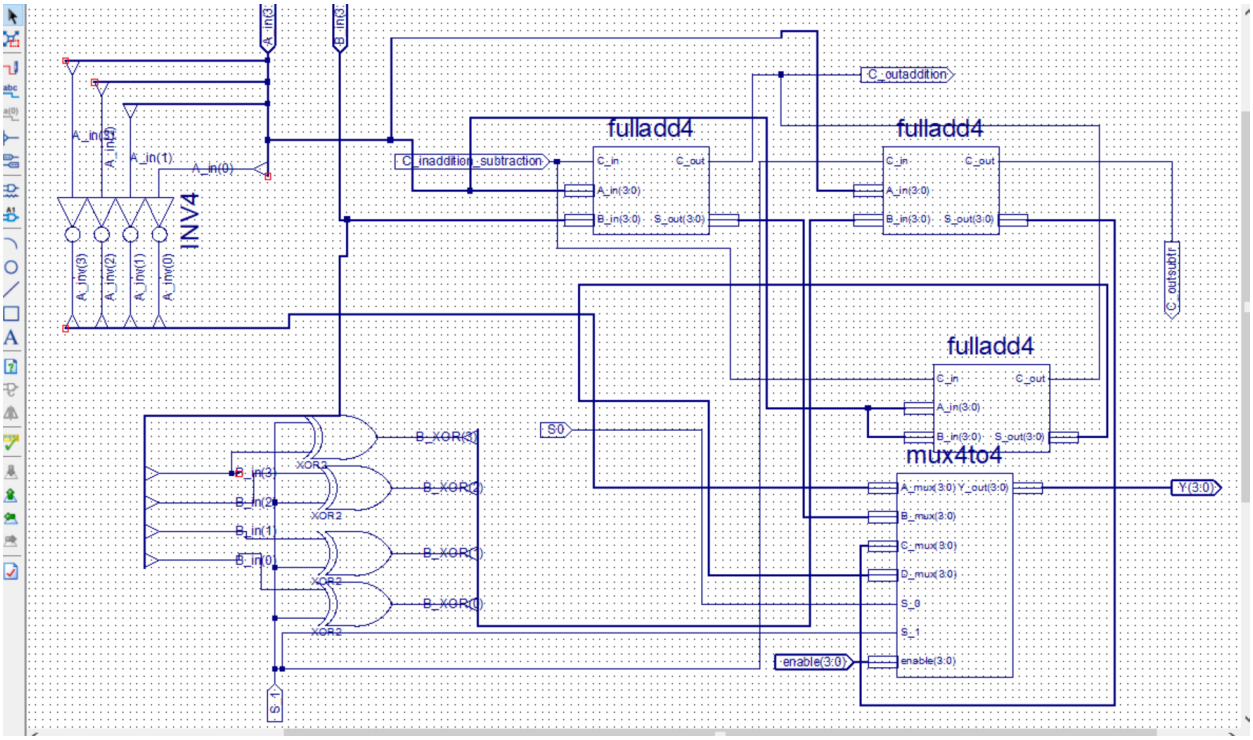
CDA 4203L
Computer System Design Lab

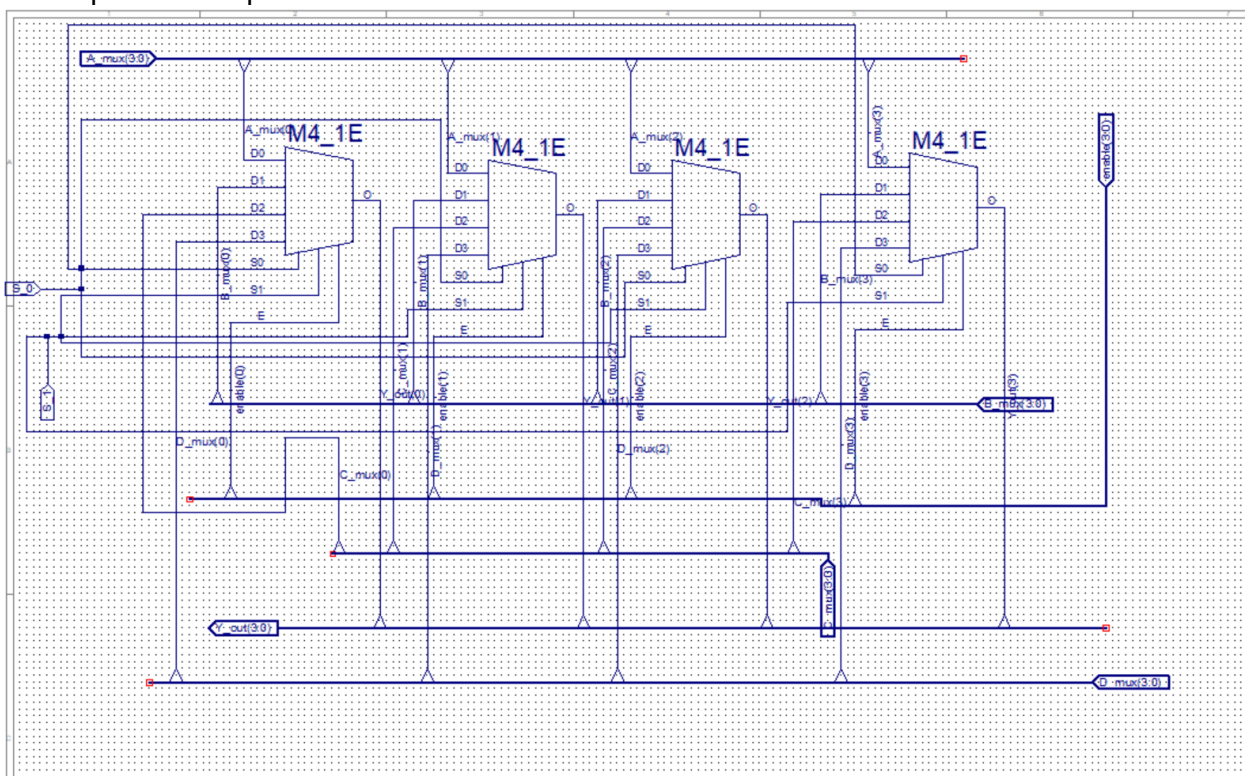
Lab 1 Report
ALU Design

Today's Date:	01/28/2022
Your Name:	Ahmed El Maliki
Your U Number:	U18019330
No. of Hours Spent:	10
Exercise Difficulty: (Easy, Average, Hard)	Average
Any Other Feedback:	

Question 1: ALU Schematic



4 bit input 4 bit output 2 bit select line MUX:



Question 1: Briefly describe how your design works.

This design uses the 4bit adder introduced in the tutorial

In this design, a custom mux had to be designed to accommodate the required tasks.

The MUX was designed separately and imported as a symbol.

The mux takes 4 lines each of 4-bit input and outputs a 4 bit output. The select line are 2 bits.

Now that the mux issues have been solved, we focus on the ALU.

When the select lines are 00, the ALU choses the first input or the “00” line, that line is an inverted A.
