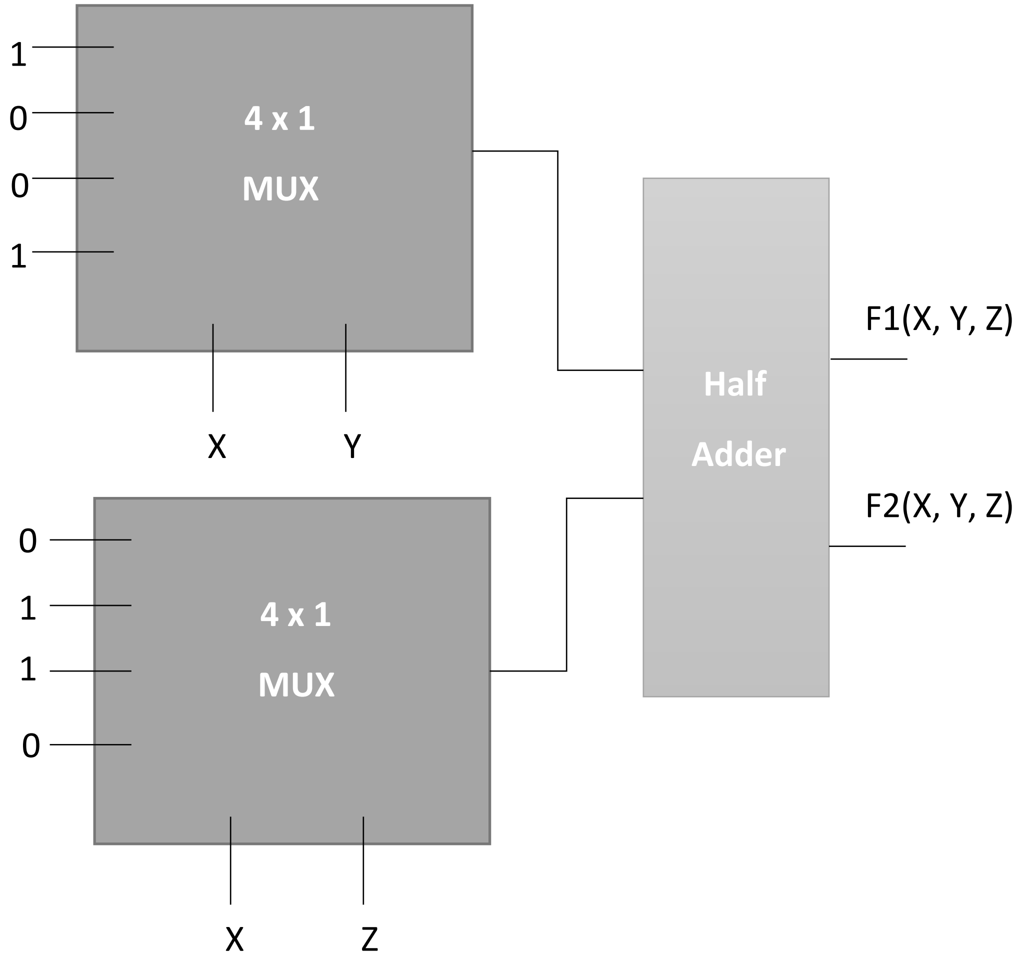
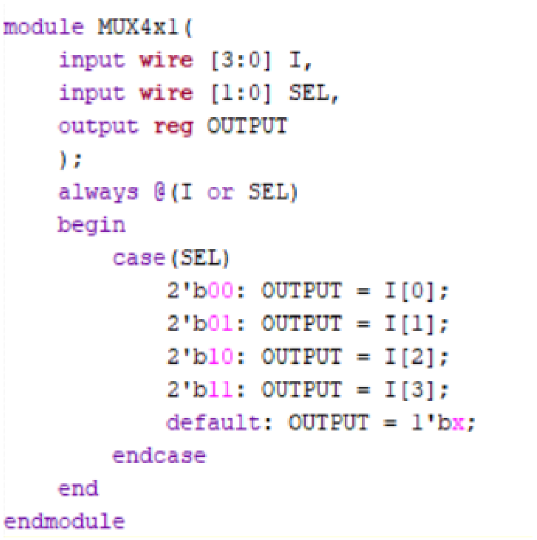
[Introduction to Computer Architecture'21] Lab: Verilog Syntax 3

Exercise 1: ***Implement*** and ***Simulate*** the output F1 (X, Y, Z) and F2 (X, Y, Z) of the below circuit: (*create a **Testbench** for the below case*)

