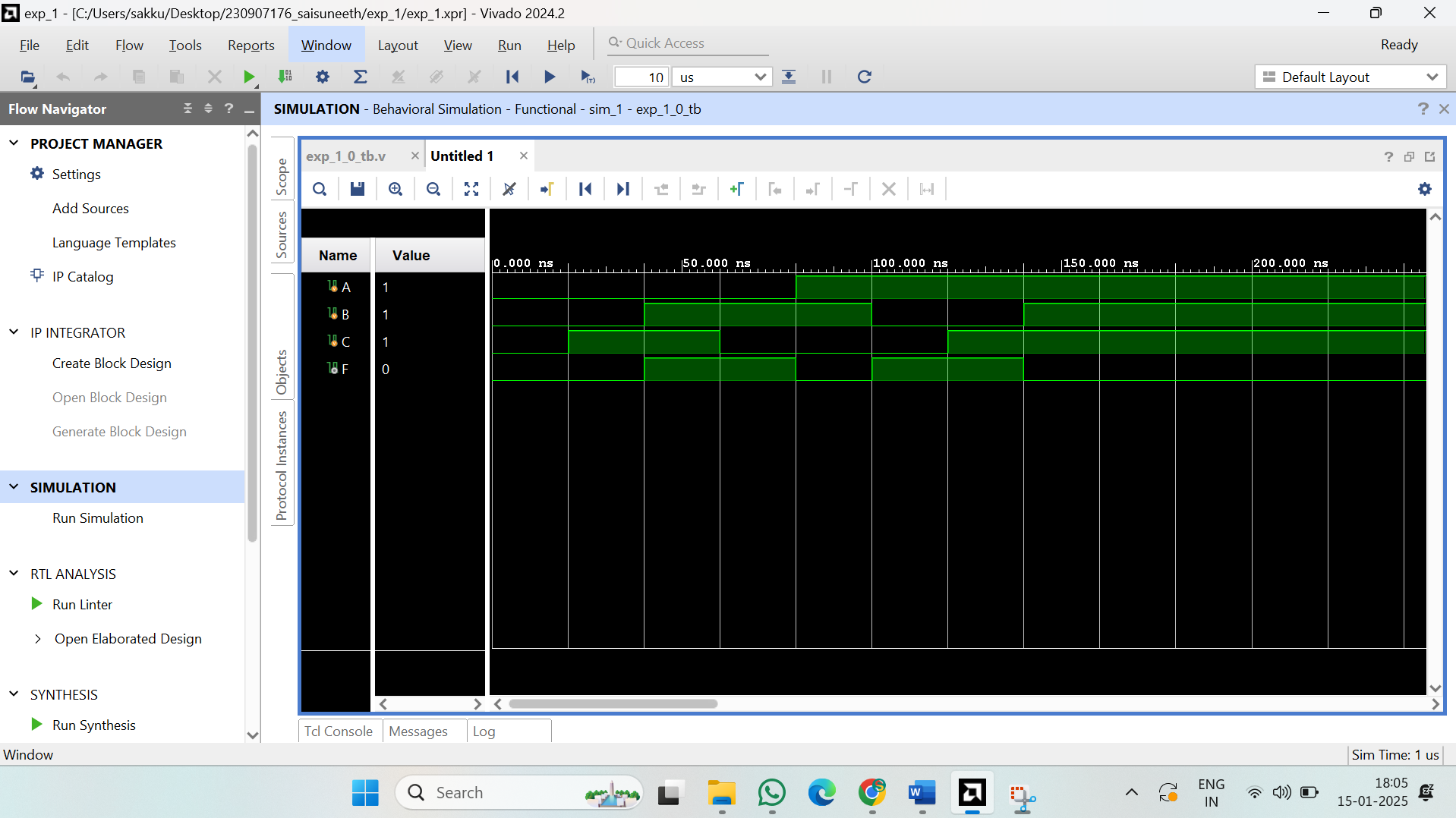
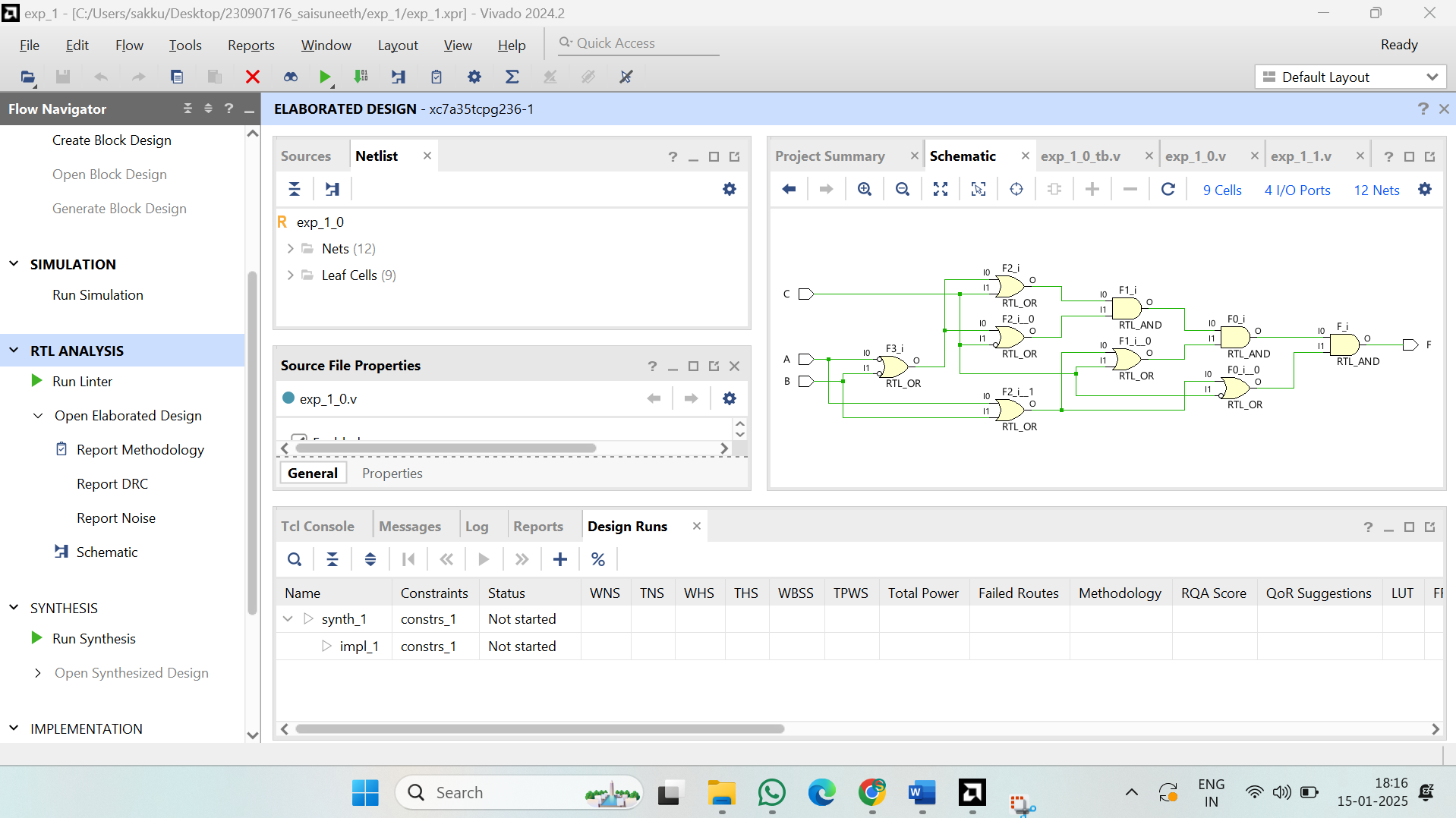
**Exp-1**

