LAB NO 10

Fall 2024

CSE-304L Computer Organization and Architecture Lab

Submitted by:

Name : **Hassan Zaib Jadoon**

Reg no**. : 22PWCSE2144**

Class Section **: A**

Signature: ___________

Submitted to:

Dr. Amaad Khalil

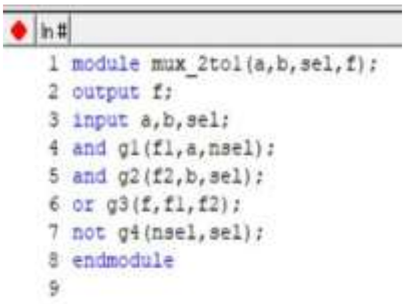
Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

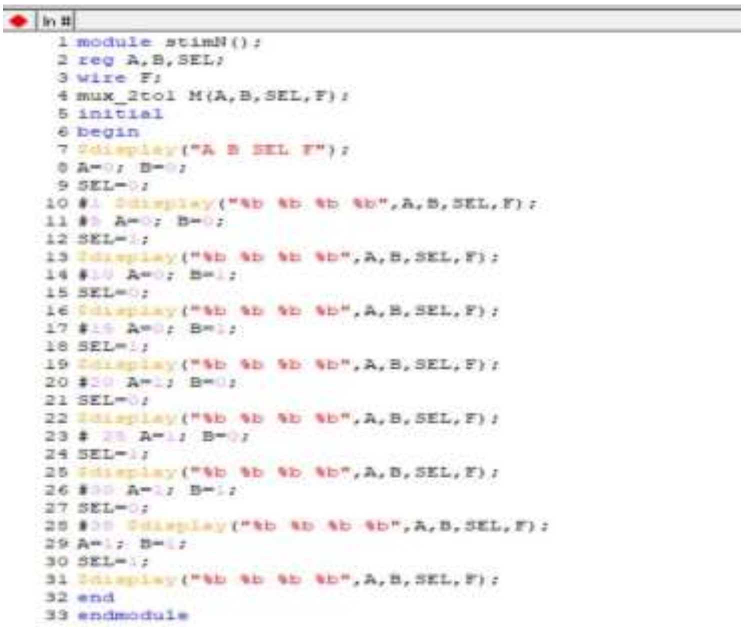
MULTIPLEXER

TASK 01:

Write a Verilog code for 2x1 Mux using Dataflow Level modeling.

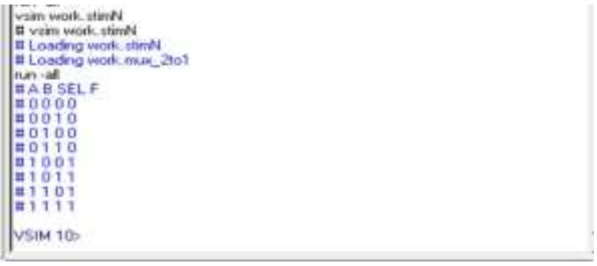


```
module mux_2to1(a,b,sel,f);
output f;
input a,b,sel;
and g1(f1,a,nsel);
and g2(f2,b,sel);
or g3(f,f1,f2);
not g4(nsel,sel);
endmodule
```

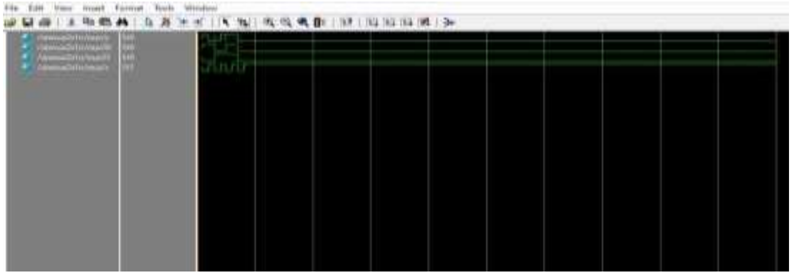


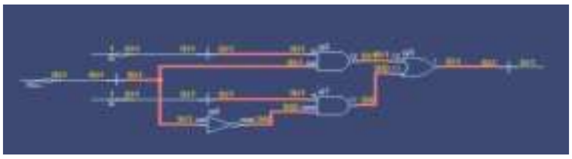
```
module stimN();
reg A,B,SEL;
wire F;
mux_2tol M(A,B,SEL,F);
initial
begin
$display("A B SEL F");
A=0; B=0;
SEL=0;
#1 $display("%b %b %b %b",A,B,SEL,F);
#5 A=0; B=0;
SEL=1;
$display("%b %b %b %b",A,B,SEL,F);
#10 A=0; B=1;
SEL=0;
$display("%b %b %b %b",A,B,SEL,F);
#15 A=0; B=1;
SEL=1;
$display("%b %b %b %b",A,B,SEL,F);
#20 A=1; B=0;
SEL=0;
$display("%b %b %b %b",A,B,SEL,F);
# 25 A=1; B=0;
SEL=1;
$display("%b %b %b %b",A,B,SEL,F);
#30 A=1; B=1;
SEL=0;
#35 $display("%b %b %b %b",A,B,SEL,F);
A=1; B=1;
SEL=1;
$display("%b %b %b %b",A,B,SEL,F);
end
endmodule
```

Output:



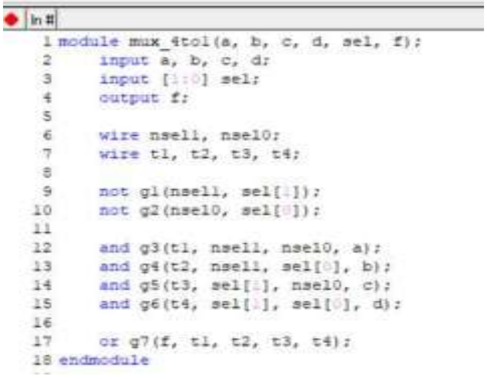
```
vsim work.stimN
# vsim work.stimN
# Loading work.stimN
# Loading work.mux_2to1
run -all
# A B SEL F
# 0 0 0 0
# 0 0 1 0
# 0 1 0 0
# 0 1 1 0
# 1 0 0 1
# 1 0 1 1
# 1 1 0 1
# 1 1 1 1

VSIM 10>
```



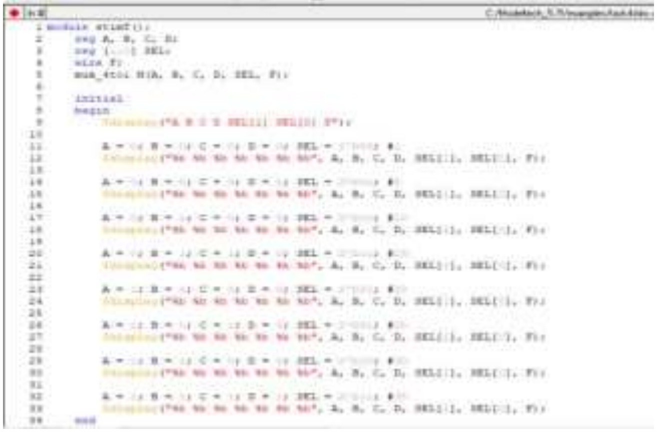


Task 02:

Write a Verilog code for 4x1 Mux using Dataflow Level modeling.



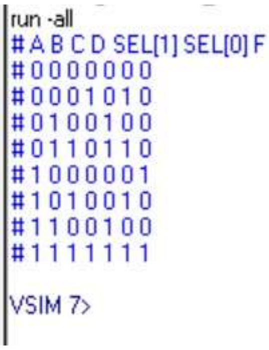
```
module mux_4to1(a, b, c, d, sel, f);
    input a, b, c, d;
    input [1:0] sel;
    output f;

    wire nsel1, nsel0;
    wire t1, t2, t3, t4;

    not g1(nsel1, sel[1]);
    not g2(nsel0, sel[0]);

    and g3(t1, nsel1, nsel0, a);
    and g4(t2, nsel1, sel[0], b);
    and g5(t3, sel[1], nsel0, c);
    and g6(t4, sel[1], sel[0], d);

    or g7(f, t1, t2, t3, t4);
endmodule
```



```
module stimf();
    reg A, B, C, D;
    reg [1:0] SEL;
    wire F;
    mux_4to1 M(A, B, C, D, SEL, F);

    initial
    begin
        $display("A B C D SEL[1] SEL[0] F");

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);

        A = [illegible]; B = [illegible]; C = [illegible]; D = [illegible]; SEL = [illegible]; #[illegible]
        $display("%b %b %b %b %b %b %b", A, B, C, D, SEL[1], SEL[0], F);
    end
```

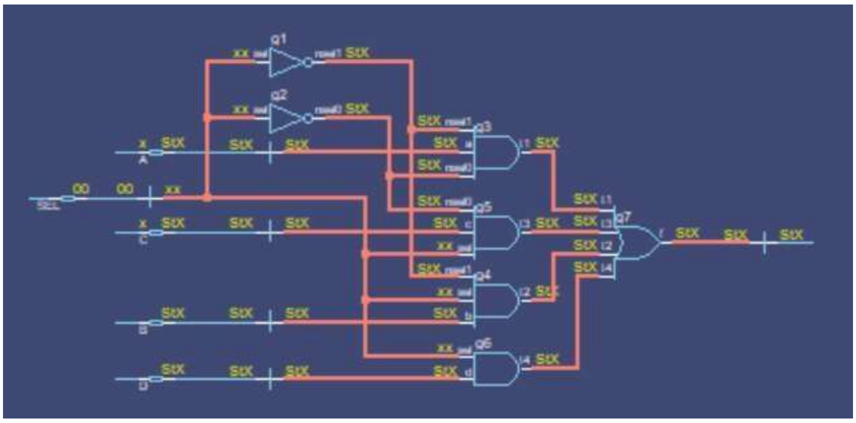
Output:

Table:



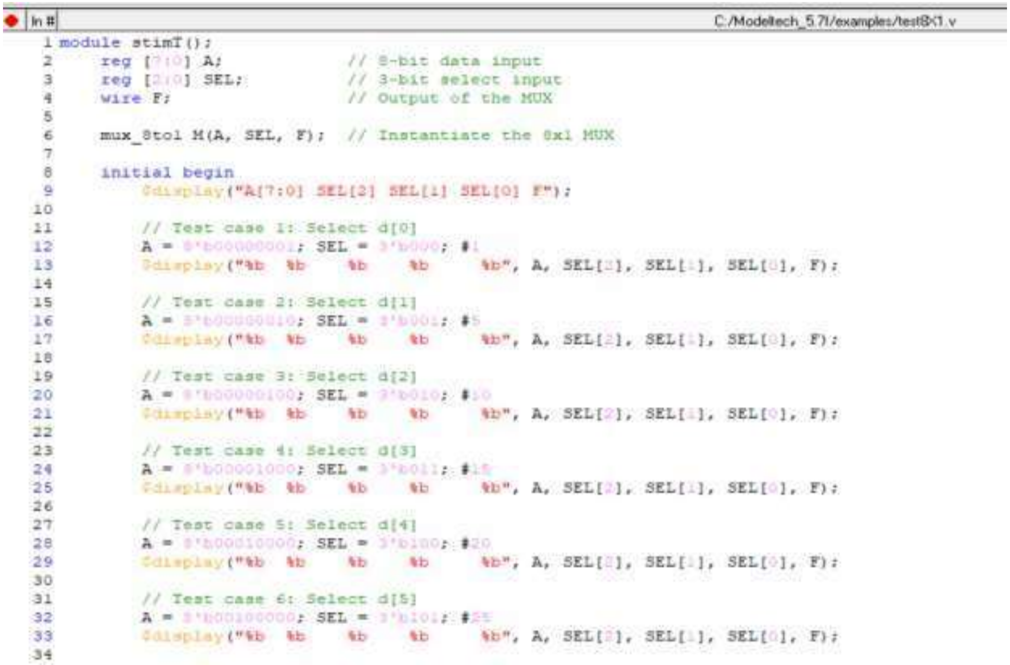
```
run -all
# A B C D SEL[1] SEL[0] F
# 0 0 0 0 0 0 0
# 0 0 0 1 0 1 0
# 0 1 0 0 1 0 0
# 0 1 1 0 1 1 0
# 1 0 0 0 0 0 1
# 1 0 1 0 0 1 0
# 1 1 0 0 1 0 0
# 1 1 1 1 1 1 1

VSIM 7>
```

Logical diagram:



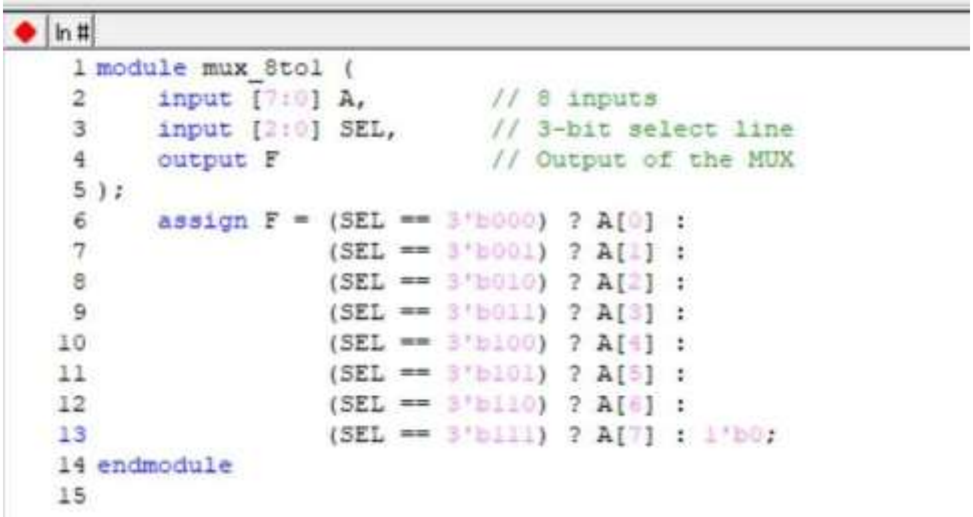
Task3:

Write a Verilog code for 8x1 Mux using Dataflow Level modeling.



C:/Modeltech_5.7l/examples/test8X1.v

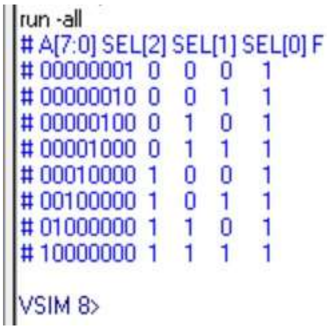
```
module stimT();
    reg [7:0] A;              // 8-bit data input
    reg [2:0] SEL;            // 3-bit select input
    wire F;                   // Output of the MUX

    mux_8to1 M(A, SEL, F);  // Instantiate the 8x1 MUX

    initial begin
        $display("A[7:0] SEL[2] SEL[1] SEL[0] F");

        // Test case 1: Select d[0]
        A = 8'b00000001; SEL = 3'b000; #1
        $display("%b  %b    %b    %b      %b", A, SEL[2], SEL[1], SEL[0], F);

        // Test case 2: Select d[1]
        A = 8'b00000010; SEL = 3'b001; #5
        $display("%b  %b    %b    %b      %b", A, SEL[2], SEL[1], SEL[0], F);

        // Test case 3: Select d[2]
        A = 8'b00000100; SEL = 3'b010; #10
        $display("%b  %b    %b    %b      %b", A, SEL[2], SEL[1], SEL[0], F);

        // Test case 4: Select d[3]
        A = 8'b00001000; SEL = 3'b011; #15
        $display("%b  %b    %b    %b      %b", A, SEL[2], SEL[1], SEL[0], F);

        // Test case 5: Select d[4]
        A = 8'b00010000; SEL = 3'b100; #20
        $display("%b  %b    %b    %b      %b", A, SEL[2], SEL[1], SEL[0], F);

        // Test case 6: Select d[5]
        A = 8'b00100000; SEL = 3'b101; #25
        $display("%b  %b    %b    %b      %b", A, SEL[2], SEL[1], SEL[0], F);

```



```
module mux_8to1 (
    input [7:0] A,          // 8 inputs
    input [2:0] SEL,        // 3-bit select line
    output F                // Output of the MUX
);
    assign F = (SEL == 3'b000) ? A[0] :
               (SEL == 3'b001) ? A[1] :
               (SEL == 3'b010) ? A[2] :
               (SEL == 3'b011) ? A[3] :
               (SEL == 3'b100) ? A[4] :
               (SEL == 3'b101) ? A[5] :
               (SEL == 3'b110) ? A[6] :
               (SEL == 3'b111) ? A[7] : 1'b0;
endmodule

```

Output:

Table:



```
run -all
# A[7:0] SEL[2] SEL[1] SEL[0] F
# 00000001  0   0   0   1
# 00000010  0   0   1   1
# 00000100  0   1   0   1
# 00001000  0   1   1   1
# 00010000  1   0   0   1
# 00100000  1   0   1   1
# 01000000  1   1   0   1
# 10000000  1   1   1   1

VSIM 8>
```