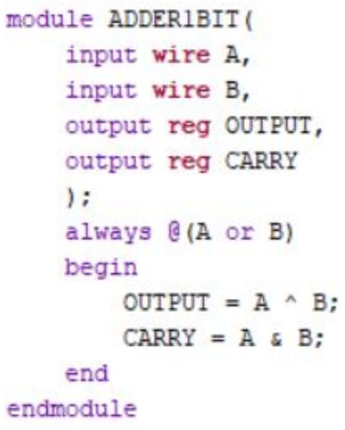


1- Create a Module for 4x1 Multiplexer



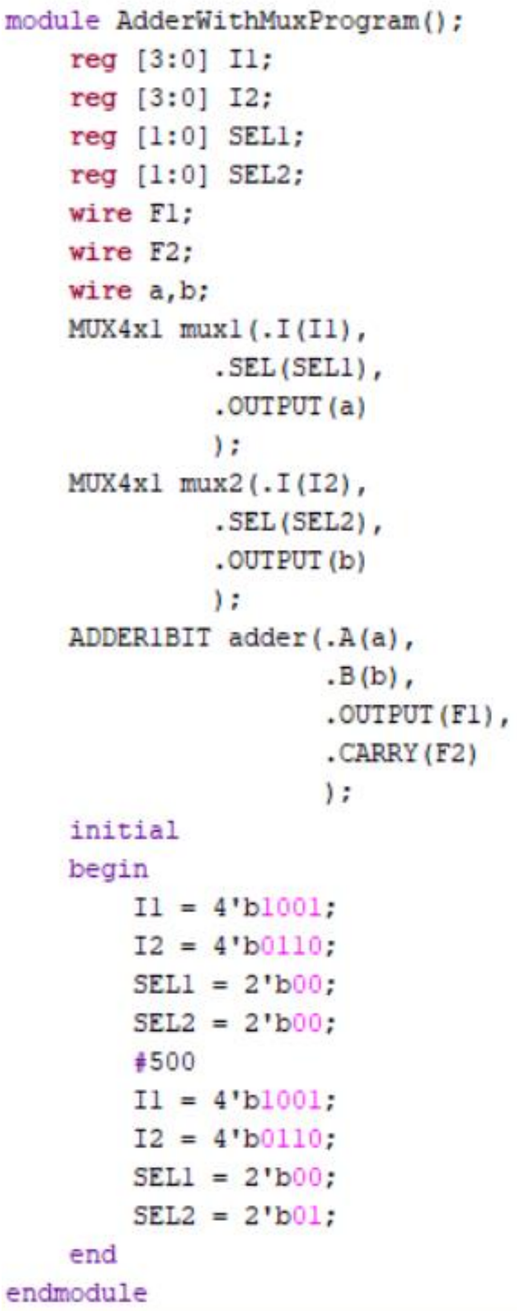
```
module MUX4x1(
    input wire [3:0] I,
    input wire [1:0] SEL,
    output reg OUTPUT
    );
    always @(I or SEL)
    begin
        case(SEL)
            2'b00: OUTPUT = I[0];
            2'b01: OUTPUT = I[1];
            2'b10: OUTPUT = I[2];
            2'b11: OUTPUT = I[3];
            default: OUTPUT = 1'bx;
        endcase
    end
endmodule
```

2- Create a Module for 1-bit Adder with carry



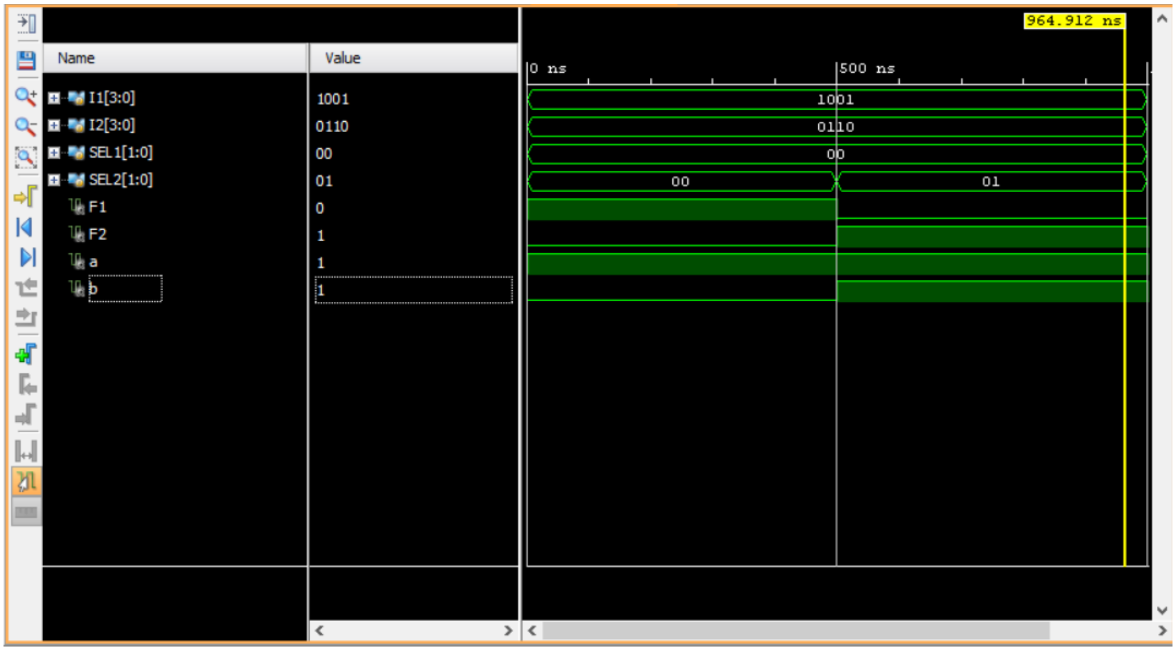
```
module ADDER1BIT(
    input wire A,
    input wire B,
    output reg OUTPUT,
    output reg CARRY
    );
    always @(A or B)
    begin
        OUTPUT = A ^ B;
        CARRY = A & B;
    end
endmodule
```

3- Create a testbench to Run & Simulate the program



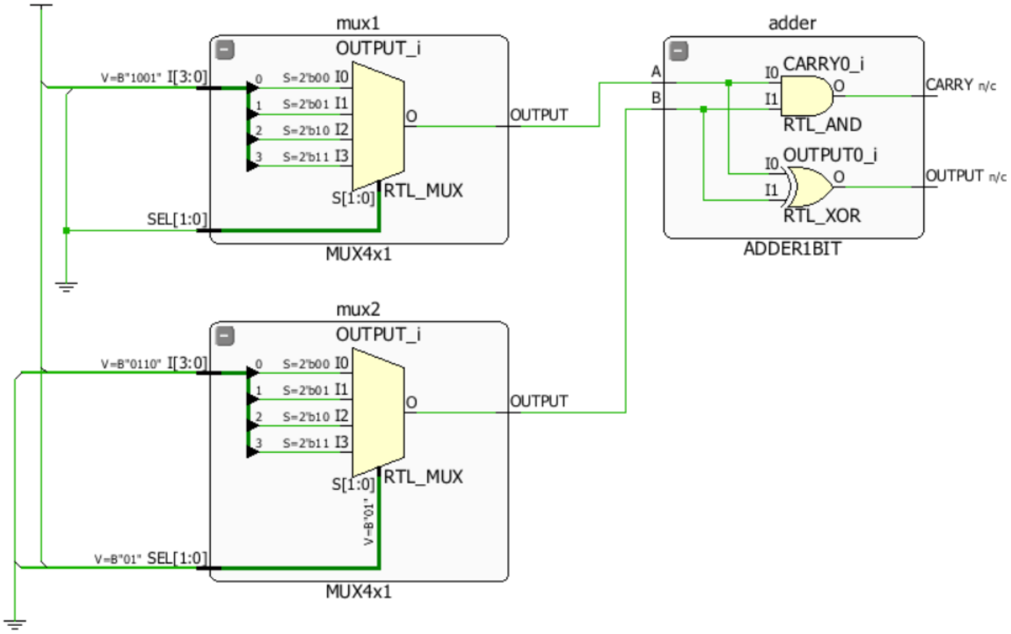
```
module AdderWithMuxProgram();
    reg [3:0] I1;
    reg [3:0] I2;
    reg [1:0] SEL1;
    reg [1:0] SEL2;
    wire F1;
    wire F2;
    wire a,b;
    MUX4x1 mux1(.I(I1),
             .SEL(SEL1),
             .OUTPUT(a)
             );
    MUX4x1 mux2(.I(I2),
             .SEL(SEL2),
             .OUTPUT(b)
             );
    ADDER1BIT adder(.A(a),
                    .B(b),
                    .OUTPUT(F1),
                    .CARRY(F2)
                    );
    initial
    begin
        I1 = 4'b1001;
        I2 = 4'b0110;
        SEL1 = 2'b00;
        SEL2 = 2'b00;
        #500
        I1 = 4'b1001;
        I2 = 4'b0110;
        SEL1 = 2'b00;
        SEL2 = 2'b01;
    end
endmodule
```

4- Run Simulation



Name	Value
I1[3:0]	1001
I2[3:0]	0110
SEL1[1:0]	00
SEL2[1:0]	01
F1	0
F2	1
a	1
b	1

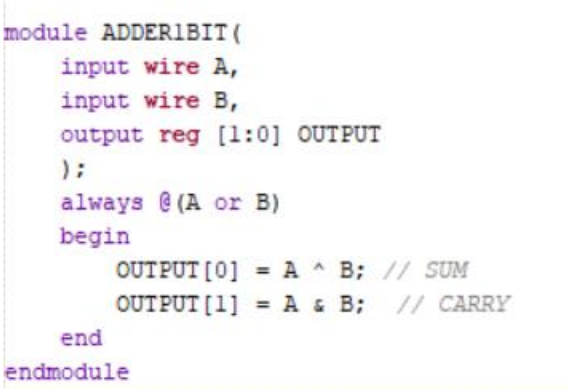
5- Run RTL Schematic



Exercise 2: ***Implement*** and ***Simulate*** a full subtractor circuit with 3 inputs x, y, z and two outputs D and B that compute (x-y-z). D is difference and B is new borrow. (*create a **Testbench***)

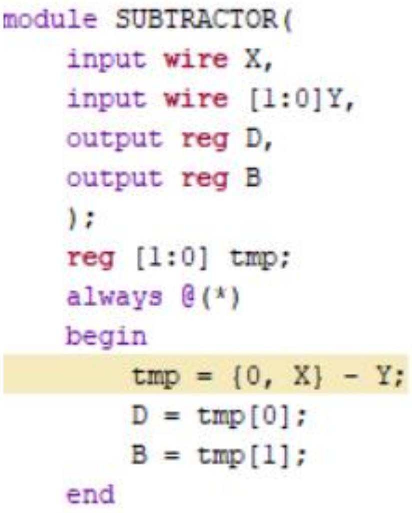
Simpler to consider it (X – (Y+Z))

1- Create a Module for 1-bit Adder



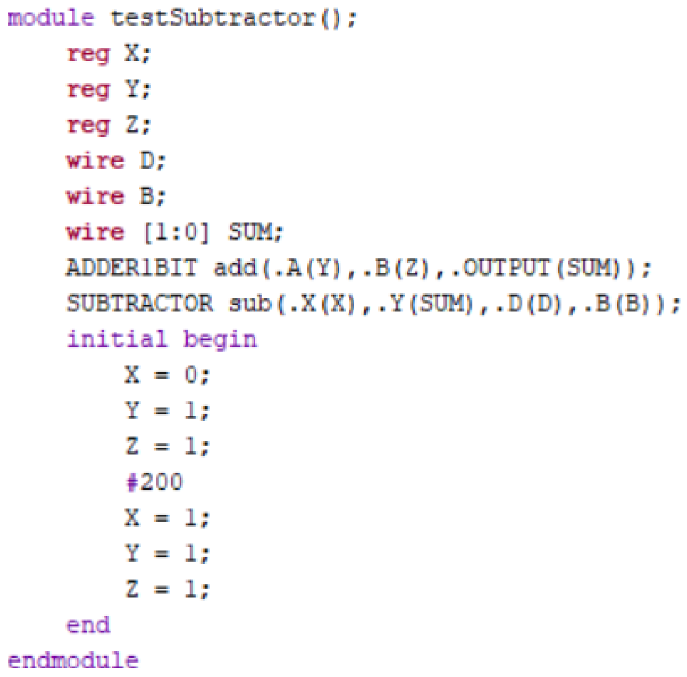
```
module ADDER1BIT(
    input wire A,
    input wire B,
    output reg [1:0] OUTPUT
    );
    always @(A or B)
    begin
        OUTPUT[0] = A ^ B; // SUM
        OUTPUT[1] = A & B;  // CARRY
    end
endmodule
```

2- Create a Module for 1-bit Subtract



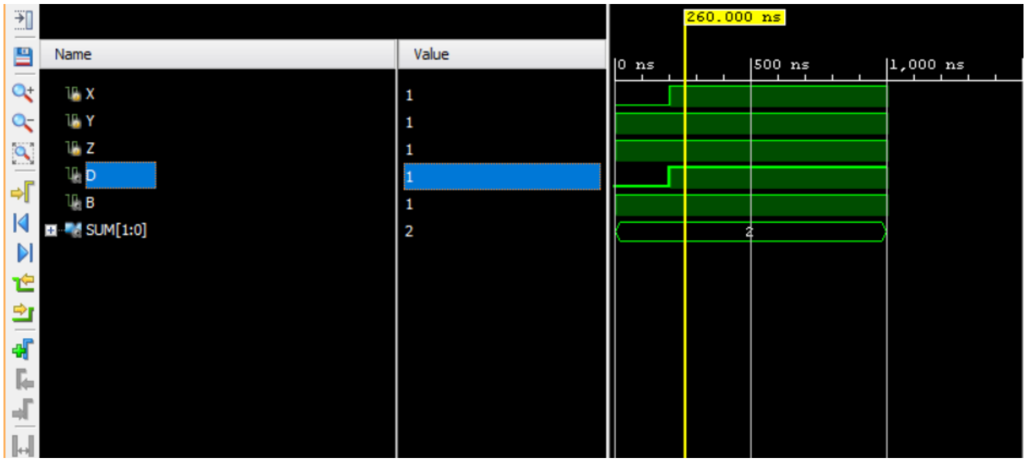
```
module SUBTRACTOR(
    input wire X,
    input wire [1:0]Y,
    output reg D,
    output reg B
    );
    reg [1:0] tmp;
    always @(*)
    begin
        tmp = {0, X} - Y;
        D = tmp[0];
        B = tmp[1];
    end
```

3- Create a Module for Testbench to Run & Simulate the program



```
module testSubtractor();
    reg X;
    reg Y;
    reg Z;
    wire D;
    wire B;
    wire [1:0] SUM;
    ADDER1BIT add(.A(Y),.B(Z),.OUTPUT(SUM));
    SUBTRACTOR sub(.X(X),.Y(SUM),.D(D),.B(B));
    initial begin
        X = 0;
        Y = 1;
        Z = 1;
        #200
        X = 1;
        Y = 1;
        Z = 1;
    end
endmodule
```

4- Run Simulation

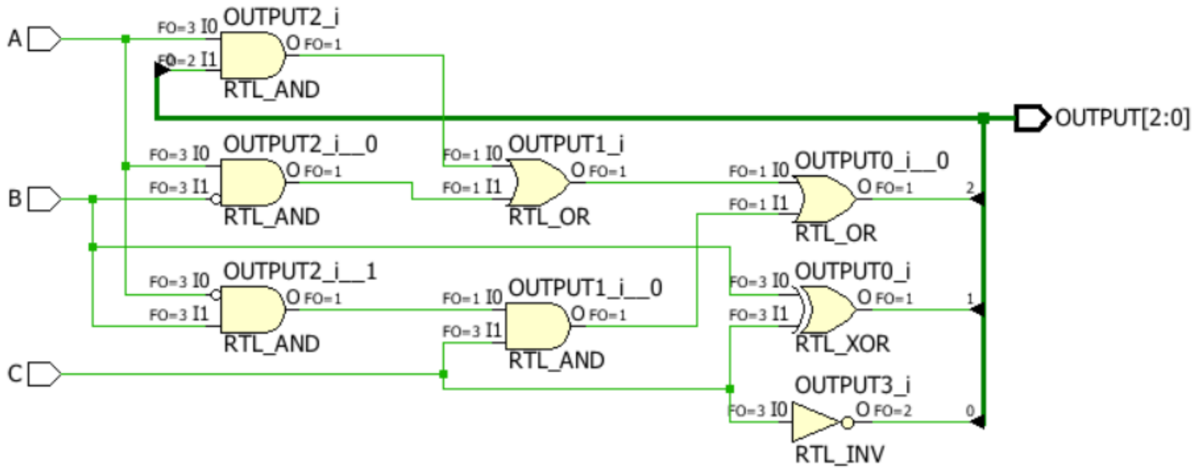


Name	Value
X	1
Y	1
Z	1
D	1
B	1
SUM[1:0]	2

Exercise 3 – (your turn): ***Implement*** and ***Simulate*** a circuit that adds one to 3 inputs A, B and C except for the input 111 the output will be zero.

(*Create a* ***Testbench*** *for this case: A = 0, B = 1 and C = 1*)

Your RTL Design should be as the below RTL:



Verilog + ZYBO Z7 board Help:

- Check this link for Verilog syntax: https://www.nandland.com/verilog/tutorials/index.html
- Check this link for ZYBO Z7 board info.: https://digilent.com/reference/programmable-logic/zybo/start

If you need any help regarding anything about the course, ask:

- Engr. Ahmad M. Abdel-Hafeez: akassem@nu.edu.eg
- Engr. Mohammad Rady: mrady@nu.edu.eg