When the lines are 01, the ALU selects the second line of the “01” line, that line is the output value of a full adder of A-B, the B line gets converted to its negative value on 2s compliment by Xoring every of its input with the select line S1 then gets inputted to a 4 bit adder along with A. Output is now A-B.

When the line 10, third mux line gets selected, the output is A+B.(result coming from a different 4 bit Adder)
When line is 11, forth line is selected and the value of the output is 2A. (A get added to it self in a diff 4 bit adder)

Question 2: Simulation Waveforms (add as many pages as you need).

Include *atleast* two test vectors per function. For example, demonstrate through the waveforms, that the ALU performs inversion correctly on two inputs, say, 0010 and 1111.

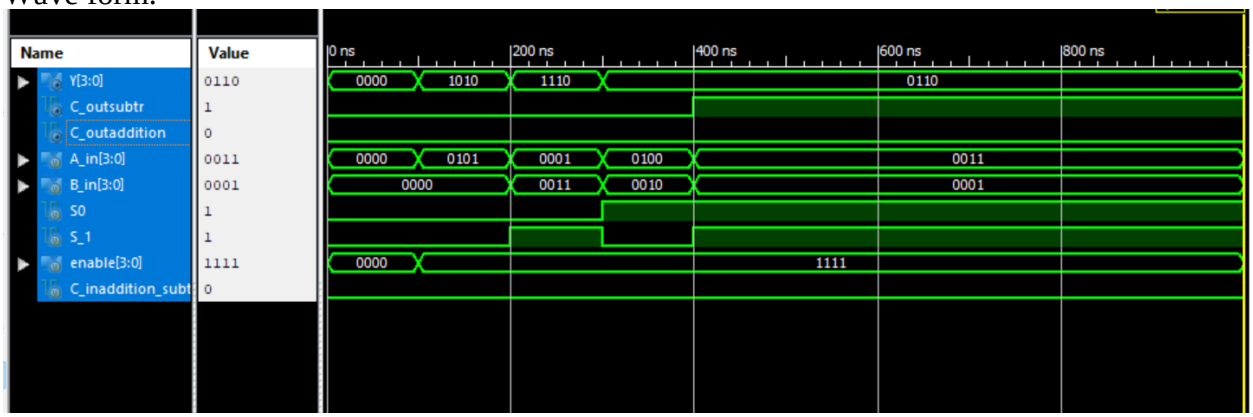
First I want to demonstrate that the ALU works for all the functions, bellow is the test bench and wave form for all the function.

