CS223-003 (DIGITAL DESIGN)

LAB 3

NAME: SERTAÇ DERYA

ID: 22003208

DATE: 31.10.2022

1-2 Decoder

Module:

```
module one_to_two_decoder(
    input a,
    input enable,
    output y0,
    output y1
);
    assign y0 = enable & ~a;
    assign y1 = enable & a;
endmodule
```

TestBench:

```
module one_to_two_decoder_sim( );
    logic a, enable, y0, y1;
    one_to_two_decoder dut(.a(a),.enable(enable),.y0(y0),.y1(y1));
    initial begin
        a = 0; enable = 0; #10;
        a = 1; #10;
        a = 0; enable = 1; #10;
        a = 1; #10;
    end
endmodule
```

2-4 Decoder

Module:

```
module two_to_four_decoder(
    input a,
    input b,
    input enable,
    output[3:0] y
);
    logic enable_for_first;
    logic enable_for_second;

    one_to_two_decoder
decoder_first(.a(a),.enable(enable),.y0(enable_for_first),.y1(enable
_for_second));

    one_to_two_decoder
decoder_zero_one(.a(b),.enable(enable_for_first),.y0(y[0]),.y1(y[1])
);
    one_to_two_decoder
decoder_two_three(.a(b),.enable(enable_for_second),.y0(y[2]),.y1(y[3
]));
endmodule
```

TestBench:

```
module two_to_four_decoder_sim( );
    logic a, b, enable;
    logic [3:0]y;
    two_to_four_decoder dut(.a(a),.b(b),.enable(enable),.y(y[3:0]));
    initial begin
        a = 0; b = 0; enable = 0; #10;
        a = 1; #10;
        a = 0; b = 1; #10;
        a = 1; #10;
        a = 0; b = 0; enable = 1; #10;
        a = 1; #10;
        a = 0; b = 1; #10;
        a = 1; #10;
    end
endmodule
```

2-1 Multiplexer

Module:

```
module two_to_one_mux(
    input a,
    input b,
    input select,
    output y
);
    logic result_a;
    logic result_b;

    assign result_a = a & ~select;
    assign result_b = b & select;

    assign y = result_a | result_b;
endmodule
```

4-1 Multiplexer

Module:

```
module four_to_one_mux(
    input a,
    input b,
    input c,
    input d,
    input s0,
    input s1,
    output y
);
    logic result_first;
    logic result_second;

    two_to_one_mux
first_mux(.a(a),.b(b),.select(s0),.y(result_first));
    two_to_one_mux
second_mux(.a(c),.b(d),.select(s0),.y(result_second));

    two_to_one_mux
result_mux(.a(result_first),.b(result_second),.select(s1),.y(y));
endmodule
```

TestBench:

```
module four_to_one_mux_sim( );
    logic a, b, c, d, s0, s1, y;
    four_to_one_mux
dut(.a(a),.b(b),.c(c),.d(d),.s0(s0),.s1(s1),.y(y));
    initial begin
        a = 0; b = 0; c = 0; d = 0; s0 = 0; s1 = 0; #10;
        a = 0; b = 0; c = 0; d = 1; s0 = 0; s1 = 0; #10;
        a = 0; b = 0; c = 1; d = 0; s0 = 0; s1 = 0; #10;
        a = 0; b = 0; c = 1; d = 1; s0 = 0; s1 = 0; #10;
        a = 0; b = 1; c = 0; d = 0; s0 = 0; s1 = 0; #10;
        a = 0; b = 1; c = 0; d = 1; s0 = 0; s1 = 0; #10;
        a = 0; b = 1; c = 1; d = 0; s0 = 0; s1 = 0; #10;
        a = 0; b = 1; c = 1; d = 1; s0 = 0; s1 = 0; #10;
        a = 1; b = 0; c = 0; d = 0; s0 = 0; s1 = 0; #10;
        a = 1; b = 0; c = 0; d = 1; s0 = 0; s1 = 0; #10;
        a = 1; b = 0; c = 1; d = 0; s0 = 0; s1 = 0; #10;
        a = 1; b = 0; c = 1; d = 1; s0 = 0; s1 = 0; #10;
        a = 1; b = 1; c = 0; d = 0; s0 = 0; s1 = 0; #10;
        a = 1; b = 1; c = 0; d = 1; s0 = 0; s1 = 0; #10;
```

```
a = 1; b = 1; c = 1; d = 0; s0 = 0; s1 = 0; #10;
a = 1; b = 1; c = 1; d = 1; s0 = 0; s1 = 0; #10;

a = 0; b = 0; c = 0; d = 0; s0 = 1; s1 = 0; #10;
a = 0; b = 0; c = 0; d = 1; s0 = 1; s1 = 0; #10;
a = 0; b = 0; c = 1; d = 0; s0 = 1; s1 = 0; #10;
a = 0; b = 0; c = 1; d = 1; s0 = 1; s1 = 0; #10;
a = 0; b = 1; c = 0; d = 0; s0 = 1; s1 = 0; #10;
a = 0; b = 1; c = 0; d = 1; s0 = 1; s1 = 0; #10;
a = 0; b = 1; c = 1; d = 0; s0 = 1; s1 = 0; #10;
a = 0; b = 1; c = 1; d = 1; s0 = 1; s1 = 0; #10;
a = 1; b = 0; c = 0; d = 0; s0 = 1; s1 = 0; #10;
a = 1; b = 0; c = 0; d = 1; s0 = 1; s1 = 0; #10;
a = 1; b = 0; c = 1; d = 0; s0 = 1; s1 = 0; #10;
a = 1; b = 0; c = 1; d = 1; s0 = 1; s1 = 0; #10;
a = 1; b = 1; c = 0; d = 0; s0 = 1; s1 = 0; #10;
a = 1; b = 1; c = 0; d = 1; s0 = 1; s1 = 0; #10;
a = 1; b = 1; c = 1; d = 0; s0 = 1; s1 = 0; #10;
a = 1; b = 1; c = 1; d = 1; s0 = 1; s1 = 0; #10;

a = 0; b = 0; c = 0; d = 0; s0 = 0; s1 = 1; #10;
a = 0; b = 0; c = 0; d = 1; s0 = 0; s1 = 1; #10;
a = 0; b = 0; c = 1; d = 0; s0 = 0; s1 = 1; #10;
a = 0; b = 0; c = 1; d = 1; s0 = 0; s1 = 1; #10;
a = 0; b = 1; c = 0; d = 0; s0 = 0; s1 = 1; #10;
a = 0; b = 1; c = 0; d = 1; s0 = 0; s1 = 1; #10;
a = 0; b = 1; c = 1; d = 0; s0 = 0; s1 = 1; #10;
a = 0; b = 1; c = 1; d = 1; s0 = 0; s1 = 1; #10;
a = 1; b = 0; c = 0; d = 0; s0 = 0; s1 = 1; #10;
a = 1; b = 0; c = 0; d = 1; s0 = 0; s1 = 1; #10;
a = 1; b = 0; c = 1; d = 0; s0 = 0; s1 = 1; #10;
a = 1; b = 0; c = 1; d = 1; s0 = 0; s1 = 1; #10;
a = 1; b = 1; c = 0; d = 0; s0 = 0; s1 = 1; #10;
a = 1; b = 1; c = 0; d = 1; s0 = 0; s1 = 1; #10;
a = 1; b = 1; c = 1; d = 0; s0 = 0; s1 = 1; #10;
a = 1; b = 1; c = 1; d = 1; s0 = 0; s1 = 1; #10;

a = 0; b = 0; c = 0; d = 0; s0 = 1; s1 = 1; #10;
a = 0; b = 0; c = 0; d = 1; s0 = 1; s1 = 1; #10;
a = 0; b = 0; c = 1; d = 0; s0 = 1; s1 = 1; #10;
a = 0; b = 0; c = 1; d = 1; s0 = 1; s1 = 1; #10;
a = 0; b = 1; c = 0; d = 0; s0 = 1; s1 = 1; #10;
a = 0; b = 1; c = 0; d = 1; s0 = 1; s1 = 1; #10;
a = 0; b = 1; c = 1; d = 0; s0 = 1; s1 = 1; #10;
a = 0; b = 1; c = 1; d = 1; s0 = 1; s1 = 1; #10;
a = 1; b = 0; c = 0; d = 0; s0 = 1; s1 = 1; #10;
```

```
        a = 1; b = 0; c = 0; d = 1; s0 = 1; s1 = 1; #10;
        a = 1; b = 0; c = 1; d = 0; s0 = 1; s1 = 1; #10;
        a = 1; b = 0; c = 1; d = 1; s0 = 1; s1 = 1; #10;
        a = 1; b = 1; c = 0; d = 0; s0 = 1; s1 = 1; #10;
        a = 1; b = 1; c = 0; d = 1; s0 = 1; s1 = 1; #10;
        a = 1; b = 1; c = 1; d = 0; s0 = 1; s1 = 1; #10;
        a = 1; b = 1; c = 1; d = 1; s0 = 1; s1 = 1; #10;
    end
endmodule
```

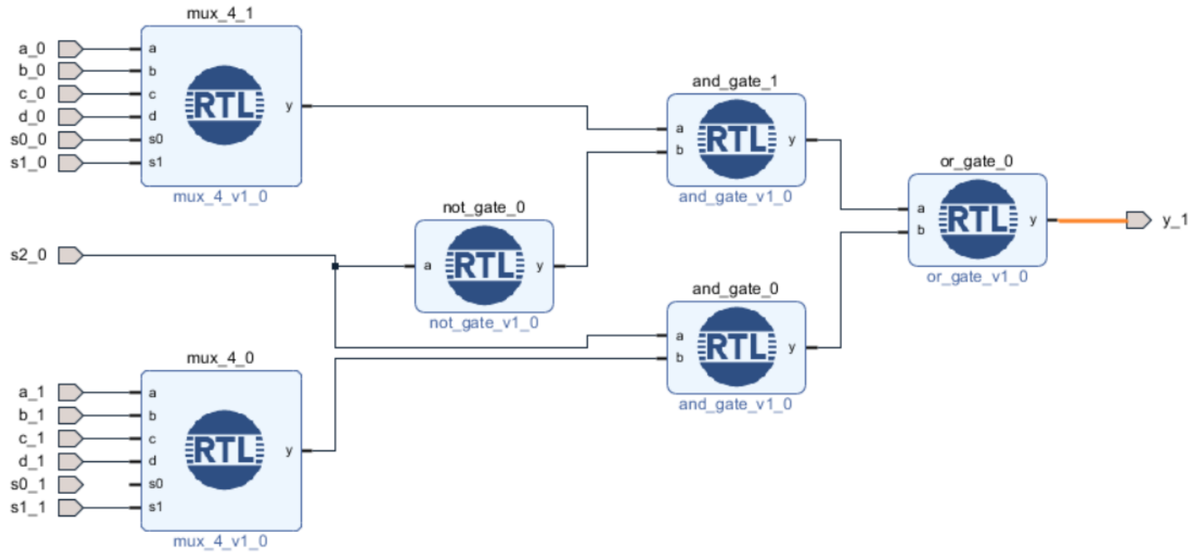
8-1 Multiplexer

Module:

```
module eight_to_one_mux(
    input[7:0] a,
    input[2:0] select,
    output y
);
    logic result_first_mux;
    logic result_second_mux;

    logic first_half;
    logic second_half;

    four_to_one_mux
first_mux(.a(a[0]),.b(a[1]),.c(a[2]),.d(a[3]),.s0(select[0]),.s1(sel
ect[1]),.y(result_first_mux));
    four_to_one_mux
second_mux(.a(a[4]),.b(a[5]),.c(a[6]),.d(a[7]),.s0(select[0]),.s1(se
lect[1]),.y(result_second_mux));

    and first(first_half,result_first_mux,~select[2]);
    and second(second_half,result_second_mux,select[2]);

    or(y,first_half,second_half);
endmodule
```

TestBench:

```
module eight_to_one_mux_sim( );
    logic [7:0] a = 8'b00000000;
    logic [2:0] select = 3'b000;
    logic y;
    eight_to_one_mux dut(.a(a[7:0]),.select(select[2:0]),.y(y));
    initial begin
        for(int i = 0; i < 64; i++) begin
            for(int j = 0; j < 8; j++) begin
                #5;
                select = select + 1;
            end
            assign select = 3'b000;
            a = a + 1;
            #5;
        end
    end
endmodule
```


mux_4_1
a_0
b_0
c_0
d_0
s0_0
s1_0
a
b
c
d
s0
s1
RTL
y
mux_4_v1_0
and_gate_1
a
b
RTL
y
and_gate_v1_0
or_gate_0
a
b
RTL
y
or_gate_v1_0
y_1
not_gate_0
s2_0
a
RTL
y
not_gate_v1_0
and_gate_0
a
b
RTL
y
and_gate_v1_0
mux_4_0
a_1
b_1
c_1
d_1
s0_1
s1_1
a
b
c
d
s0
s1
RTL
y
mux_4_v1_0

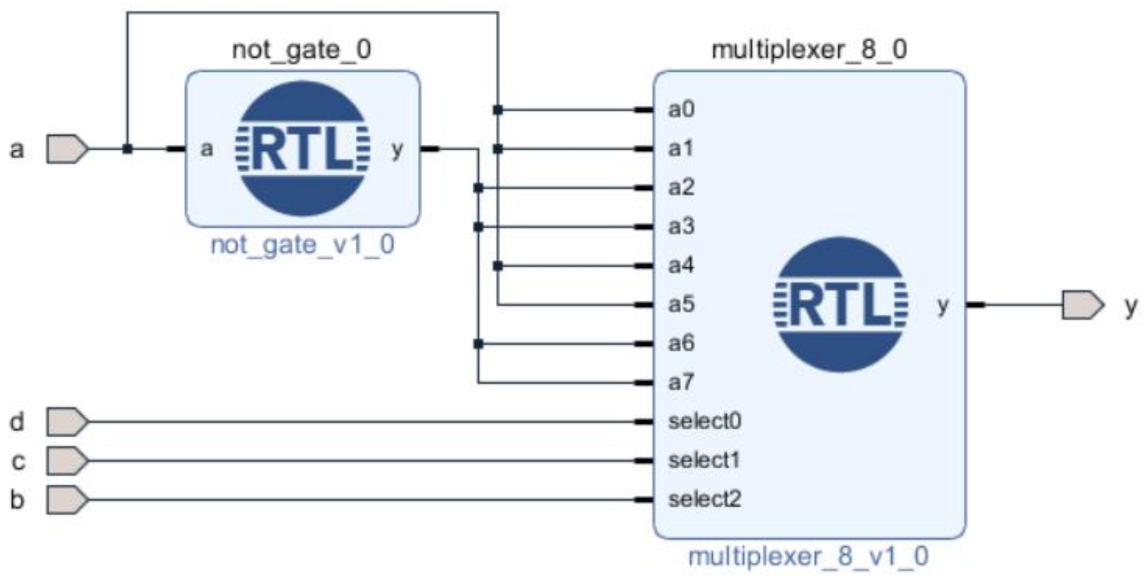
Lab Function

Module:

```
module lab_func(
    input a,
    input b,
    input c,
    input d,
    output y
);
    logic[7:0] bitwise_a;
    logic[3:0] bitwise_select;

    assign bitwise_a[7:0] = {~a,~a,a,a,~a,~a,a,a};
    assign bitwise_select[3:0] = {b,c,d};

    eight_to_one_mux
F(.select(bitwise_select[3:0]),.a(bitwise_a[7:0]),.y(y));
endmodule
```

TestBench:

```
module lab_func_sim( );
    logic a, b, c, d, y;
    lab_func dut(.a(a),.b(b),.c(c),.d(d),.y(y));
    initial begin
        a = 0; b = 0; c = 0; d = 0; #10;
        a = 0; b = 0; c = 0; d = 1; #10;
        a = 0; b = 0; c = 1; d = 0; #10;
        a = 0; b = 0; c = 1; d = 1; #10;
        a = 0; b = 1; c = 0; d = 0; #10;
        a = 0; b = 1; c = 0; d = 1; #10;
        a = 0; b = 1; c = 1; d = 0; #10;
        a = 0; b = 1; c = 1; d = 1; #10;
        a = 1; b = 0; c = 0; d = 0; #10;
        a = 1; b = 0; c = 0; d = 1; #10;
        a = 1; b = 0; c = 1; d = 0; #10;
        a = 1; b = 0; c = 1; d = 1; #10;
        a = 1; b = 1; c = 0; d = 0; #10;
        a = 1; b = 1; c = 0; d = 1; #10;
        a = 1; b = 1; c = 1; d = 0; #10;
        a = 1; b = 1; c = 1; d = 1; #10;
    end
endmodule
```



not_gate_0
multiplexer_8_0
a
a
RTL
y
not_gate_v1_0
a0
a1
a2
a3
a4
a5
a6
a7
select0
select1
select2
RTL
y
y
d
c
b
multiplexer_8_v1_0