8-Bit Arithmetic Logic Unit (ALU)

Specification:

An 8-bit arithmetic logic unit (ALU) is a combinational circuit which operates on two 8-bit input buses based on selection inputs. The ALU performs common arithmetic (addition and subtraction) and logic (AND, INV, XOR, and OR) functions. These operations are common to all computer systems and thus are an essential part of computer architecture.

Inputs and outputs are as follows:

Name	Width	Input/Output	Description
A	8	Input	first operand
B	8	Input	second operand
c_in	1	Input	used to get more operations
alu_sel	3	Input	chooses operation alu performs
enable	1	Input	enable alu
M	8	Output	result of operation performed
m7	1	Output	sign bit if signed numbers
v	1	Output	overflow assuming signed numbers
c_out	1	Output	carry out

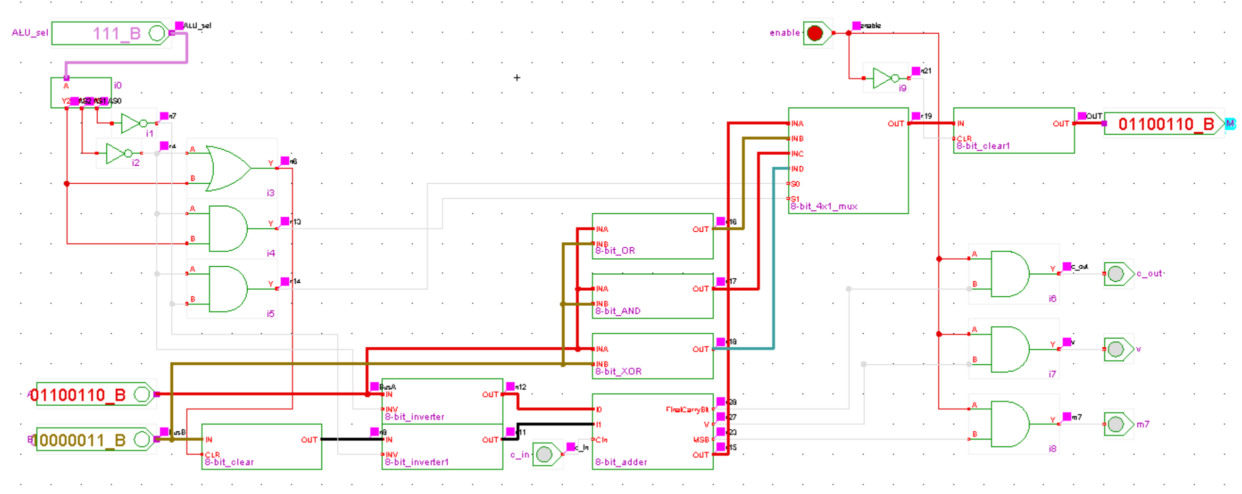
Operations performed based on alu_sel selection inputs are as follows:

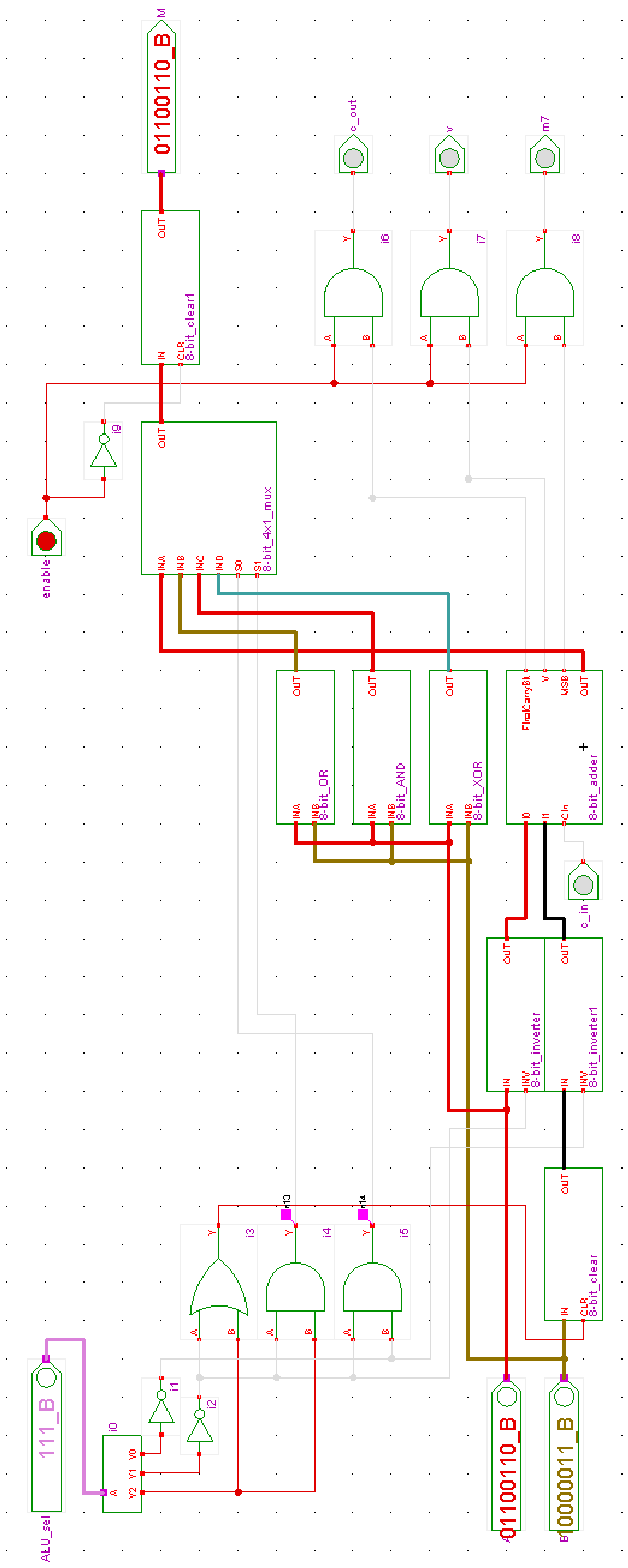
alu_sel	action
0	A OR B
1	not(A)
2	A + not(B) + c_in
3	A + B + c_in
4	A XOR B
5	A AND B
6	A - 1 + c_in
7	A + c_in

Design:

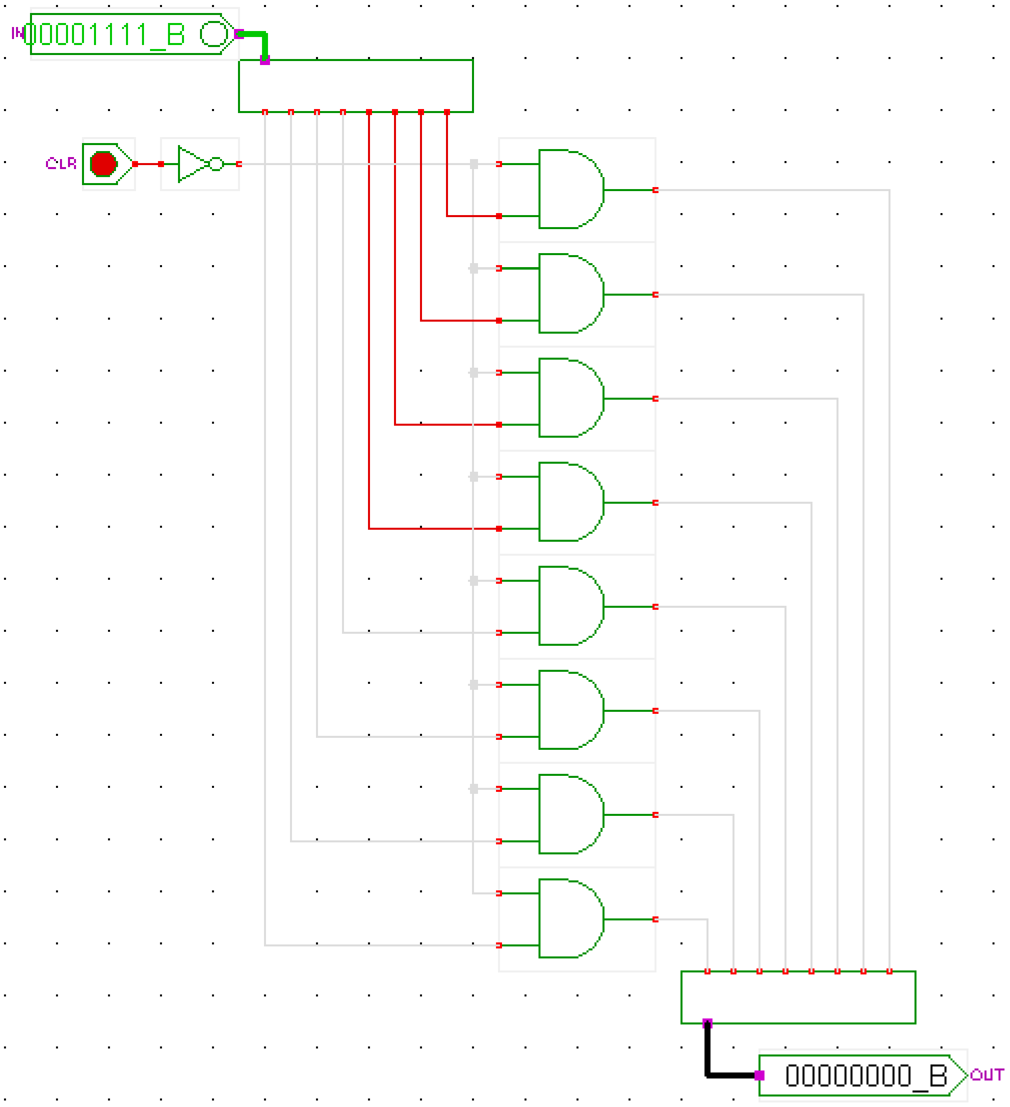
This ALU design is unique, effective, and minimally engineered. The basic design is as follows: two 8-bit input buses (“A” and “B”) and 1-bit c_in get passed straight to the 8-bit logic gates or to an 8-bit inverter and/or “8-bit clear” and then to the 8-bit adder to be added together. The “8-bit clear” is also used as an enable as it sets all of the bits to “0”. This “8-bit clear” is also used when sending “not(A)”, “A – 1 + c_in”, and “A + c_in” through to the adder as it eliminates bus “B” from these equations. The 8-bit inverter is used to flip every bit of the respective buses in the cases of “not(A)” and “A + not(B) + c_in”. All of the respective outputs of these operations are sent to a 8-bit 4x1 mux that is used for selecting which of these results will be sent to the final output bus “M” (labeled “OUT” in the diagram below). This designed coupled with a combinational circuit which take the alu_sel parameters as inputs and determines whether or not the 8-bit inverters and 8-bit clears are turned on and determines which inputs of the 8-bit 4x1 mux to use for the final output. All components are shown below.

ALU DESIGN

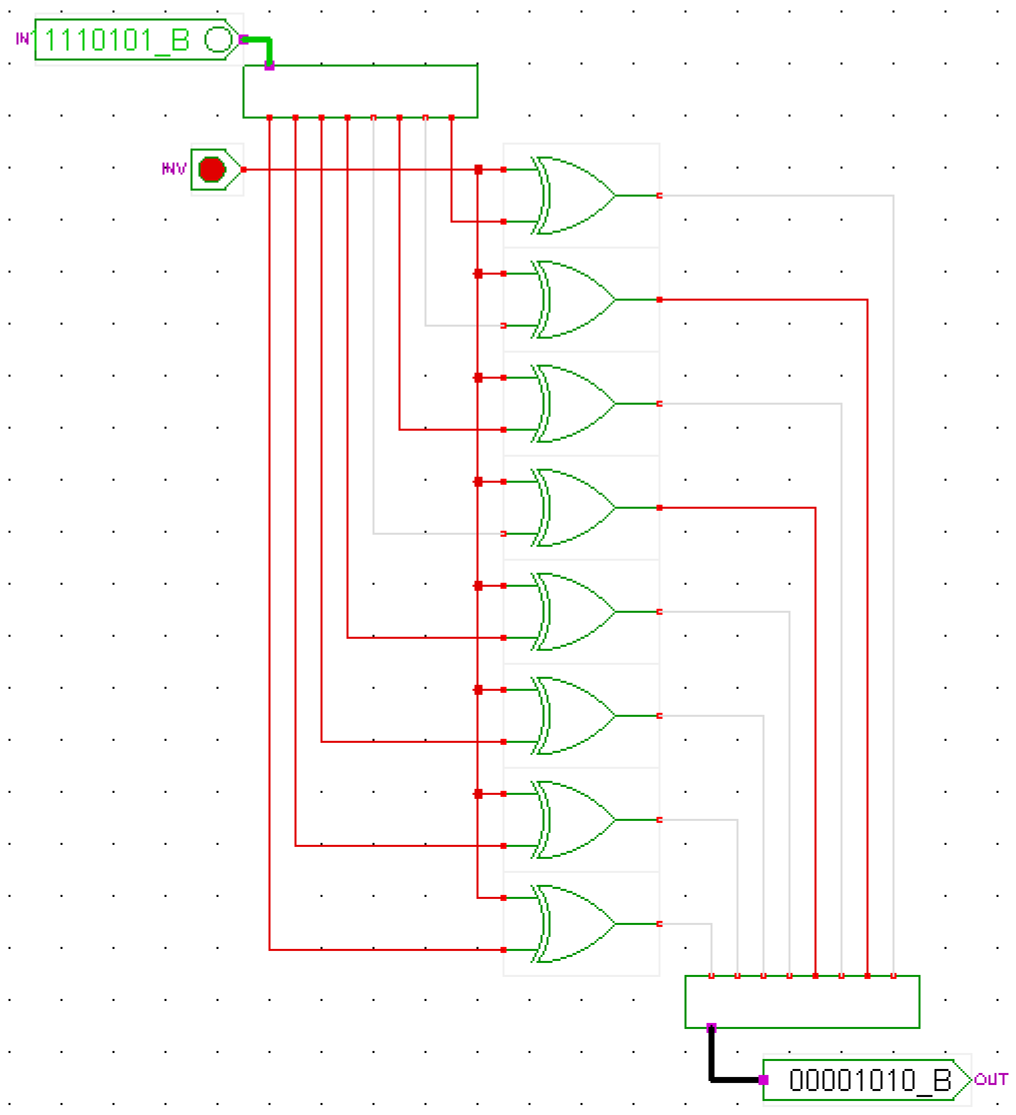



01100110_B
M
c_out
v
m7
i6
i7
i8
8-bit_clear1
8-bit_4x1_mux
enable
i9
8-bit_OR
8-bit_AND
8-bit_XOR
8-bit_adder
FlagCarryBit
MSB
c_in
8-bit_inverter
8-bit_inverter1
8-bit_clear
i3
i4
i5
n13
n14
i0
i1
i2
111_B
ALU_sel
A
01100110_B
B
10000011_B

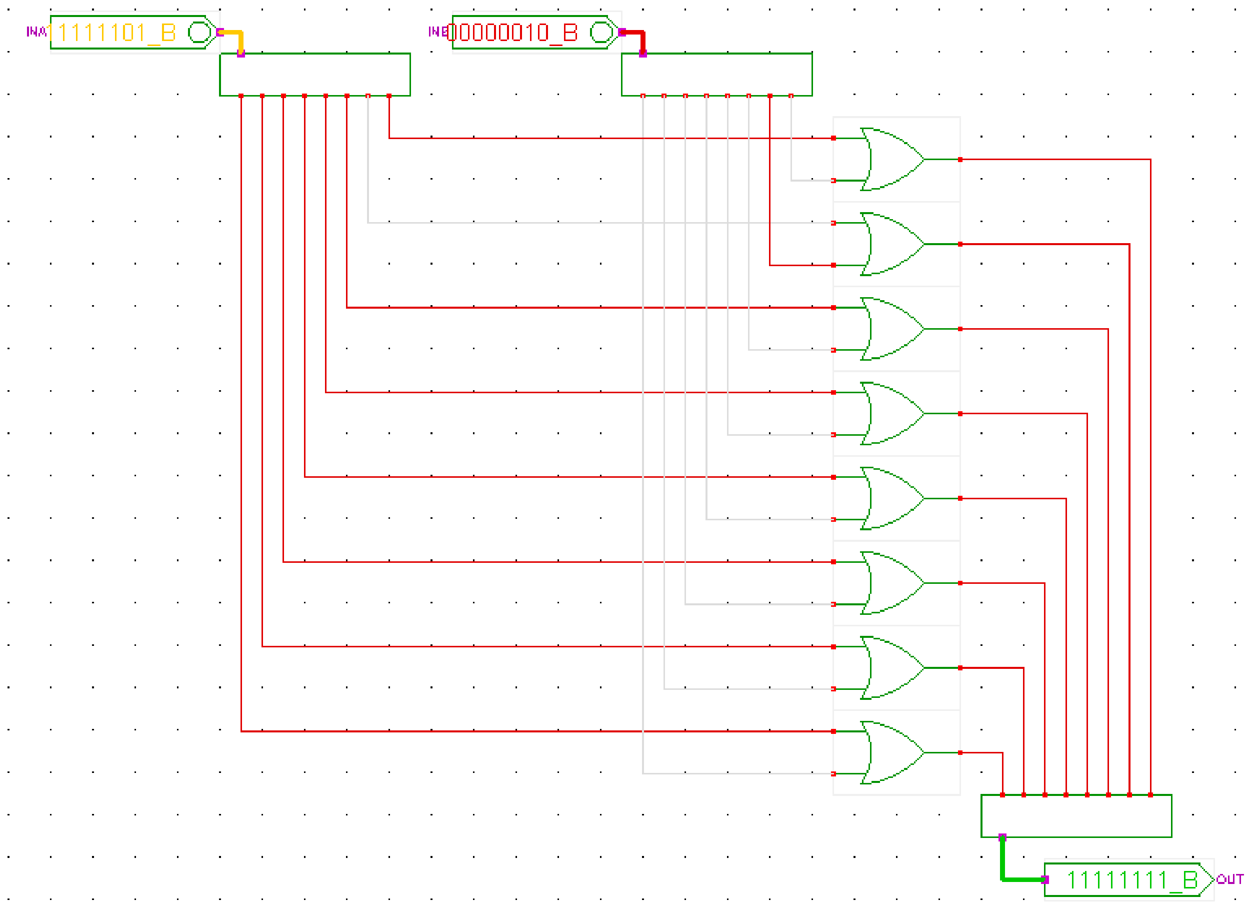
8-BIT CLEAR DESIGN



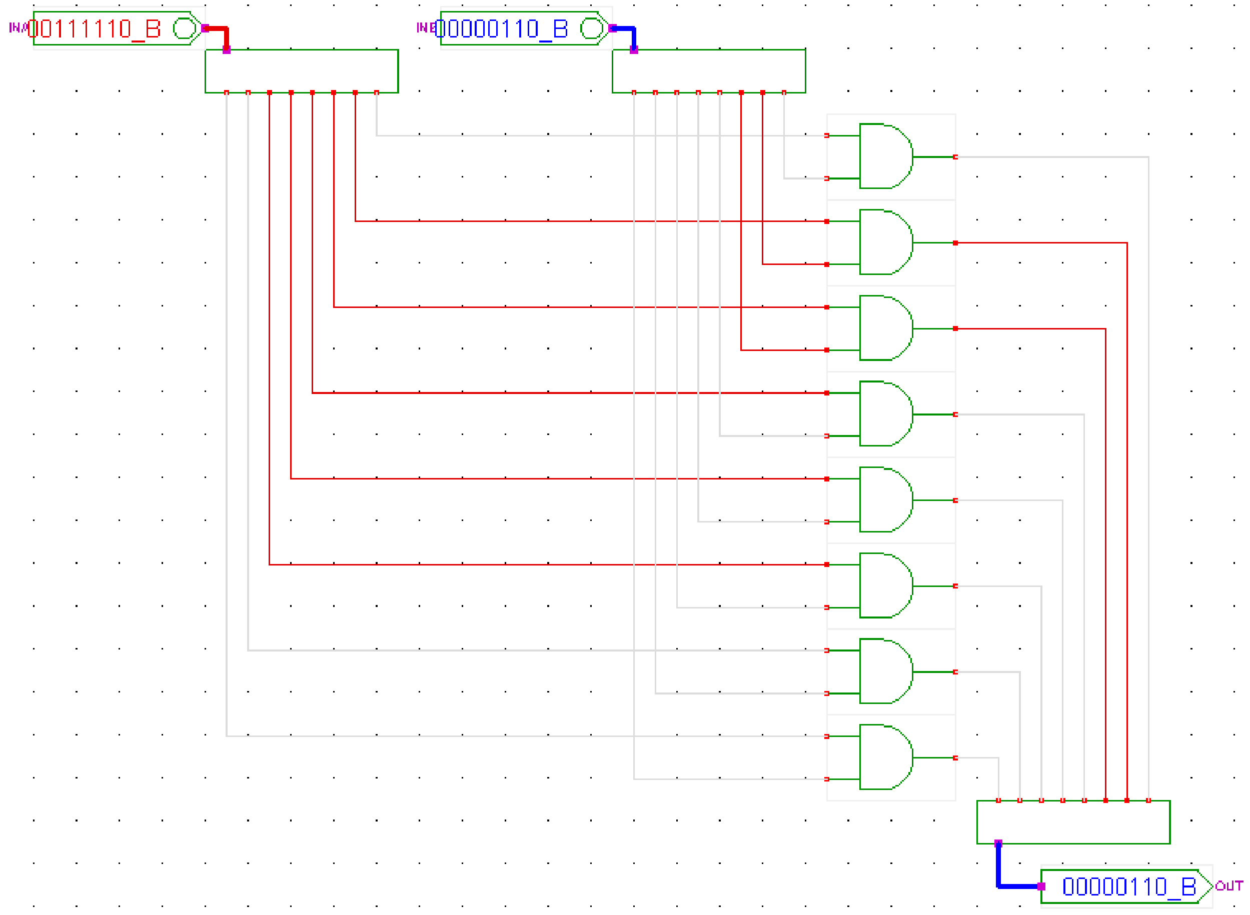
8-BIT INVERTER DESIGN



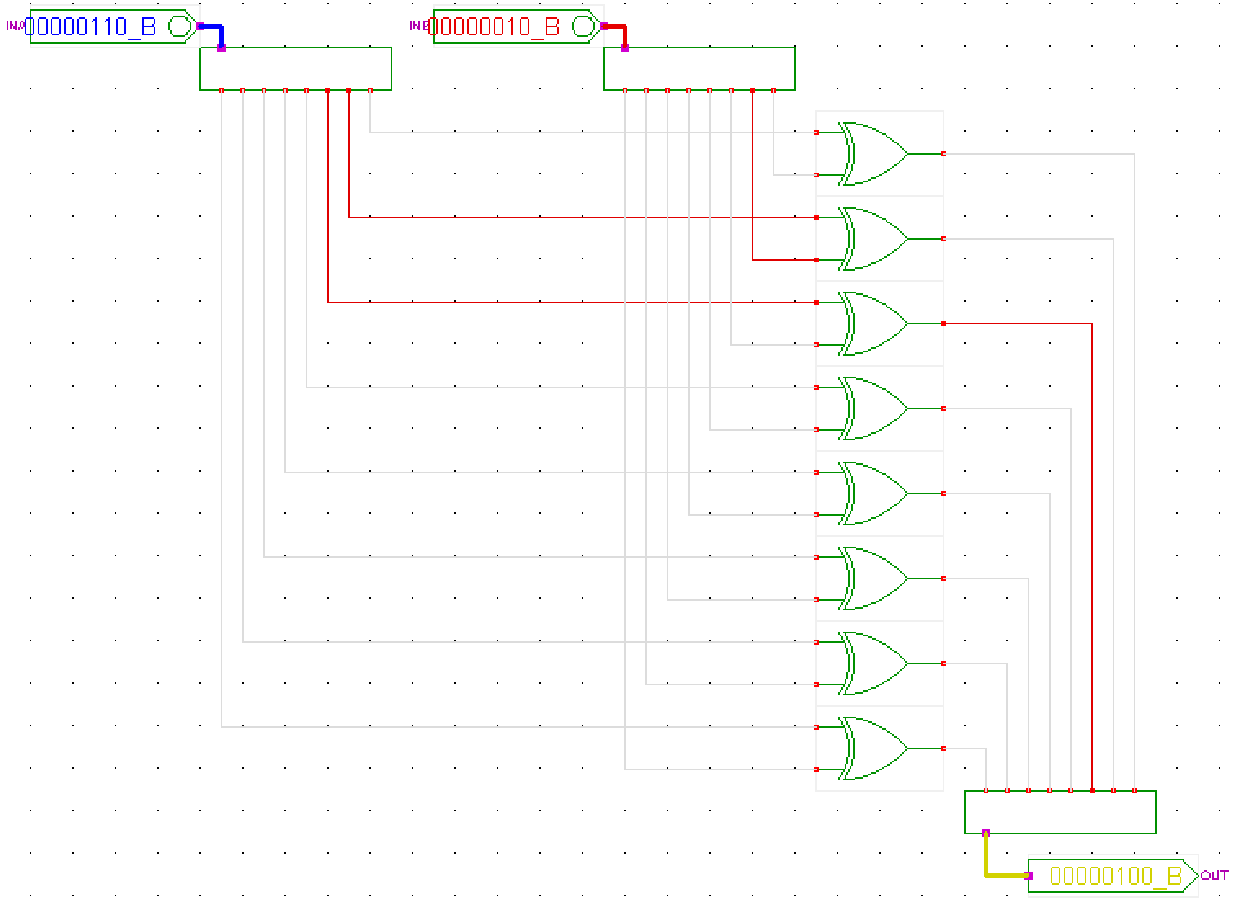
8-BIT OR DESIGN



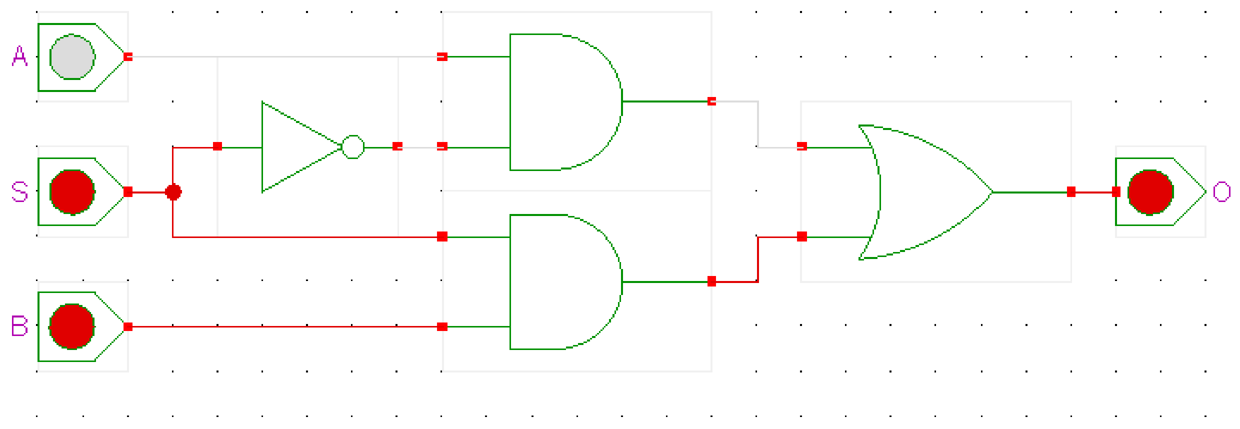
8-BIT AND DESIGN



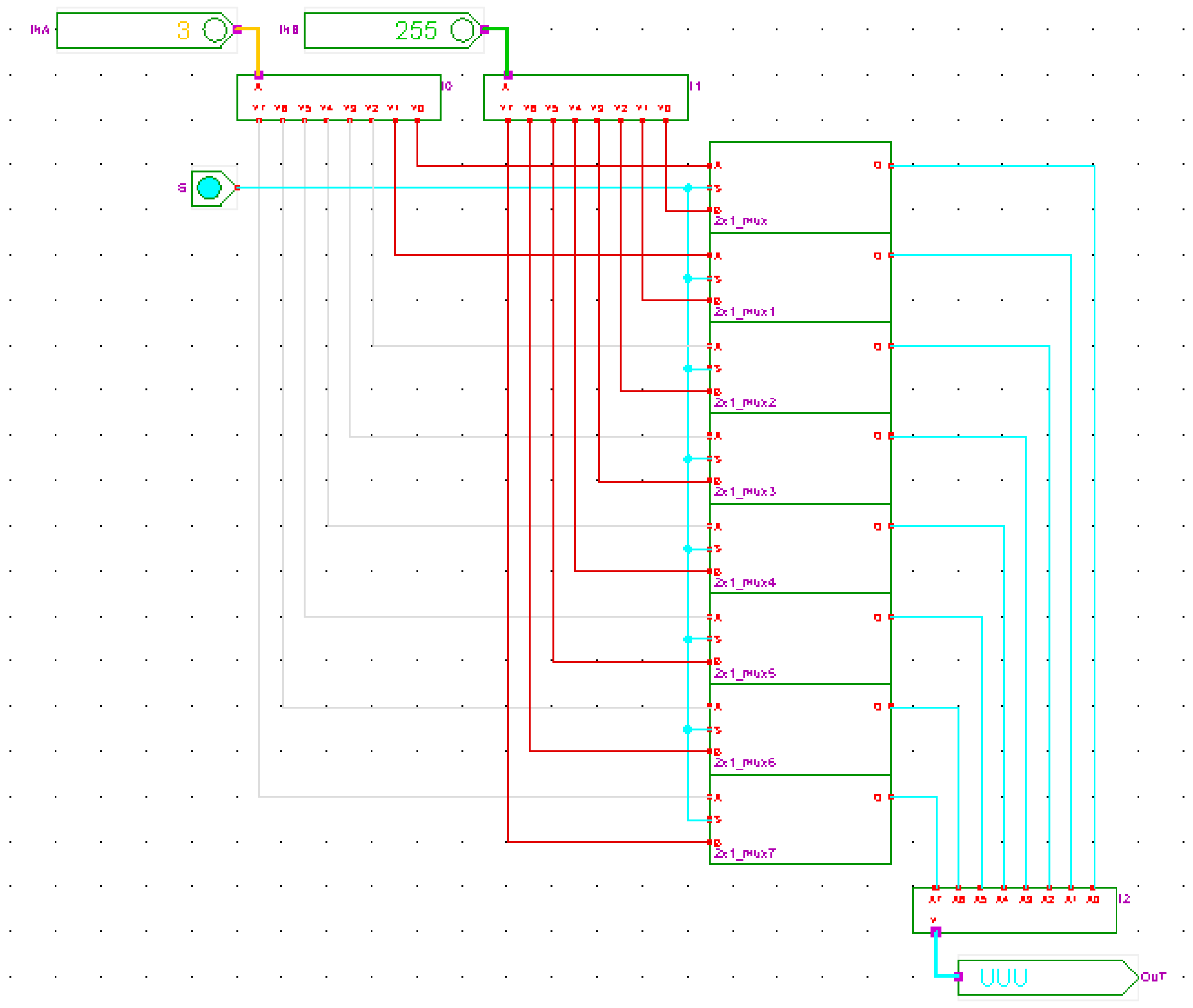
8-BIT XOR DESIGN



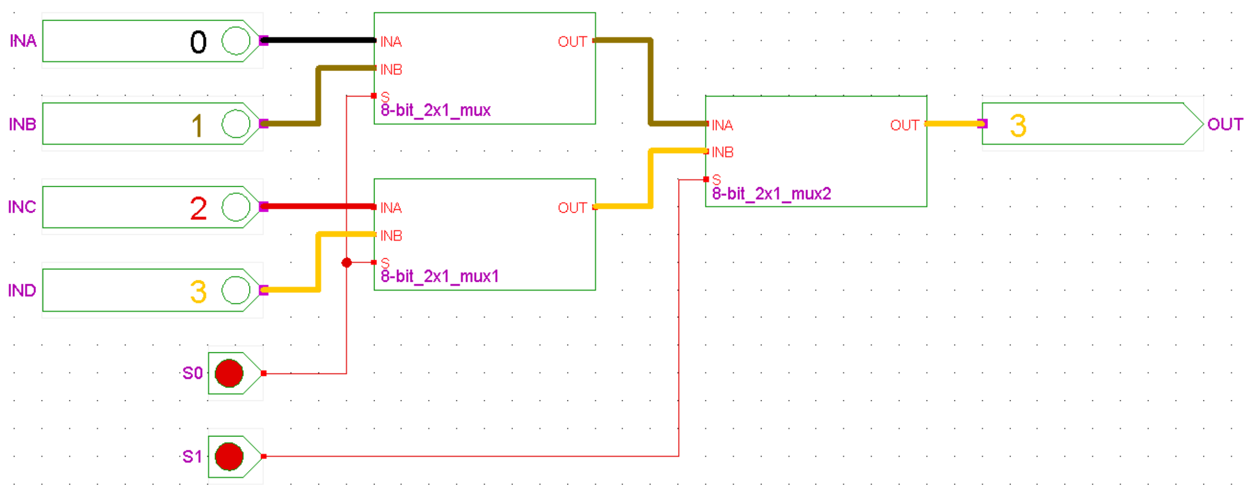
2X1 MUX DESIGN



8-BIT 2X1 MUX DESIGN



8-BIT 4X1 MUX DESIGN



Verification:

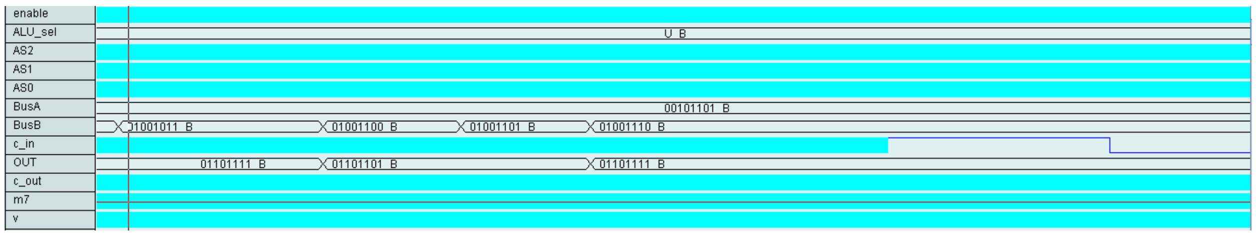
To verify the design of this circuit I tested each of the possible alu_sel settings and manually verified the bits to make sure that they were correct. For the alu_sel settings which performed addition/subtraction I verified cases where overflow occurred and thus demonstrated that my overflow output parameters, namely “c_out” and “v” worked correctly. For each of the 8-bit logic circuits (8-bit AND, OR, XOR, INV, etc) I tested each circuit to make sure that each individual bit was being operated on correctly before I implemented these circuits in my implementation.

Through induction I have verified that each simple gate (AND, OR, XOR, INV, etc) works effectively, and thus each complex circuit (multiplexor, etc) works effectively, and thus this circuit which is composed of such circuits also works effectively.

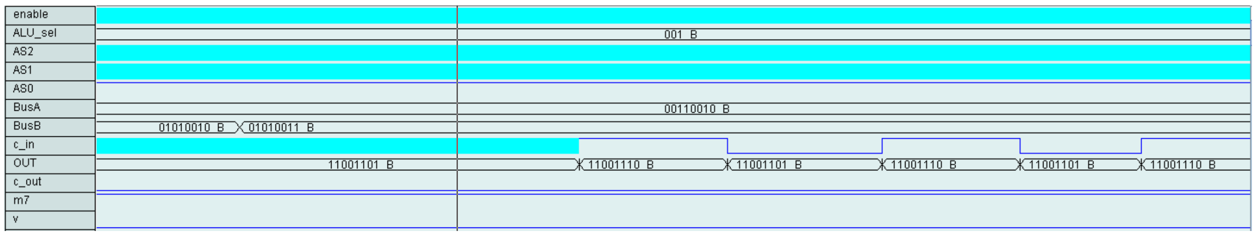
Waveforms:

Although I have tested many more situations than are included or mentioned: I include waveforms for the reader to verify for themselves the functionality of this circuit, in a few cases for each specific alu_sel setting.

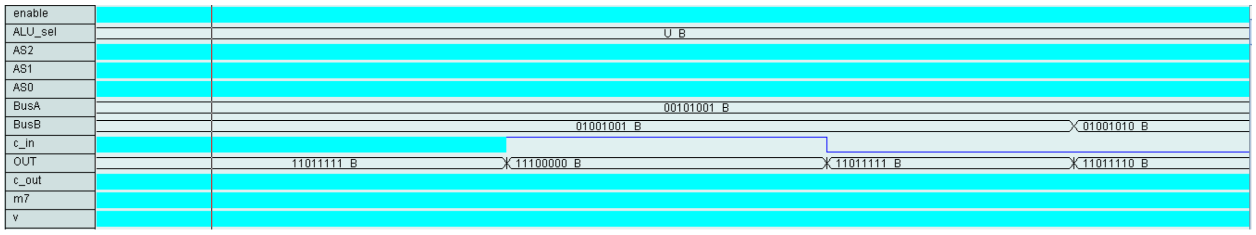
AUL_sel set to 000:



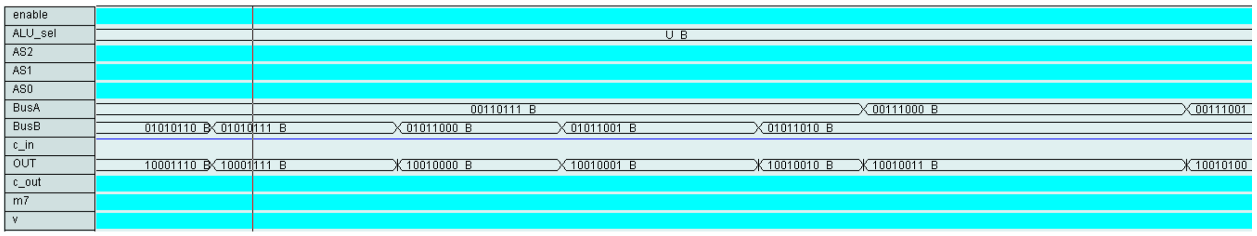
AUL_sel set to 001:



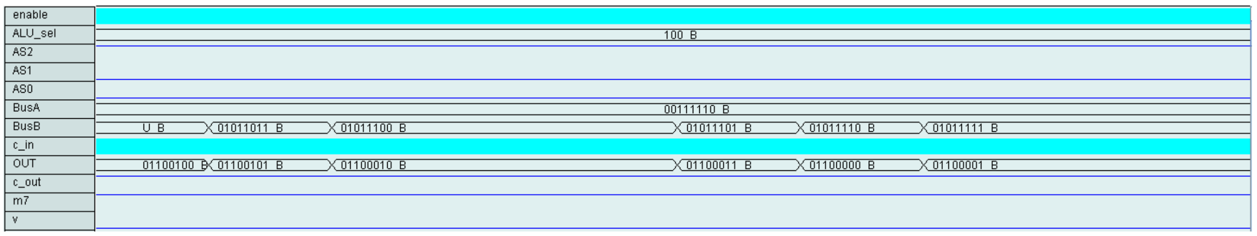
AUL_sel set to 010:



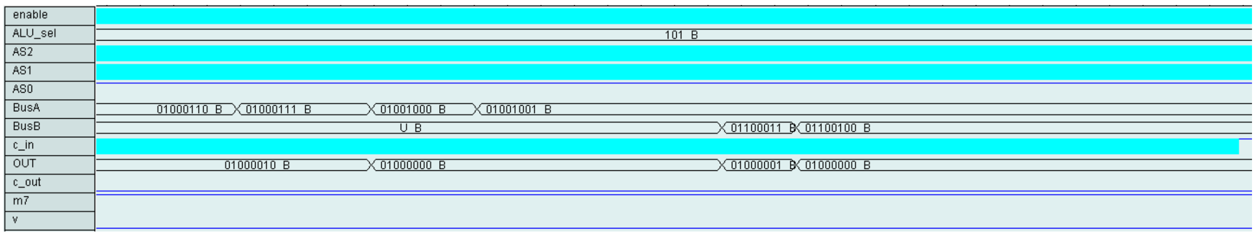
AUL_sel set to 011:



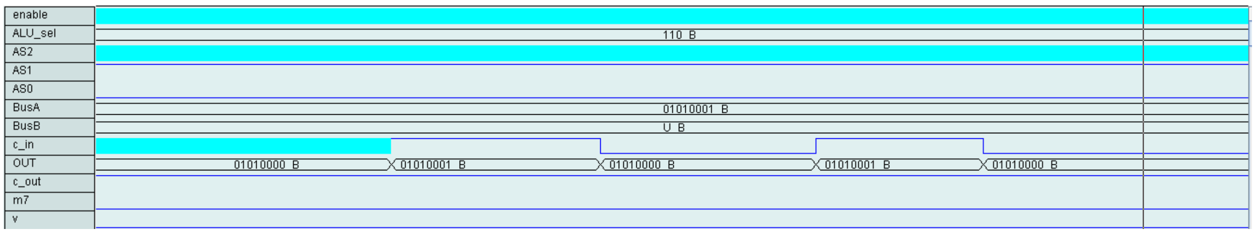
AUL_sel set to 100:



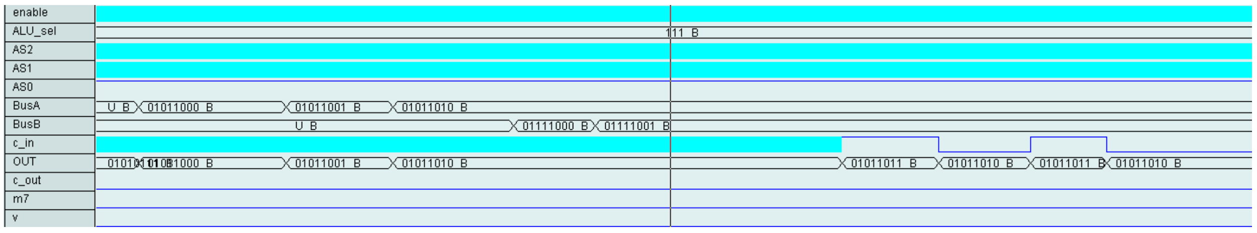
AUL_sel set to 101:



AUL_sel set to 110:



AUL_sel set to 111:


enable
ALU_sel
AS2
AS1
AS0
BusA
BusB
c_in
OUT
c_out
m7
v
111 B
U B
01011000 B
01011001 B
01011010 B
U B
01111000 B
01111001 B
01011000 B
01011001 B
01011010 B
01011011 B
01011010 B
01011011 B
01011010 B