Test bench:

```
// Verilog Test Fixture Template
`timescale 1ns / 1ps

module ALU_ALU_sch_tb();

  reg [3:0] A_in;
  reg [3:0] B_in;
  reg S0;
  reg S_1;
  reg [3:0] enable;
        reg C_inaddition_subtraction;


// Output
  wire [3:0] Y;
        wire C_outsubtr;
        wire C_outaddition;

// Bidirs

// Instantiate the UUT
  ALU UUT (
                .Y(Y[3:0]),
                .A_in(A_in[3:0]),
                .B_in(B_in[3:0]),
                .S0(S0),
                .S_1(S_1),
                .enable(enable[3:0]),
                .C_outsubtr,
                .C_outaddition,
                .C_inaddition_subtraction

  );

     initial begin
```

```
A_in =0;
B_in = 0;
S0=0;
S_1=0;
enable=0;
C_inaddition_subtraction=0;

#100;
A_in =5;
B_in = 0;
S0=0;
S_1=0;
enable=15;
C_inaddition_subtraction=0;

#100;
A_in = 1;
B_in = 3;
S0 = 0;
S_1 = 1;
enable=15;
C_inaddition_subtraction=0;

#100;
A_in =4;
B_in = 2;
S0 = 1;
S_1 = 0;
enable=15;
C_inaddition_subtraction=0;

#100;
A_in =3;
B_in= 1;
S0 = 1;
S_1 = 1;
enable=15;
C_inaddition_subtraction=0;

  end
endmodule
```

Wave form:



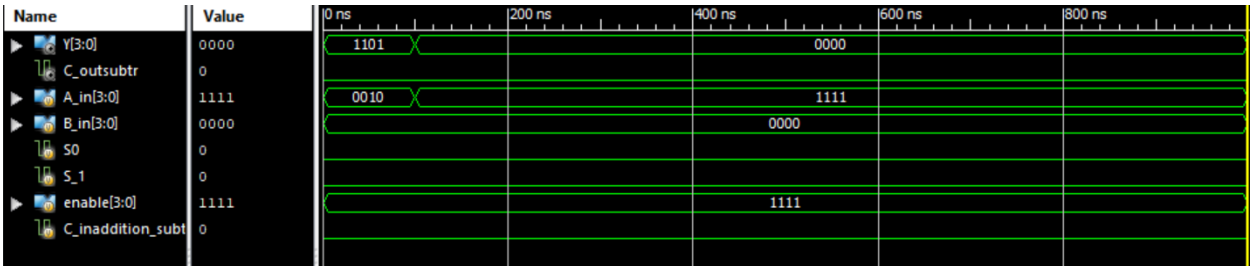
Now, as requested per the instruction for values 0010 1111

Inversion code:

```
`timescale 1ns / 1ps

module ALU_ALU_sch_tb();

  reg [3:0] A_in;
  reg [3:0] B_in;
  reg S0;
  reg S_1;
  reg [3:0] enable;
        reg C_inaddition_subtraction;


// Output
  wire [3:0] Y;
        wire C_outsubtr;
        wire C_outaddition;

// Bidirs

// Instantiate the UUT
  ALU UUT (
                .Y(Y[3:0]),
                .A_in(A_in[3:0]),
                .B_in(B_in[3:0]),
                .S0(S0),
                .S_1(S_1),
                .enable(enable[3:0]),
                .C_outsubtr,
                .C_outaddition,
                .C_inaddition_subtraction
```

```
);

    initial begin

                A_in =2;
                B_in = 0;
                S0=0;
                S_1=0;
                enable=15;
                C_inaddition_subtraction=0;

                #100;
                A_in =15;
                B_in = 0;
                S0=0;
                S_1=0;
                enable=15;
                C_inaddition_subtraction=0;



                  end
endmodule
```

ScreenShot:

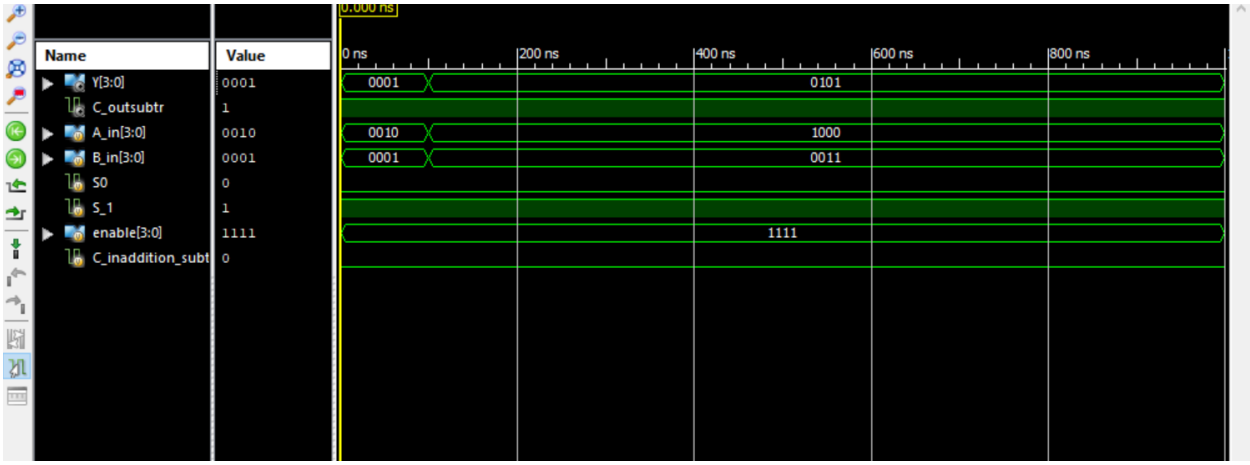


Subtraction:
(code bellow only affects change not the whole code)

```
                A_in =2;
                B_in = 1;
                S0=0;
                S_1=1;
                enable=15;
                C_inaddition_subtraction=0;

                #100;
                A_in =8;
```

B_in = 3;
S0=0;
S_1=1;
enable=15;
C_inaddition_subtraction=0;

Wave form:

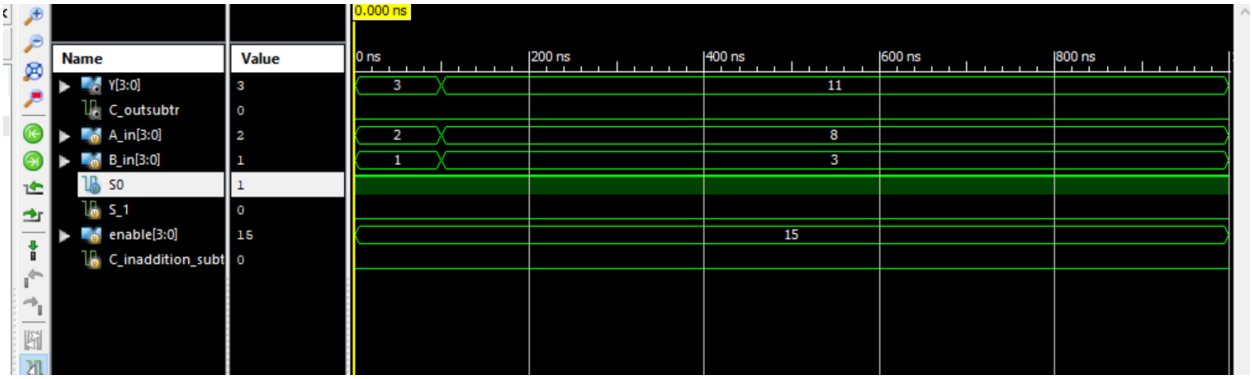


Addition:
Code:

A_in =2;
B_in = 1;
S0=1;
S_1=0;
enable=15;
C_inaddition_subtraction=0;

#100;
A_in =8;
B_in = 3;
S0=1;
S_1=0;
enable=15;
C_inaddition_subtraction=0;

Waveform:



Double:

Code:

```
A_in =2;
B_in = 0;
S0=1;
S_1=1;
enable=15;
C_inaddition_subtraction=0;

#100;
A_in =5;
B_in = 0;
S0=1;
S_1=1;
enable=15;
C_inaddition_subtraction=0;
```

Waveform: