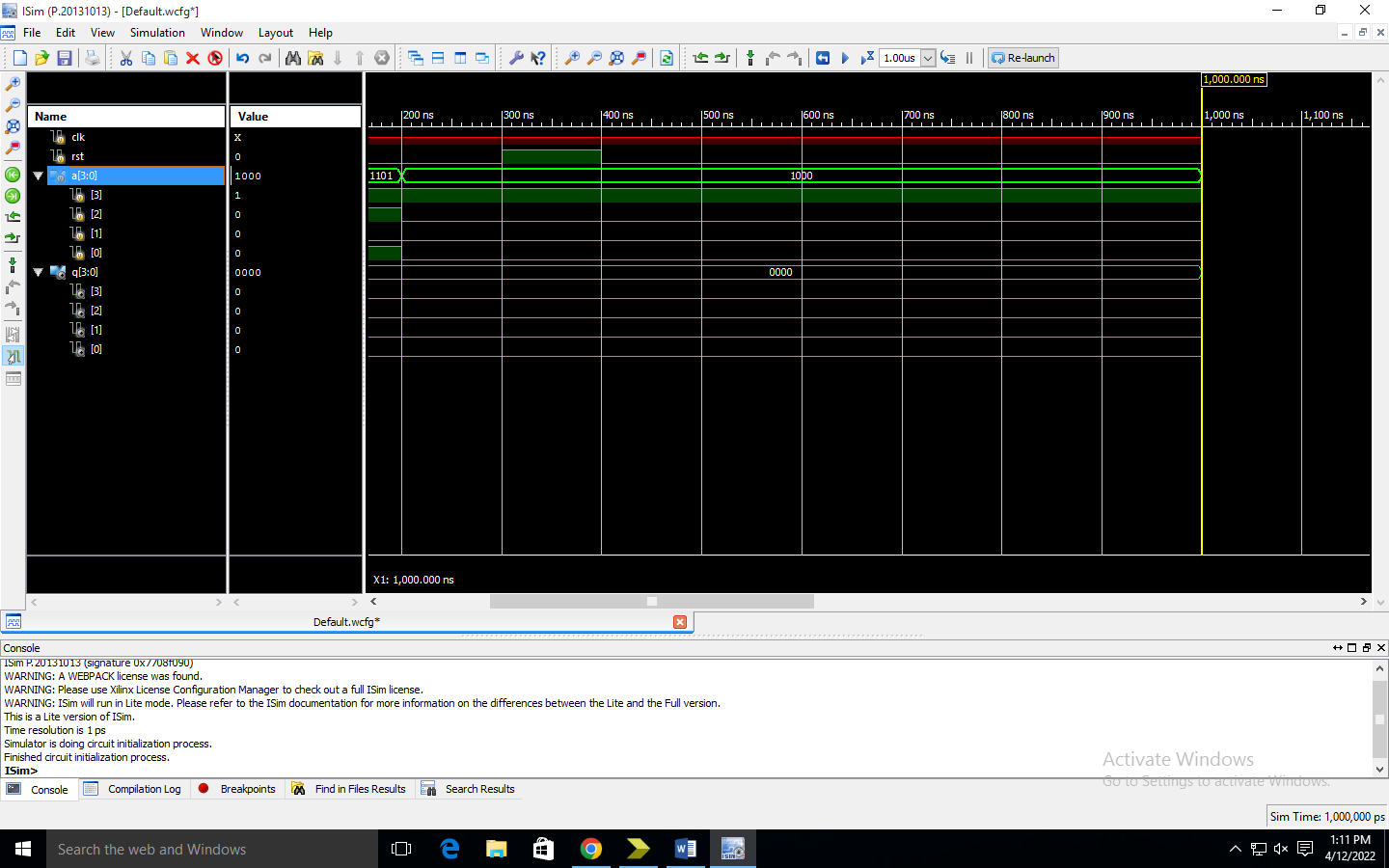
#100 rst=1'b0;

// Add stimulus here

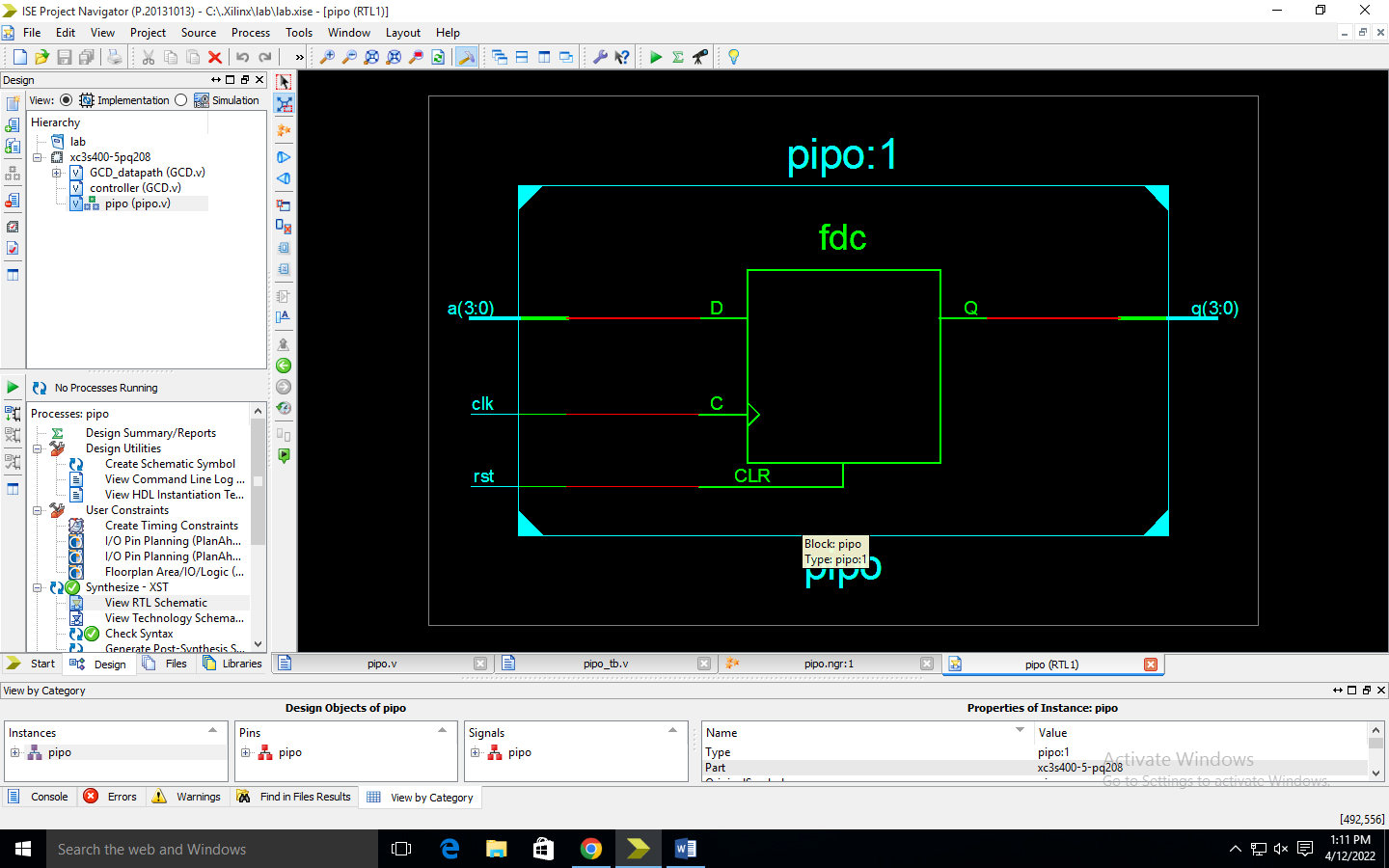
end

endmodule

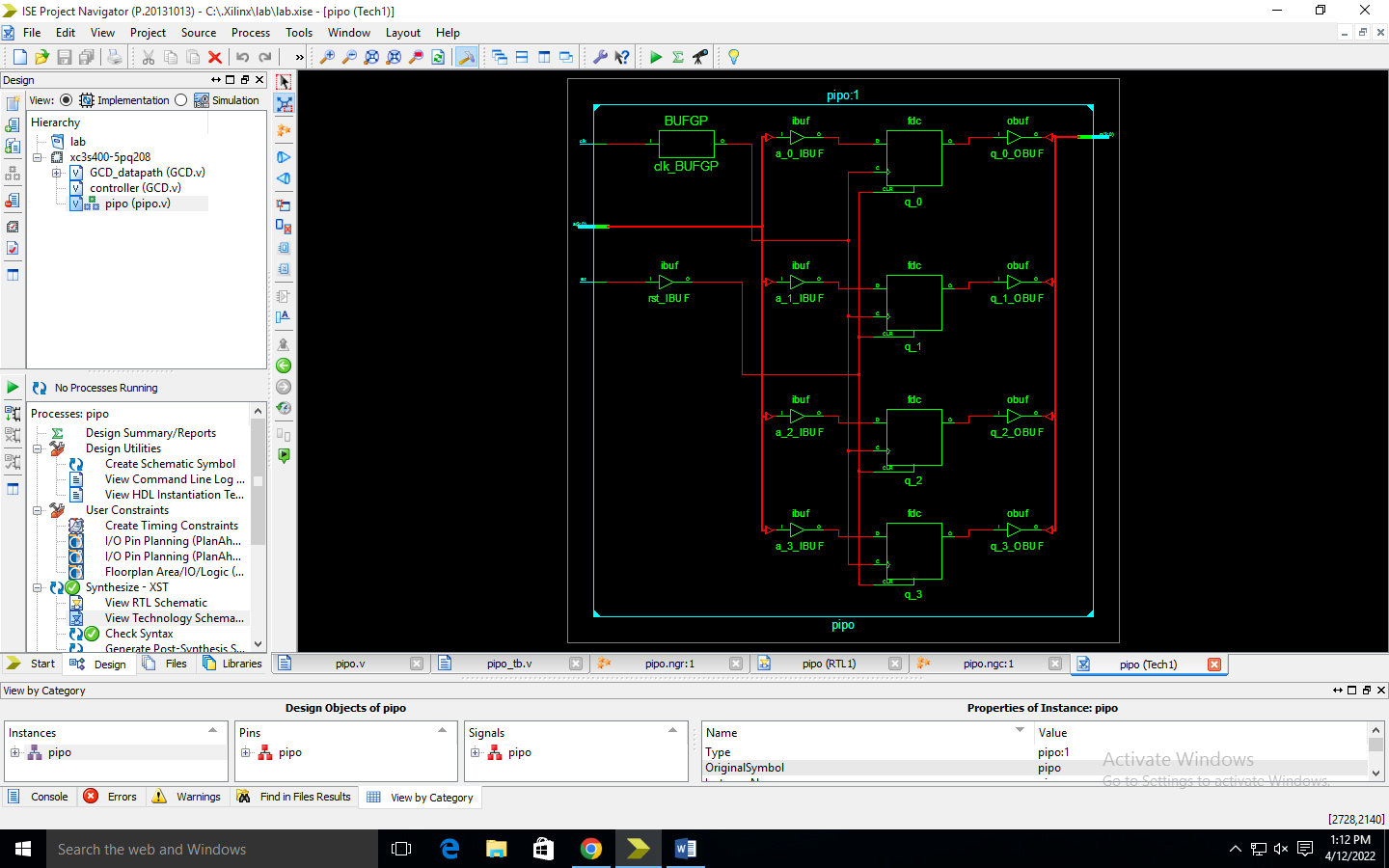
simulation



Rtl



Tech



GCD

module GCD\_datapath(gt,lt,eq,ldA,ldB,sel1,sel2,sel\_in,data\_in,clk);

input ldA,ldB,sel1,sel2,sel\_in,clk;

input [15:0] data\_in;

output gt,lt,eq;

wire[15:0] Aout,Bout,X,Y,Bus,SubOut;

PIPO A(Aout,Bus,ldA,clk);

PIPO B(Bout,Bus,ldB,clk);

MUX MUX\_in1(X,Aout,Bout,sel1);

MUX MUX\_in2(Y,Aout,Bout,sel2);

MUX MUX\_load(Bus,SubOut,data\_in,sel\_in);

SUB SB(SubOut,X,Y);

COMPARE COMP(lt,gt,eq,Aout,Bout);

endmodule

module PIPO(data\_out,data\_in,load,clk);

input[15:0] data\_in;

input load,clk;

output reg[15:0] data\_out;

always@(posedge clk)

if(load)

data\_out<=data\_in;

endmodule

module SUB(out,in1,in2);

input[15:0] in1,in2;

output reg[15:0] out;

always@(\*)

out=in1-in2;

endmodule

module COMPARE(lt,gt,eq,data1,data2);

input[15:0] data1,data2;

output lt,gt,eq;

assign lt=data1<data2;

assign gt=data1>data2;

assign eq=data1==data2;

endmodule

module MUX(out,in0,in1,sel);

input[15:0] in0,in1;

input sel;

output[15:0] out;

assign out = sel?in1:in0;

endmodule

module controller(ldA,ldB,sel1,sel2,sel\_in,done,clk,lt,gt,eq,start);

input clk,lt,gt,eq,start;

output reg ldA,ldB,sel1,sel2,sel\_in,done;

reg[2:0] state;

parameter S0=3'b000,S1=3'b001,S2=3'b010,S3=3'b011,S4=3'b100,S5=3'b101;

always@(posedge clk)

begin

case(state)

S0: if(start) state<=S1;

S1: state<=S2;

S2: #2 if(eq) state <=S5;

else if(lt) state<=S3;

else if(gt) state<=S4;

S3: #2 if(eq) state <=S5;

else if(lt) state<=S3;

else if(gt) state<=S4;

S4: #2 if(eq) state <=S5;

else if(lt) state<=S3;

else if(gt) state<=S4;

S5: state<=S5;

default: state<=S0;

endcase

end

always@(state)

begin

case(state)

S0: begin sel\_in=1;ldA=1;ldB=0;done=0;end

S1: begin sel\_in=1;ldA=01;ldB=1;end

S2: if(eq) done=1;

else if(lt) begin

sel1=1; sel2=0; sel\_in=0;

#1 ldA=1; ldB=0;

end

S3: if(eq) done=1;

else if(lt) begin

sel1=1; sel2=0; sel\_in=0;

#1 ldA=0; ldB=1;

end

else if(gt) begin

sel1=0; sel2=1; sel\_in=0;

#1 ldA=1; ldB=0;

end

S4: if(eq) done=1;

else if(lt) begin

sel1=1; sel2=0; sel\_in=0;

#1 ldA=0; ldB=1;

end

else if(gt) begin

sel1=0; sel2=1; sel\_in=0;

#1 ldA=1; ldB=0;

end

S5: begin

done=1; sel1=0; sel2=0; ldA=0;

ldB=0;

end

default: begin ldA=0; ldB=0; end

endcase

end

endmodule

testbench