**Examples**

**Exp 1.0**

****

****

**Verilog code:**

module exp\_1\_0(

input A,B,C,

output F

);

wire Abar,Bbar,Cbar;

assign Abar=~A;

assign Bbar=~B;

assign Cbar=~C;

assign F=((Abar|Bbar|C)&(Abar|Bbar|Cbar)&(A|B|C)&(A|B|Cbar));

endmodule

**Test bench:**

module exp\_1\_0\_tb;

reg A;

reg B;

reg C;

wire F;

exp\_1\_0 label (

.A(A),

.B(B),

.C(C),

.F(F)

);

initial

begin

A =0;

B =0;

C =0;

#20; C =1;

#20; B =1;

#20; C=0;

#20; A =1;

#20; B=0;

#20; C=1;

#20; B=1;

#40;

end

endmodule

**Exp 1.1**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Verilog code:**

module decoder(

input A,B,E,

output D0,D1,D2,D3

);

assign D0= A|B|E;

assign D1= A|~B|E;

assign D2= ~A|B|E;

assign D3= ~A|~B|E;

endmodule

**Test bench:**

module exp\_1\_1\_tb;

reg A;

reg B;

reg E;

wire D0;

wire D1;

wire D2;

wire D3;

decoder d1 (

.A(A),

.B(B),

.E(E),

.D0(D0),

.D1(D1),

.D2(D2),

.D3(D3)

);

initial begin

A = 0;B = 0;E = 0;#10;

A = 1;B = 0;E = 0;#10;

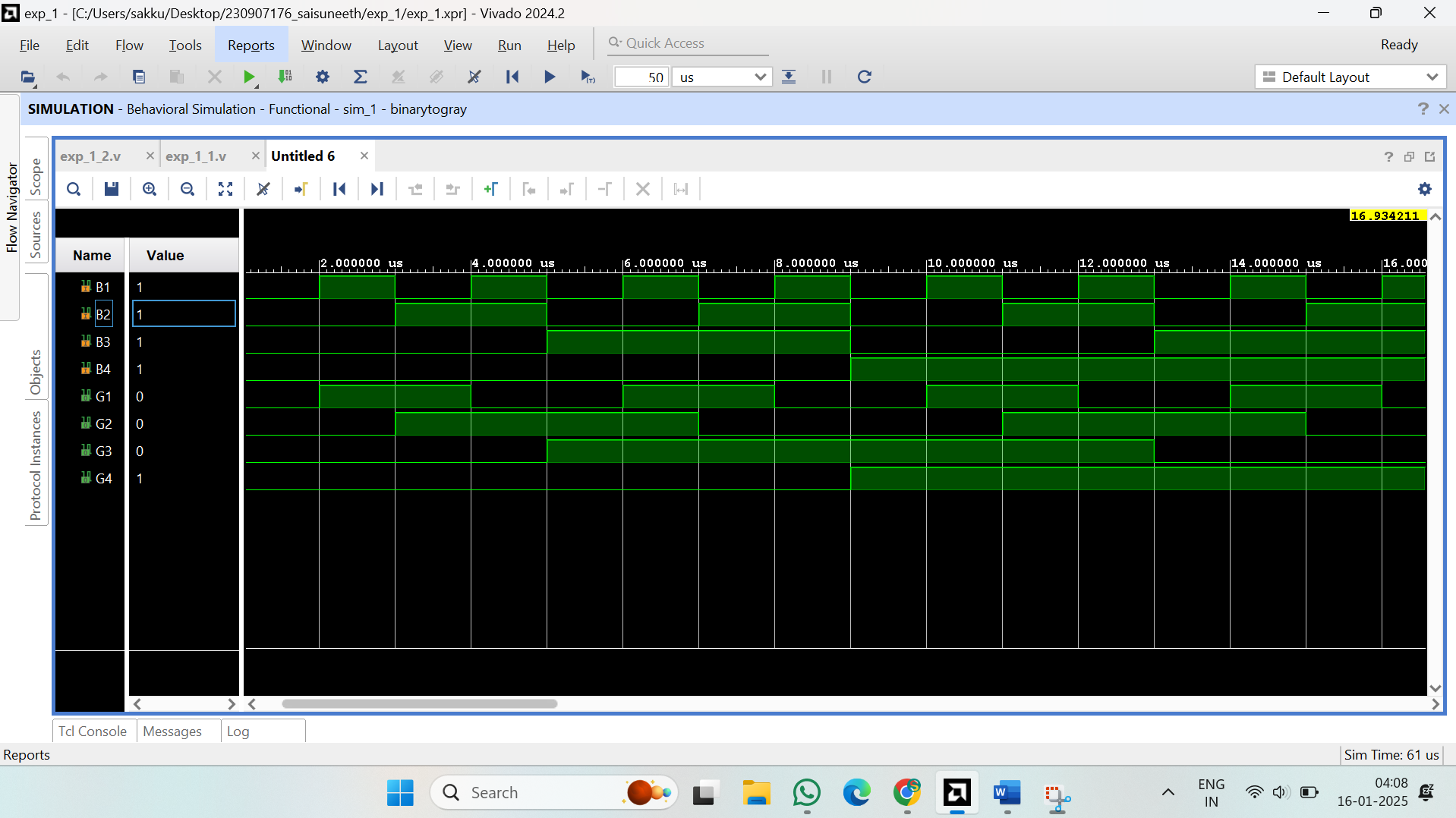
A = 0;B = 1;E = 0;#10;

A = 1;B = 1;E = 0;#10;

End

Endmodule

**Exp 1.2**

****

**A screenshot of a computer

Description automatically generated**

**Verilog code:**

module binarytogray(

input B1,B2,B3,B4,

output G1,G2,G3,G4

);

assign G4=B4;

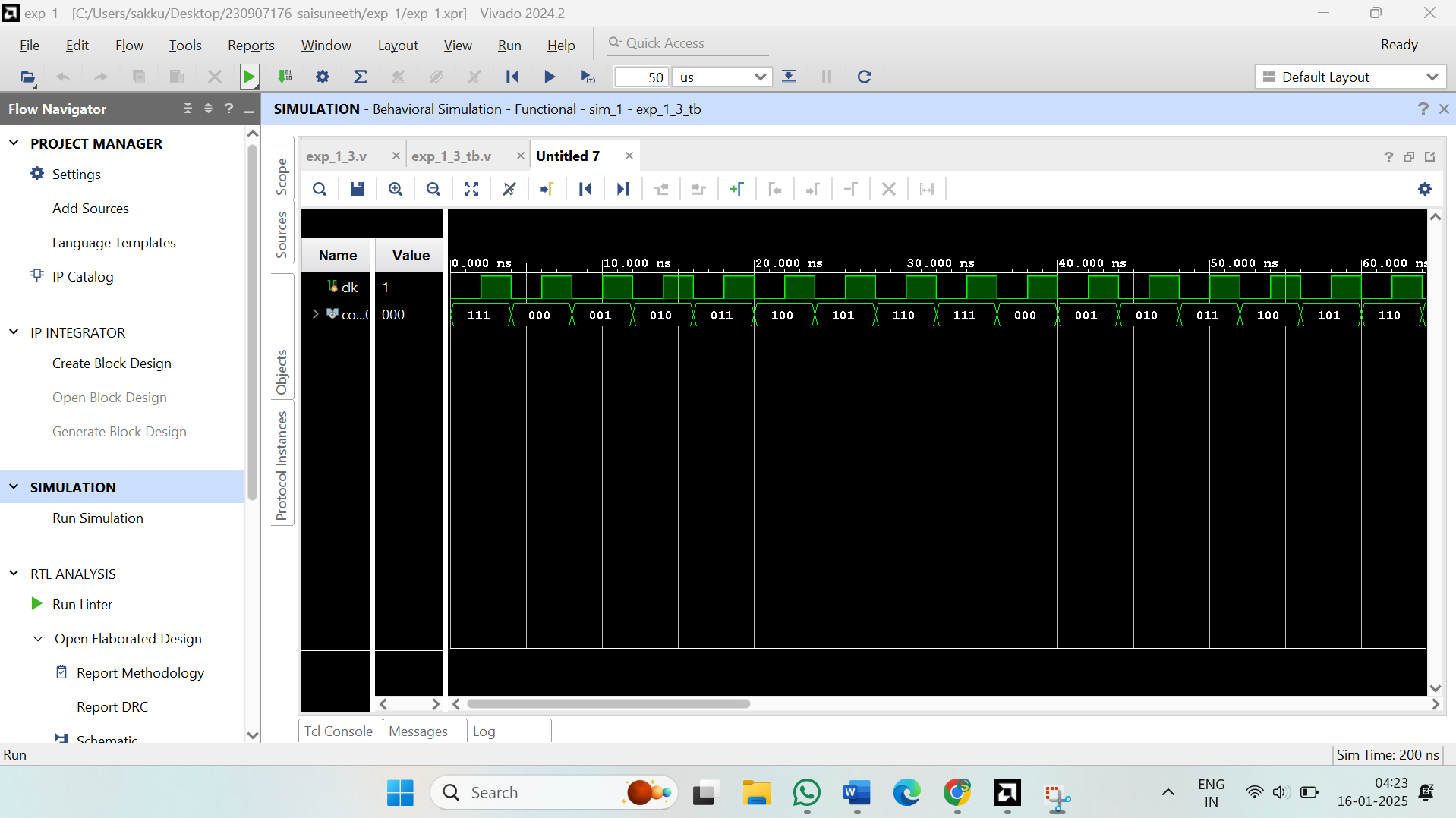
assign G3=B4^B3;

