EL467 DIGITAL

PROGRAMMING LAB-8

REPORT

DETROJA AVI – 202201452

RIDHAM PATEL - 202201430

Aim: Introduction to switch level modeling and strengths in Verilog HDL.

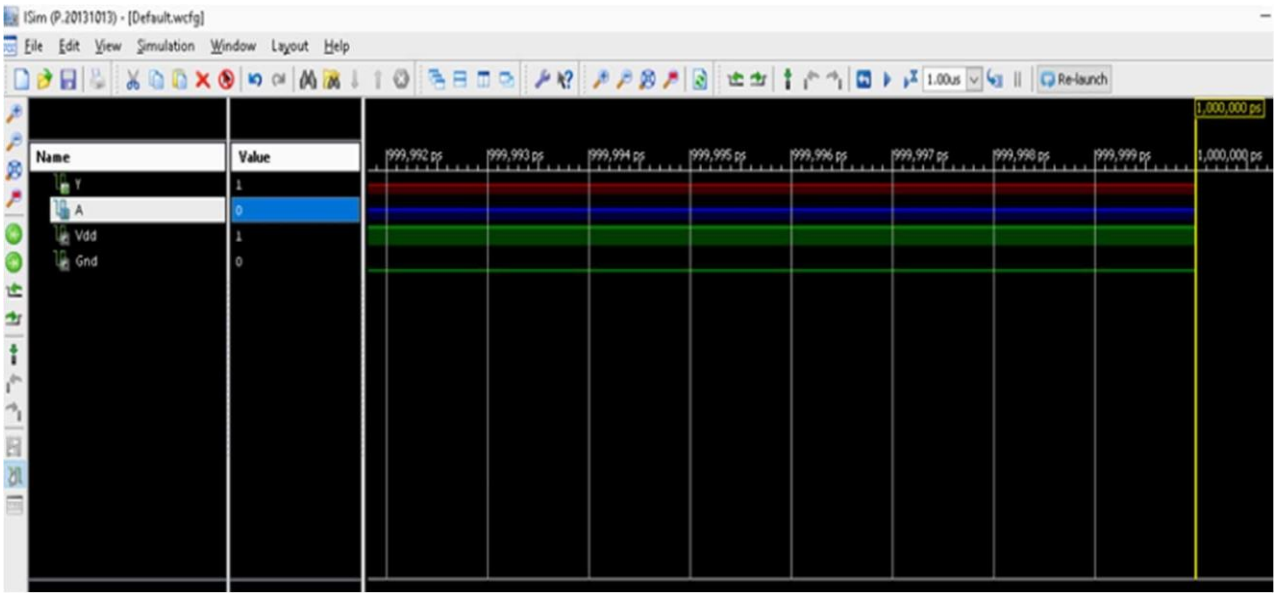
Q.1

a) NOT

Code:

```
module Q1_not(output Y, input A);
supply1 Vdd;
supply0 Gnd;
pmos P1 (Y, Vdd, A);
nmos N1 (Y, Gnd, A);
endmodule
```

Output:

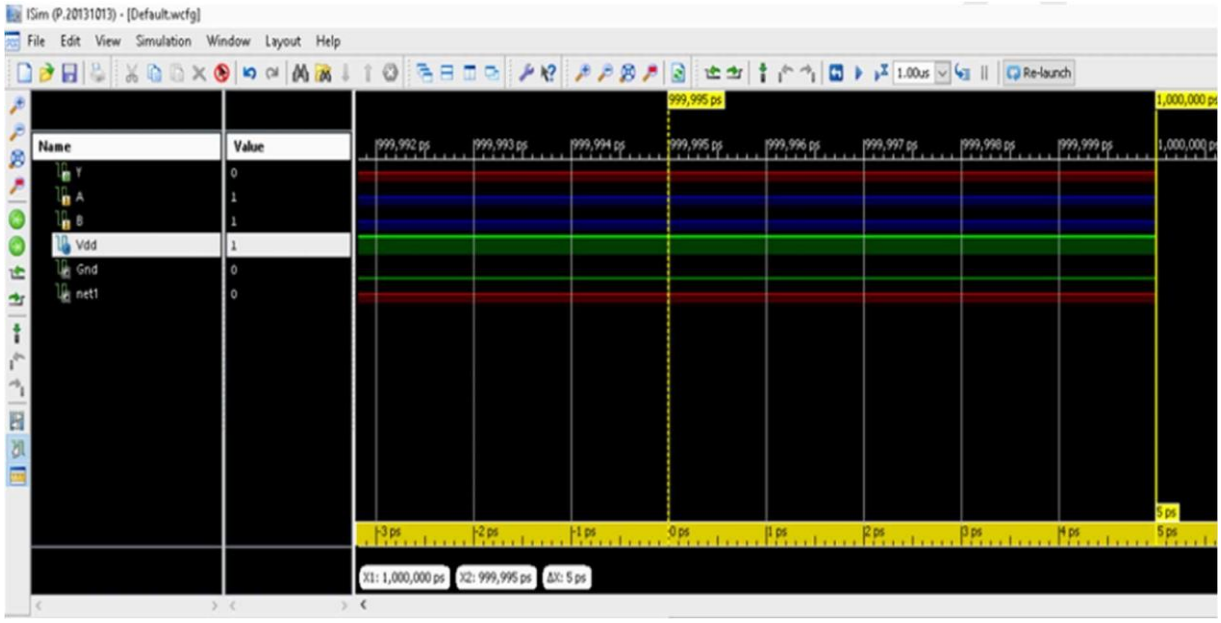


b) NOR

Code:

```
module Q1_nor (output Y, input A, input B);
supply1 Vdd;
supply0 Gnd;
pmos P1 (Y, Vdd, A);
pmos P2 (Y, Vdd, B);
nmos N1 (Y, net1, A);
nmos N2 (net1, Gnd, B);
endmodule
```

Output:

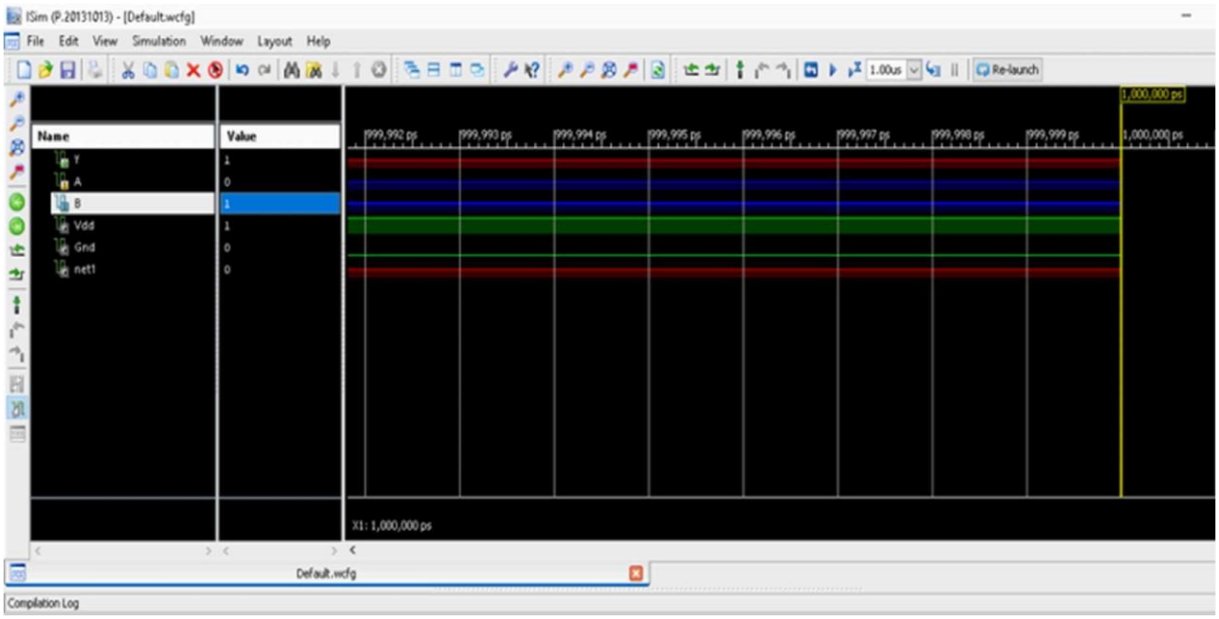


c) NAND

Code:

```
module Q1_nand (output Y, input A, input B);
supply1 Vdd;
supply0 Gnd;
pmos P1 (Y, Vdd, A);
pmos P2 (Y, Vdd, B);
nmos N1 (Y, net1, A);
nmos N2 (net1, Gnd, B);
endmodule
```

Output:

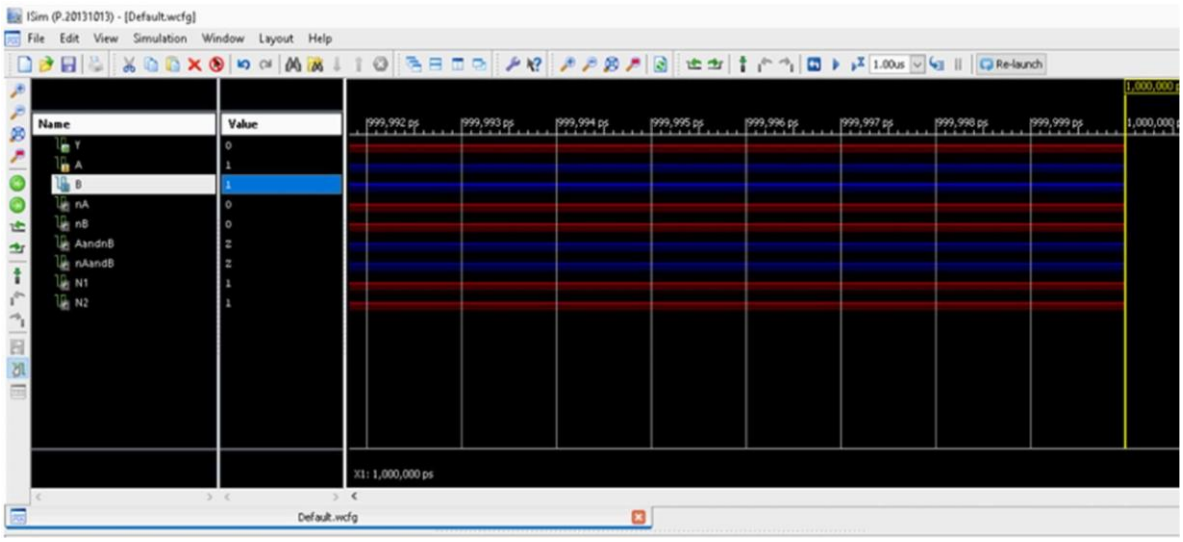


d) XOR

Code:

```
module Q1_xor(output Y, input A, input B);
wire nA, nB;
wire AandnB, nAandB;
lab8_not not1(nA, A);
lab8_not not2(nB, B);
nand (N1, A, nB);
nand (N2, nA, B);
lab8_nor or1(Y, N1, N2);
endmodule
```

Output:

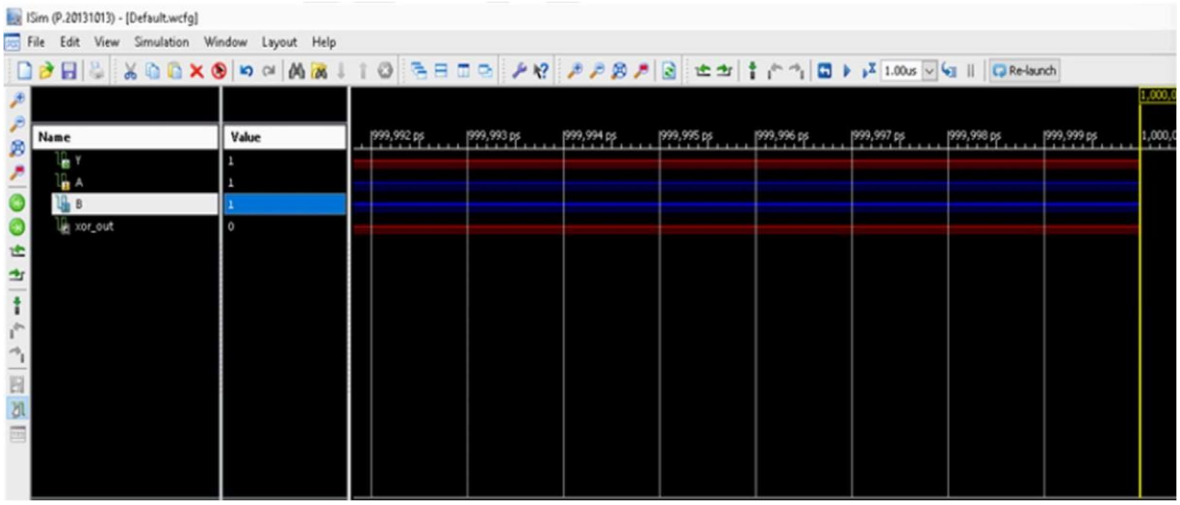


e) XNOR

Code:

```
module Q1_xnor(output Y, input A, input B);
wire xor_out;
lab8_xor xor1(xor_out, A, B);
lab8_not not1(Y, xor_out);
endmodule
```

Output:



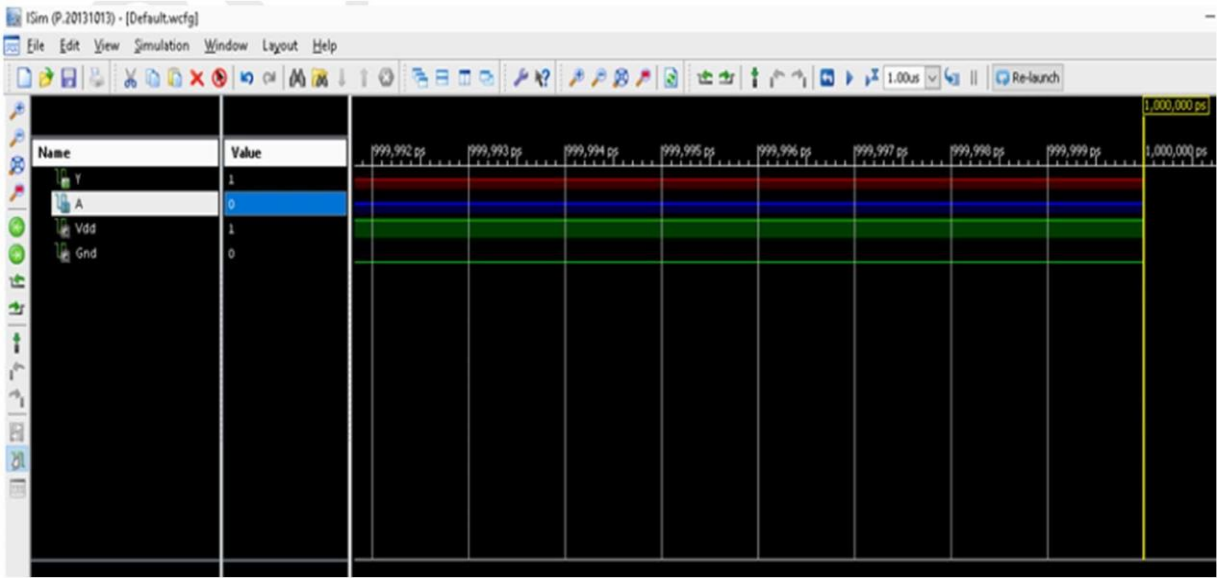
Q.2

a) NOT

Code:

```
module Q2_not(output Y, input A);
supply1 Vdd;
supply0 Gnd;
nmos N1 (Y, Gnd, A);
assign Y = A? Gnd: Vdd;
endmodule
```

Output:

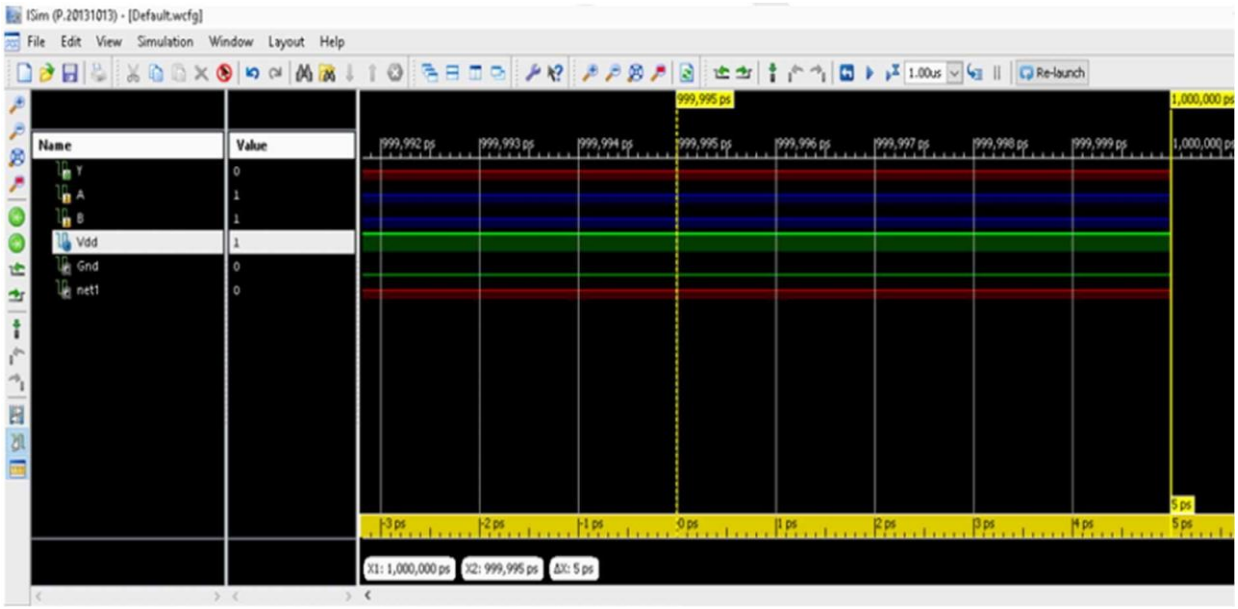


b) NOR

Code:

```
module Q2_nor(output Y, input A, input B);
supply1 Vdd;
supply0 Gnd;
nmos N1 (Y, Gnd, A);
nmos N2 (Y, Gnd, B);
assign Y = (A&B)? Gnd Vdd;
endmodule
```

Output:

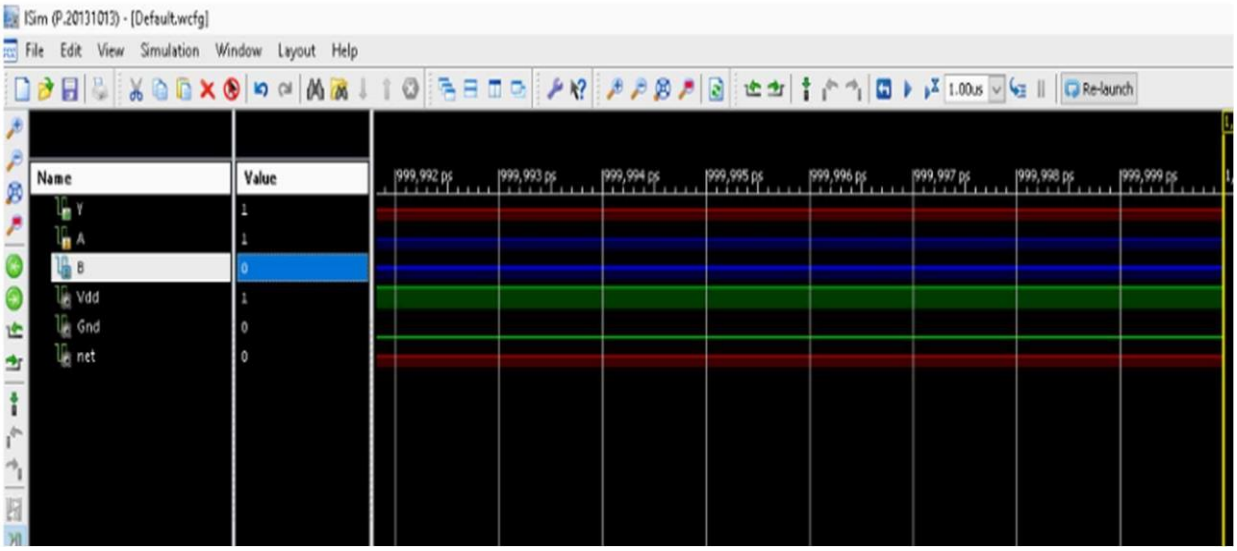


c) NAND

Code:

```
module Q2_nand(output Y, input A, input B);
supply1 Vdd;
supply@ Gnd;
nmos N1 (net, Gnd, A);
nmos N2 (Y, net, B);
assign Y = (A & B) ? Gnd: Vdd;
endmodule
```

Output:

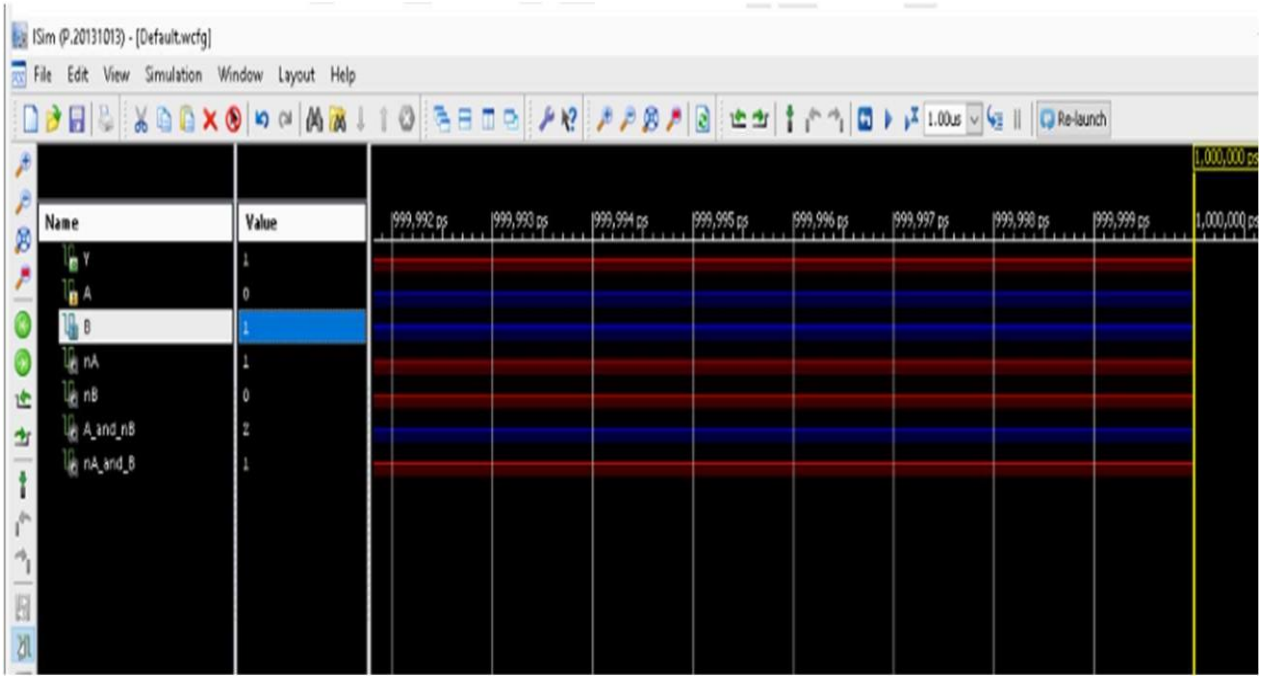


d) XOR

Code:

```
module Q2_xor_2(output Y, input A, input B);
wire nA, nB, A_and_nB, nA_and_B;
not U1 (nA, A);
not U2 (nB, B);
nmos N1 (A_and_nB, A, nB);
nmos N2 (nA_and_B, nA, B);
or U3 (Y, A_and_nB, nA_and_B);
endmodule
```

Output:

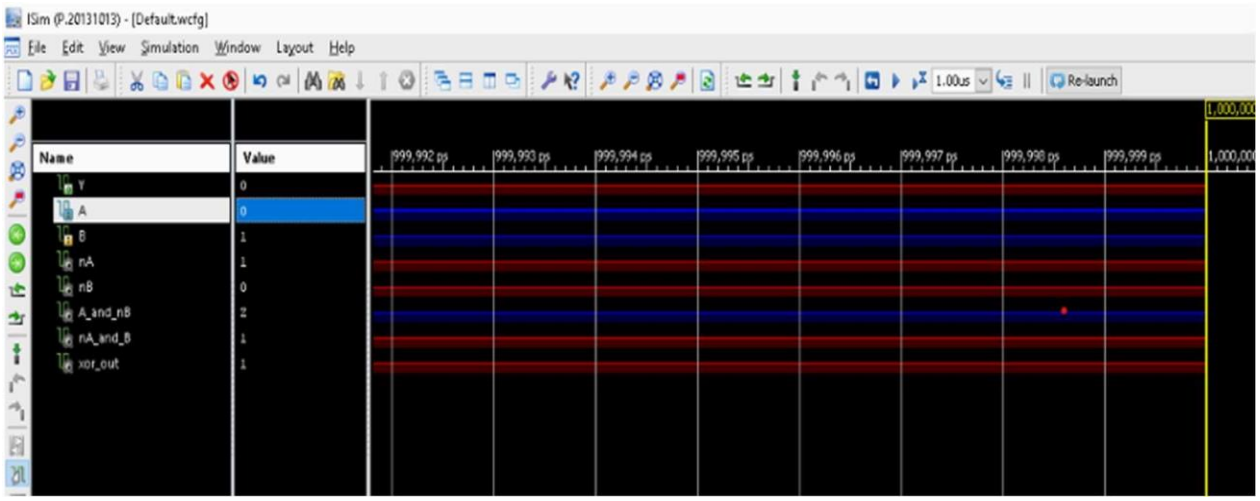


e) XNOR

Code:

```
module Q2_xnor_2(output Y, input A, input B);
wire nA, nB, A_and_nB, nA_and_B, xor_out;
not U1 (nA, A);
not U2 (nB, B);
nmos N1 (A_and_nB, A, nB);
nmos N2 (nA_and_B, nA, B);
or U3 (xor_out, A_and_nB, nA_and_B);
not U4 (Y, xor_out);
endmodule
```

Output:



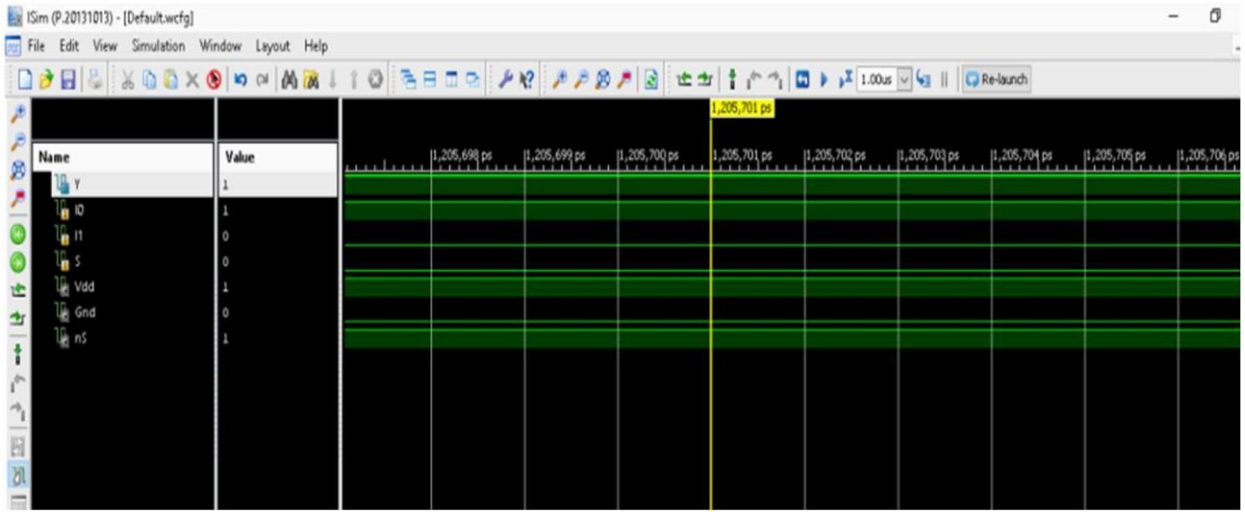
Q.3

a) 2 to 1 mux

Code:

```
Module Q3_21mux (output Y, input 10, 11, 5);
supply1 Vdd;
supply0 Gnd;
wire nS;
not U1 (nS, S);
nmos N1 (Y, 10, nS);
nmos N2 (Y, 11, S);
endmodule
```

Output:

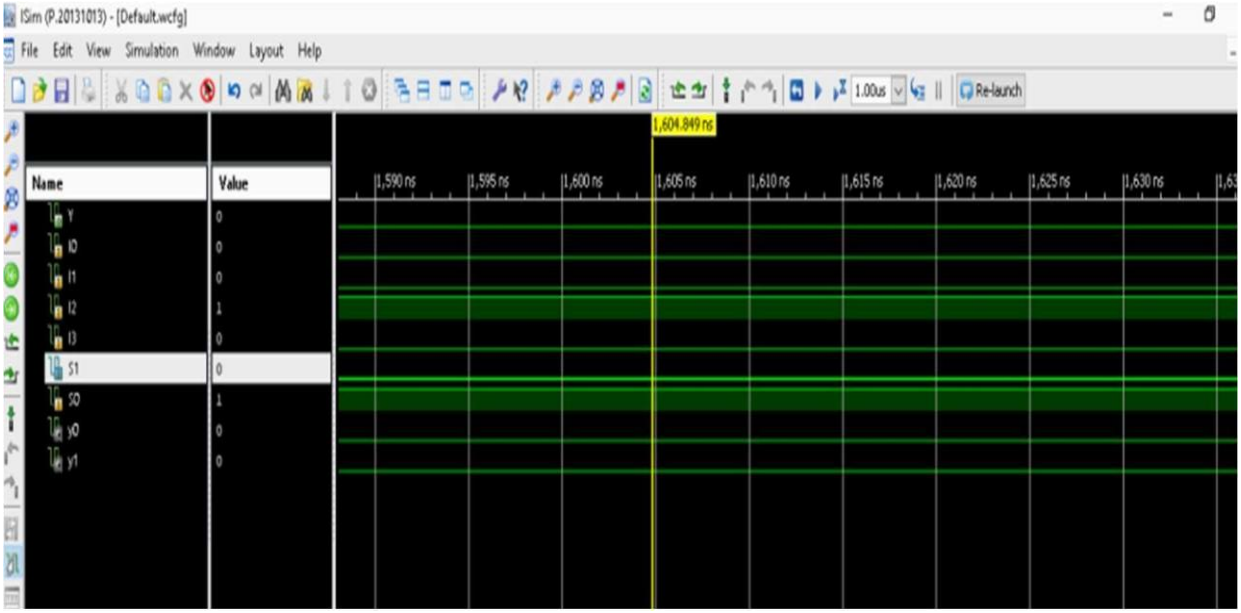


b) 4 to 1 mux

Code:

```
module Q3_41mux (output Y, input 10, 11, 12, 13, input S1, S0);
wire yo, y1;
lab8_21mux g1(y0, 10, 11, 50);
lab8_21mux g2(y1, 12, 13, 50);
lab8_21mux g3(Y, yo, y1, 51);
endmodule
```

Output:



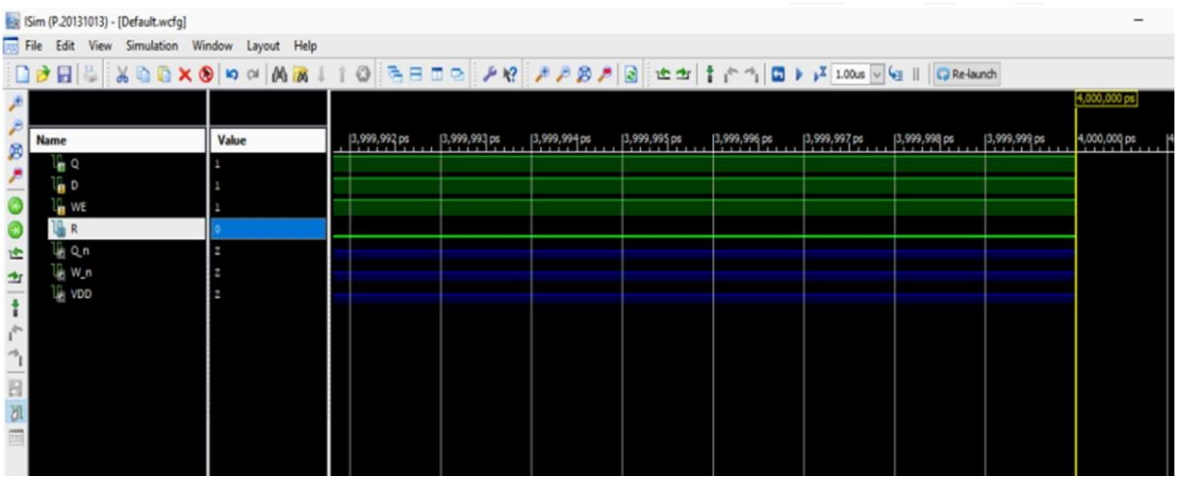
Q4) Design SRAM

(i) Without using strength in Verilog

Code:

```
module Q4_SRAM(output Q, input D, input WE, input R);
wire Qn, W_n;
nmos (Q, D, WE);
nmos (Qn, Q, R);
pmos (Q, VDD, WE);
pmos (Q_n, VDD, R);
endmodule
```

Output:

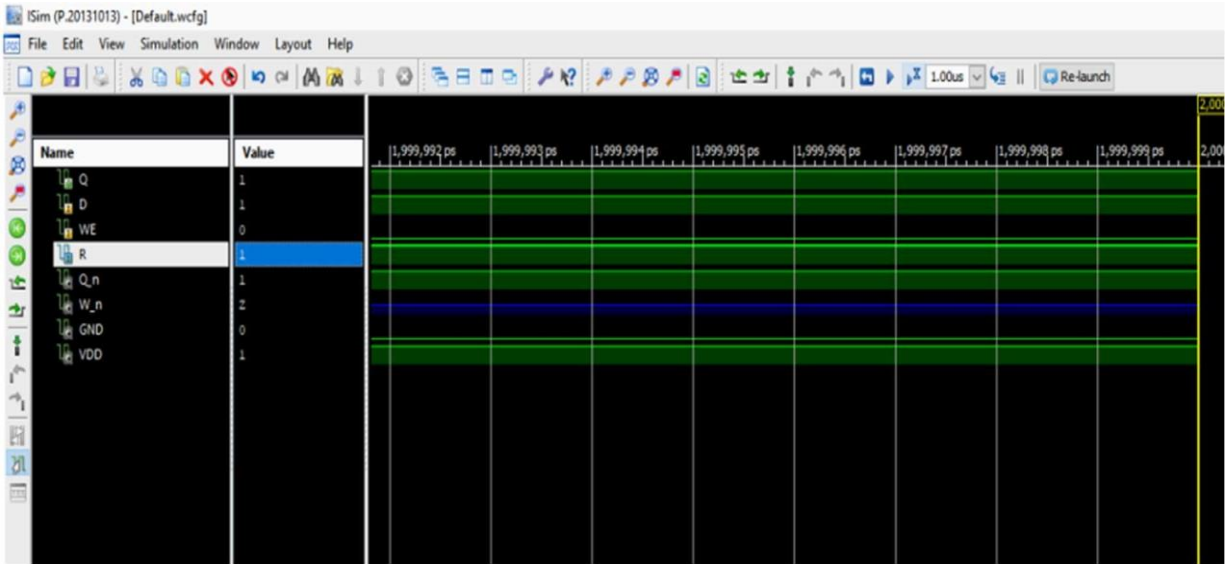


(ii) Using strength in Verilog

Code:

```
module Q4_SRAM_withstrength(output Q, input D, input WE,
input R);
wire Q_n, W_n;
supply@ GND;
supply1 VDD;
nmos #(1, 1) (Q, D, WE);
nmos #(1, 1) (Qn, Q, R); pmos #(2, 2) (Q, VDD, WE);
pmos #(2, 2) (Q_n, VDD, R);
endmodule
```

Output:



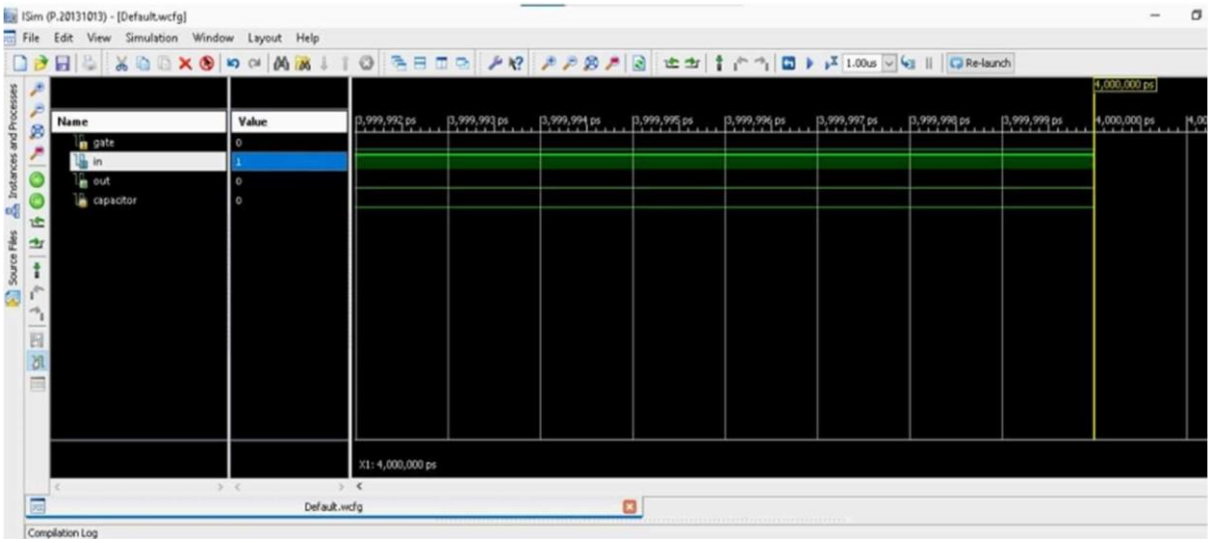
Lab Exercise

Q.1

Code:

```
module exc_1(input gate, input in, output reg out);
reg capacitor;
always @(gate or in) begin
if (gate) begin
capacitor = in;
end else begin
out = capacitor;
end
end
endmodule
```

Output:



Q.2

Code:

```
module full_adder_1bit (input a, b, cin, output sum, cout);
wire xor1_out, and1_out, and2_out;
xor xor1(xorl_out, a, b);
xor xor2(sum, xor1_out, cin);
and and1 (and1_out, a, b);
and and2 (and2_out, xor1_out, cin);
or or1(cout, and1_out, and2_out);
endmodule
module Q2_exc_2(input [3:0] a, b, input cin, output [3:0] sum,
output
cout);
wire c1, c2, c3;
full_adder_1bit fał (.a(a[0]), .b(b[0]), .cin(cin),
sum(sum[0]), .cout(c1));
full_adder_1bit fal (.a(a[1]), .b(b[1]), .cin(c1),
sum(sum[1]), .cout(c2));
full_adder_1bit fa2 (.a(a[2]), .b(b[2]), .cin(c2), .sum(sum[2]),
.cout(c3));
full_adder_1bit fa3 (.a(a[3]), .b(b[3]), .cin(c3), .sum(sum[3]),
.cout(cout));
Endmodule
```

Output: