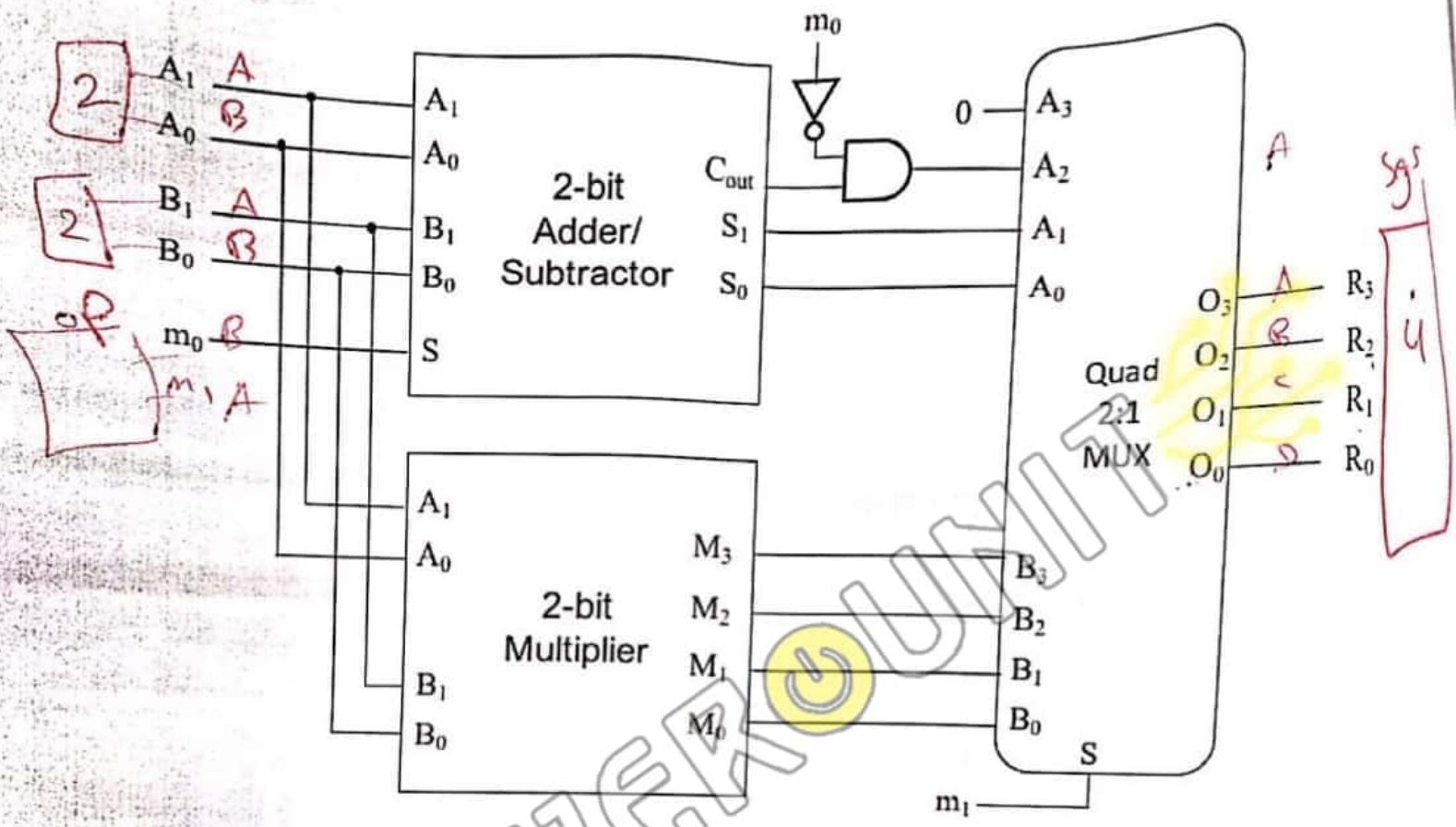


Figure1. 2-bit ALU Circuit

m_1	m_0	Arithmetic Operation
0	0	Addition (+)
0	1	Subtraction (-)
1	x	Multiplication (*)

Figure2. ALU Operations

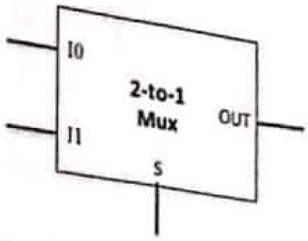
In this experiment, our goal is to design a simple 2-bit ALU combinational circuit that performs the following unsigned operations: addition, subtraction and multiplication. As shown by Figure 1, the ALU circuit has two 2-bit unsigned input numbers A{A_1A_0} and B {B_1B_0} and a 4-bit unsigned output number R{$R_3R_2R_1R_0$}. The ALU circuit also has 2-bit control signal m_1m_0 that are used to choose the ALU operation (see Figure 2).

The experiment is divided into four parts.First, you will implement the quad 2-to-1 multiplexer.Second, you will implement a 2-bit ripple-carry adder/subtractor. Third, you will implement a 2-bit multiplier. Finally, you will combine all the aforementioned modules to implement the ALU circuit.

Part1: Quad 2-to-1 Multiplexer

a) Write the Boolean equation of the output of the 2-to-1 Multiplexer:

OUT= $\bar{S}I_0 + S\bar{I}_1$



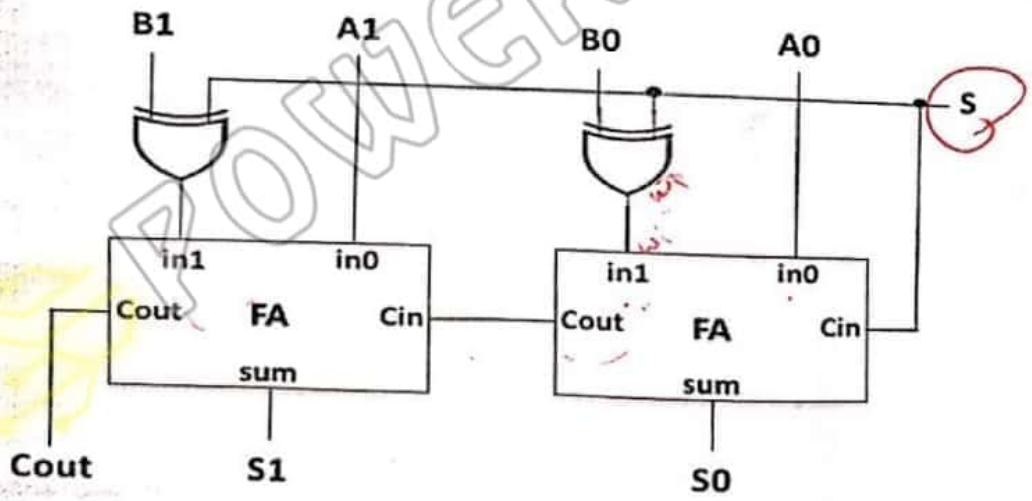
b) Inside Mux2to1.v file, write a behavioral Verilog code to implement the 2-to-1 multiplexer module. This module has 3 inputs (I0, I1 and S) and one output (OUT).

c) Using four instances of 2-to-1 multiplexer, design a circuit of a quad 2-to-1 multiplexer. (PreLab)

d) Inside QuadMux2to1.v file, write a structural Verilog code to implement the quad 2-to-1 multiplexer module based on your design in part *c*. This module has nine inputs (A3, A2, A1, A0, B3, B2, B1, B0 and S) and four outputs (O3, O2, O1 and O0). (PreLab, Write the module code on your computer and print it)

Part2:The 2-bit Adder/Subtractor

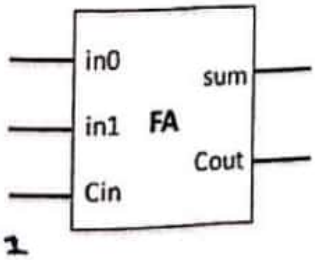
An adder/subtractor circuit can be built using two cascaded full adders (see the figure below).



a) Write the Boolean equations of the outputs of the full adder (FA) circuit: (PreLab)

sum= $Cin \oplus in1 \oplus in0$

Cout= $in0 \cdot in1 + in_0 \cdot cin + in_1 \cdot cin_1$

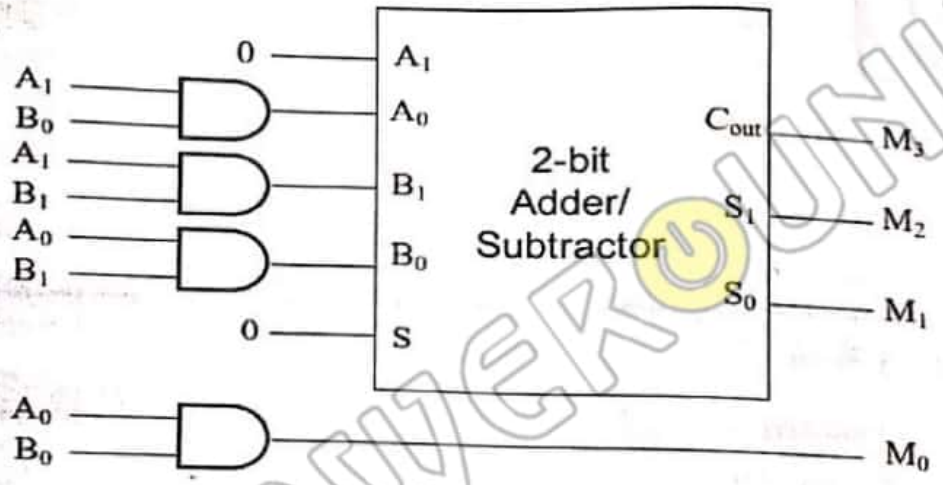


b) Inside FA.v file, write a behavioral Verilog code to implement the full adder module. This module has 3 inputs (in0, in1 and Cin) and 2outputs (sum and Cout).

c) Inside TwoBitAdderSubtractor.v file, write a structural Verilog code to implement the 2-bit adder/subtractor module shown by the above figure. This module has five inputs (A0,A1, B0, B1, and S) and three outputs (S0, S1, and Cout).

Part3: The 2-bit Multiplier.

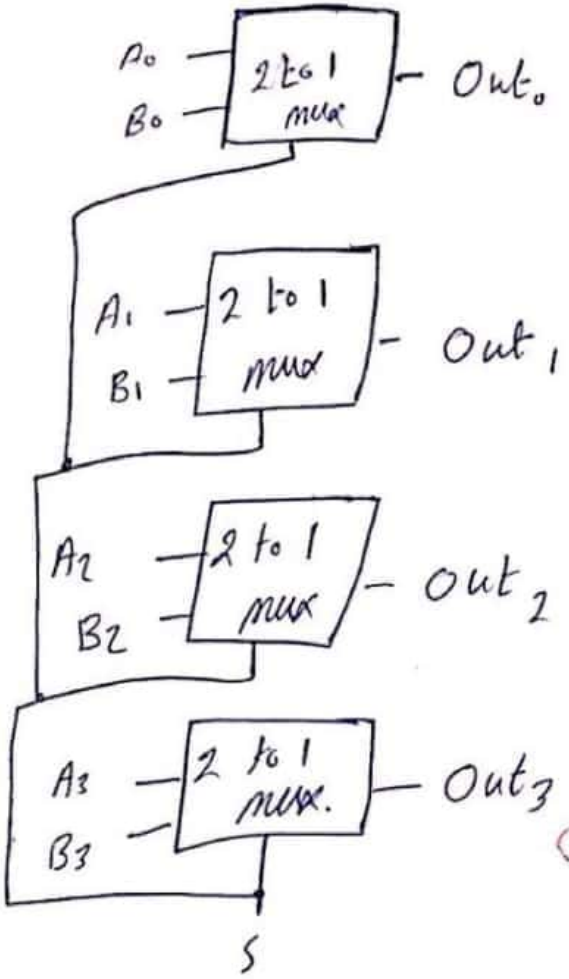
A 2-bit multiplication circuit can be built using four instances of AND gates and one instance of the 2-bit adder/subtractor circuit we implemented in Part1.



a) Inside Mul.v file, write a structural Verilog code to implement the 2-bitmultipliermodule shown by the above figure. This module has four inputs (A0, A1, B0, and B1) and four outputs (M0, M1, M2 and M3).

Part4:ALU circuit design.

a) Inside ALU.v file, write a structural Verilog code to implement the full ALU circuit module. For convenience, we will use **four 7-segment displays** to show the values of the operand A, operand B, the operation, and the result. Inside segdriver2.vand segdriver4.vfiles, we already have written the two 7-segement driver modules that you need.
The ALU circuit has the following inputs and outputs:
   - 6-inputs (A0, A1, B0, B1, m0, m1)
   - 28-Outputs:
     - The outputs 'a0, b0, c0, d0, e0, f0, g0', which will be connected to the 7-segemnt display of the result $\{R_3R_2R_1R_0\}$.
     - The outputs 'a1, b1, c1, d1, e1, f1, g1', which will be connected to the 7-segemnt display of operand B $\{B_1B_0\}$.



Module quadmux ($A_3, A_2, A_1, A_0, B_3, B_2, B_1, B_0, out_3, out_2, out_1, out_0$)

input $A_3, A_2, A_1, A_0, B_3, B_2, B_1, B_0$;

output $out_3, out_2, out_1, out_0$;

Mux$_1$ (B_0, A_0, S_1, out_0) ;

Mux$_1$ (B_1, A_1, S_1, out_1) ;

Mux$_1$ (B_2, A_2, S_1, out_2) ;

Mux$_1$ (B_3, A_3, S_1, out_3) ;

end module

- The outputs 'a2, b2, c2, d2, e2, f2, g2', which will be connected to the operation selection lines $\{m_1 m_0\}$.
- The outputs 'a3, b3, c3, d3, e3, f3, g3', which will be connected to the 7-segemnt display of operand A $\{A_1 A_0\}$.

b) Make the pins assignment for the ALU circuit's inputs and outputs, as follows:

Hex (0): Result		
a0	oHEX0_D[0]	PIN_AE8
b0	oHEX0_D[1]	PIN_AF9
c0	oHEX0_D[2]	PIN_AH9
d0	oHEX0_D[3]	PIN_AD10
e0	oHEX0_D[4]	PIN_AF10
f0	oHEX0_D[5]	PIN_AD11
g0	oHEX0_D[6]	PIN_ AD12
Hex (1): Operand B		
a1	oHEX1_D[0]	PIN_AG13
b1	oHEX1_D[1]	PIN_AE16
c1	oHEX1_D[2]	PIN_AF16
d1	oHEX1_D[3]	PIN_AG16
e1	oHEX1_D[4]	PIN_AE17
f1	oHEX1_D[5]	PIN_AF17
g1	oHEX1_D[6]	PIN_AD17
Hex (2): Operation		
a2	oHEX2_D[0]	PIN_AE7
b2	oHEX2_D[1]	PIN_AF7
c2	oHEX2_D[2]	PIN_AH5
d2	oHEX2_D[3]	PIN_AG4
e2	oHEX2_D[4]	PIN_AB18
f2	oHEX2_D[5]	PIN_AB19
g2	oHEX2_D[6]	PIN_AE19
Hex (3): Operand A		
a3	oHEX3_D[0]	PIN_P6
b3	oHEX3_D[1]	PIN_P4
c3	oHEX3_D[2]	PIN_N10
d3	oHEX3_D[3]	PIN_N7
e3	oHEX3_D[4]	PIN_M8
f3	oHEX3_D[5]	PIN_M7
g3	oHEX3_D[6]	PIN_M6
Input Switches		
B0	iSW[1]	PIN_AB26
B1	iSW[2]	PIN_AB25
m0	iSW[3]	PIN_AC27
m1	iSW[4]	PIN_AC26
A0	iSW[5]	PIN_AC24
A1	iSW[6]	PIN_AC23