assign G2=B3^B2;

assign G1=B1^B2;

endmodule

**Exp 1.3**

****

**A screenshot of a computer

Description automatically generated**

**Verilog code:**

module counter( clk, count );

input clk;

output[2:0] count;

reg[2:0] count;

wire clk;

initial

count = 3'b0;

always @( negedge clk )

count[0] <= ~count[0];

always @( negedge count[0] )

count[1] <= ~count[1];

always @( negedge count[1] )

count[2] <= ~count[2];

endmodule

**Test bench:**

module exp\_1\_4\_tb;

reg clk;

reg clr;

wire [3:0] q;

Ringcounter R1(

.q(q),

.clk(clk),

.clr(clr)

);

always

begin

#50 clk=1'b1;

#50 clk=1'b0;

end

initial begin

clk = 0;

clr = 0;

#50 clr = 1'b1;

#100 clr = 1'b0;

#100;

end

endmodule

**Exp 1.4**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Verilog code:**

module counter( clk, count );

input clk;

output[2:0] count;

reg[2:0] count;

wire clk;

initial

count = 3'b0;

always @( negedge clk )

count[0] <= ~count[0];

always @( negedge count[0] )

count[1] <= ~count[1];

always @( negedge count[1] )

count[2] <= ~count[2];

endmodule

**Test bench:**

module exp\_1\_4\_tb;

reg clk;

reg clr;

wire [3:0] q;

Ringcounter R1(

.q(q),

.clk(clk),

.clr(clr)

);

always

begin

#50 clk=1'b1;

#50 clk=1'b0;

end

initial begin

clk = 0;

clr = 0;

#50 clr = 1'b1;

#100

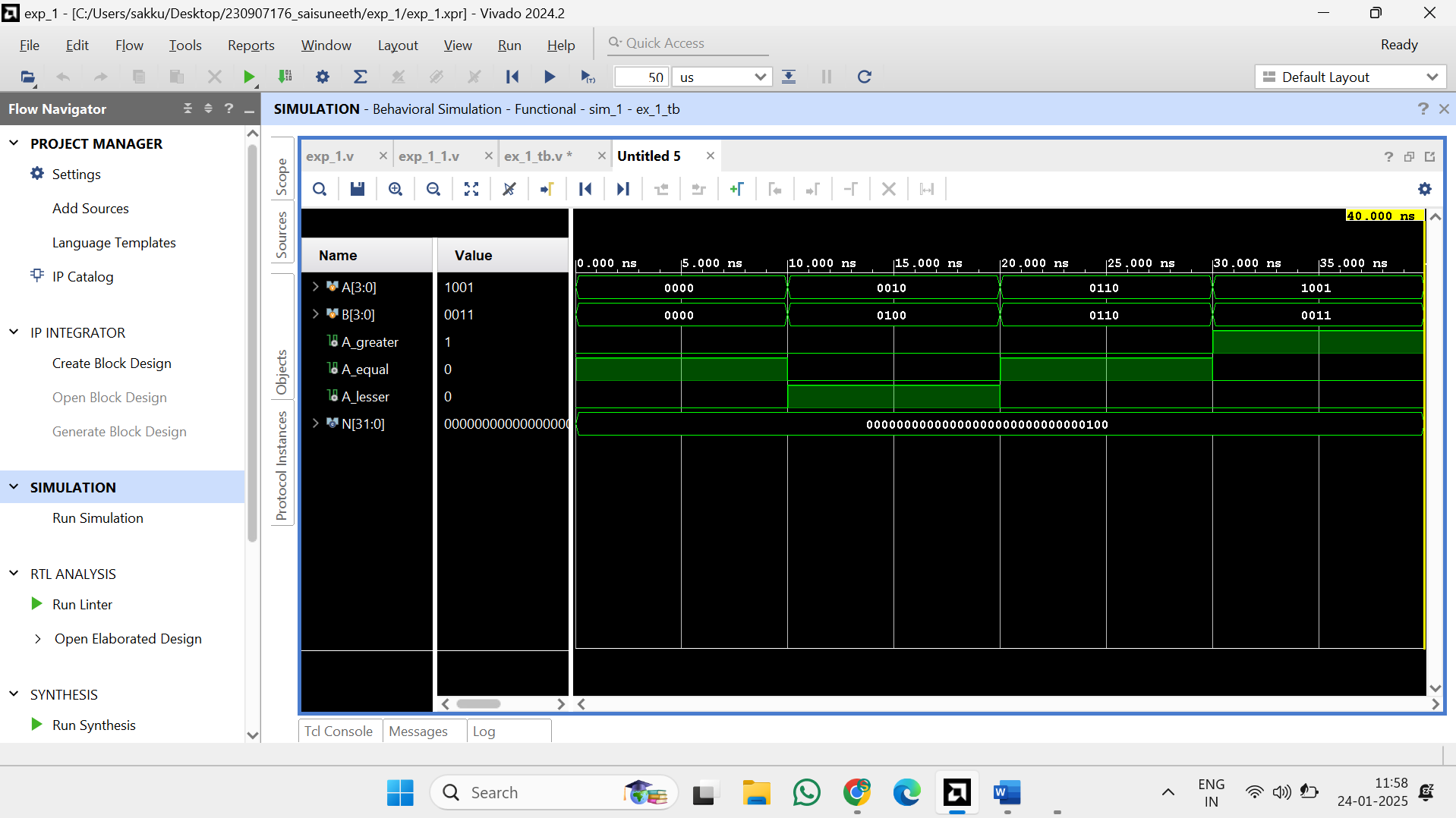
clr = 1'b0;

#100;

end

endmodule

**Ex 1:**

****

**Verilog code:**

module mag\_comp #(parameter N = 4) (

input [N-1:0] A,

input [N-1:0] B,

output A\_greater,

output A\_equal,

output A\_lesser

);

assign A\_greater = (A > B);

assign A\_equal = (A == B);

assign A\_lesser = (A < B);

endmodule

**Test bench:**

module ex\_1\_tb;

parameter N = 4;

reg [N-1:0] A;

reg [N-1:0] B;

wire A\_greater;

wire A\_equal;

wire A\_lesser;

mag\_comp #(N) uut (

.A(A),

.B(B),

.A\_greater(A\_greater),

.A\_equal(A\_equal),

.A\_lesser(A\_lesser)

);

initial begin

A = 0; B = 0;

#10;

// A<B

A = 4'b0010;

B = 4'b0100;

#10;

//A == B

A = 4'b0110; // 6

B = 4'b0110; // 6

#10;

//A > B

A = 4'b1001; // 9

B = 4'b0011; // 3

#10;

$finish;

end

endmodule

**Ex 2:**

**A screenshot of a computer

Description automatically generated**

**Verilog code:**

module four\_bit\_adder(

input [3:0]A,B,

input cin,

output [3:0]sum,

output cout);

assign {cout,sum}=A+B+cin;

endmodule

**Test bench:**

module ex\_2\_tb;

reg [3:0] A, B;

reg cin;

wire [3:0] sum;

wire cout;

four\_bit\_adder uut (

.A(A),

.B(B),

.cin(cin),

.sum(sum),

.cout(cout)

);

initial begin

//A=0, B=0, cin=0

A = 4'b0000; B = 4'b0000; cin = 1'b0;

#10;

//A=5, B=3, cin=0

A = 4'b0101; B = 4'b0011; cin = 1'b0;

#10;

//A=7, B=8, cin=1

A = 4'b0111; B = 4'b1000; cin = 1'b1;

#10;

//A=15, B=1, cin=1 (Overflow scenario)

A = 4'b1111; B = 4'b0001; cin = 1'b1;

#10;

//A=6, B=9, cin=0

A = 4'b0110; B = 4'b1001; cin = 1'b0;

#10;

//A=0, B=15, cin=1

A = 4'b0000; B = 4'b1111; cin = 1'b1;

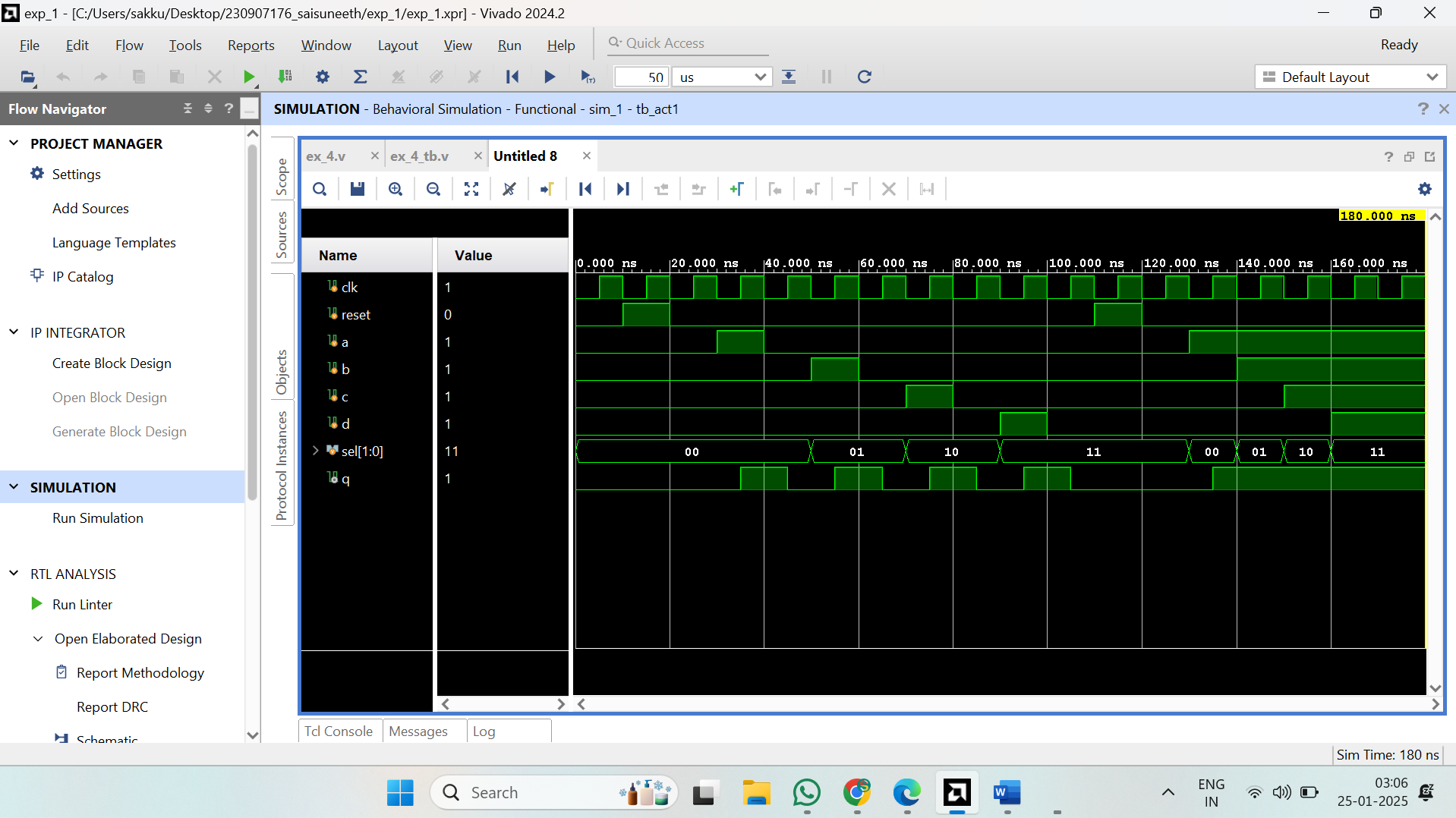
#10;

$finish;

end

endmodule

**Ex 4:**

****

**Verilog code:**

module act1(

input clk,reset,a,b,c,d,

input [1:0]sel,

output reg q

);

reg mux\_out;

always @(\*)

begin

case(sel)

2'b00:mux\_out=a;

2'b01:mux\_out=b;

2'b10:mux\_out=c;

2'b11:mux\_out=d;

endcase

end

initial

q=0;

always @(posedge clk or posedge reset)

begin

if (reset)

q<=0;

else

q<=mux\_out;

end

endmodule

**Test bench:**

module tb\_act1;

reg clk, reset, a, b, c, d;

reg [1:0] sel;

wire q;

// Instantiate the module under test (DUT)

act1 uut (

.clk(clk),

.reset(reset),

.a(a),

.b(b),

.c(c),

.d(d),

.sel(sel),

.q(q)

);

// Clock generation

initial begin

clk = 0;

forever #5 clk = ~clk; // 10 ns clock period

end

// Test sequence

initial begin

// Monitor signals for debugging

$monitor("Time=%0t | reset=%b | sel=%b | a=%b b=%b c=%b d=%b | mux\_out=%b | q=%b",

$time, reset, sel, a, b, c, d, uut.mux\_out, q);

// Initialize inputs

reset = 0; a = 0; b = 0; c = 0; d = 0; sel = 2'b00;

// Apply reset

#10 reset = 1;

#10 reset = 0;

// Test Case 1: Select input `a`

#10 sel = 2'b00; a = 1;

#10 a = 0;

// Test Case 2: Select input `b`

#10 sel = 2'b01; b = 1;

#10 b = 0;

// Test Case 3: Select input `c`

#10 sel = 2'b10; c = 1;

#10 c = 0;

// Test Case 4: Select input `d`

#10 sel = 2'b11; d = 1;

#10 d = 0;

// Test reset during operation

#10 reset = 1;

#10 reset = 0;

// Test multiple changes

#10 sel = 2'b00; a = 1;

#10 sel = 2'b01; b = 1;

#10 sel = 2'b10; c = 1;

#10 sel = 2'b11; d = 1;

// End simulation

#20 $finish;

end

endmodule

**Ex 5:**

**Verilog code:**

**Test bench:**

**Ex 6:**

**Verilog code:**

**Test bench:**

**Ex 7:**

**Verilog code:**

**Test bench:**

**Ex 8:**

**Verilog code:**

**Test bench:**

**Ex 9:**

**Verilog code:**

**Test bench:**