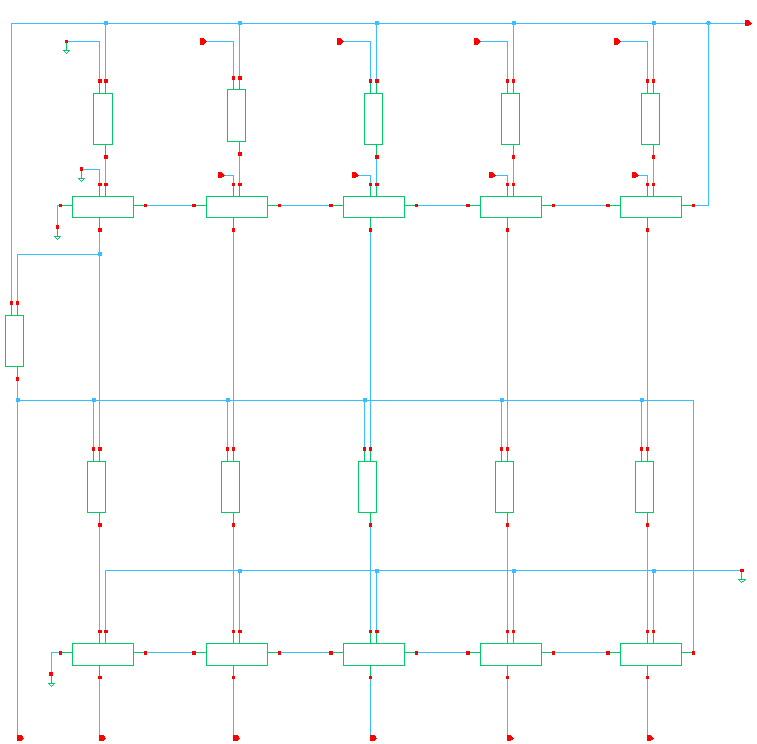
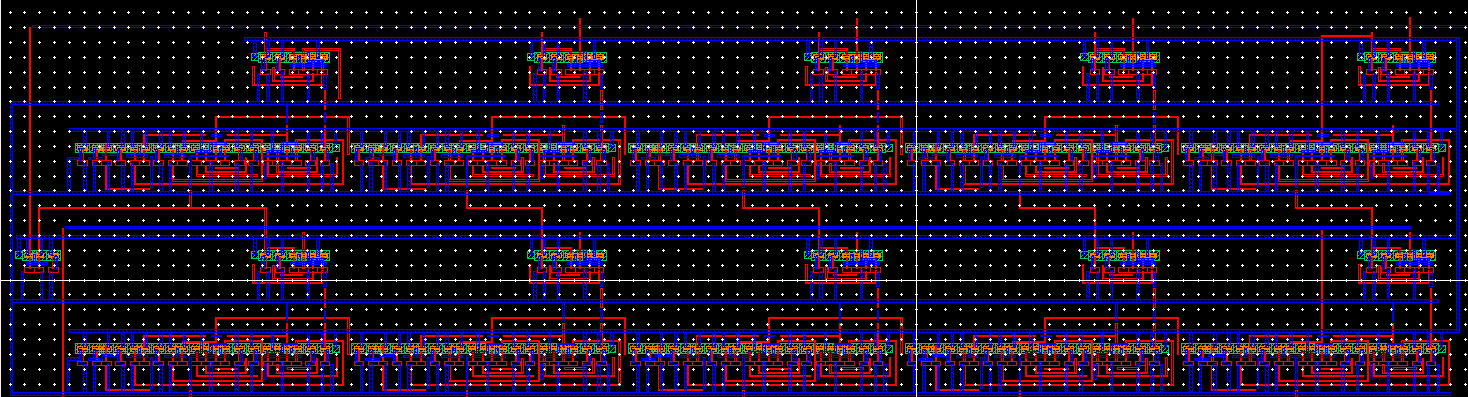
## **EECS 119: Design Project 3**

**Design of a Four Bit Adder/Subtractor**



Layout of the entire circuit:



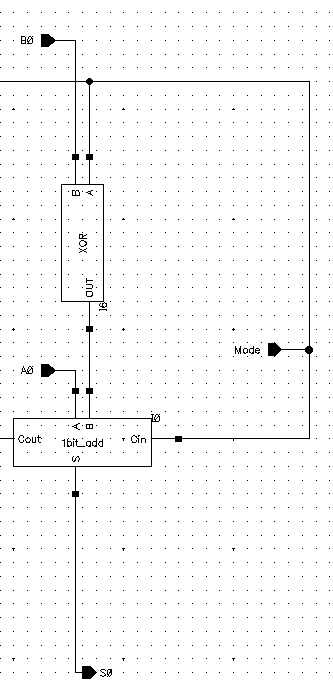
In the above circuit we are trying to make a 4 bit adder and subtractor. We have 9 inputs namely:

A3,A2,A1,A0 which corresponds to the first number

B3,B2,B1,B0 which corresponds to the second number.

And Mode to choose between add and sub (0 for add and 1 for sub)

Since this was an overview, now we are going to expand each component till the gate.



A part of the above image has been magnified. In this circuit we are going to be utilizing an XOR gate and the mode as an input in order to choose between add and subtract option with our mode as one of the inputs.

Mode | B | Mode XOR B

0 | 0 | 0

0 | 1 | 1

1 | 0 | 1

1 | 1 | 0

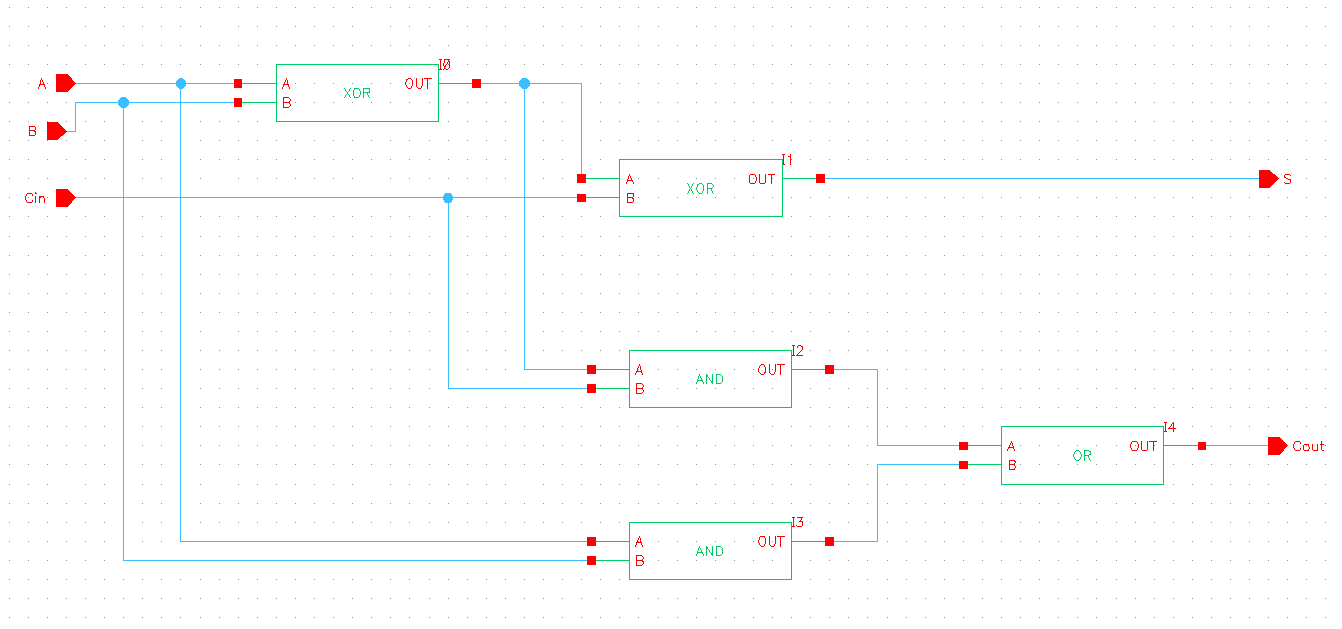
When the Mode is 0 the output is B,  
When the mode is 1 the output is B’

This is being utilized to realize a+b and a-b

If we add the individual parts for add we should not have a problem.

If we subtract the XOR gate takes care of one's complement of B.

The mode is connected to Cin to make it A+(-B) + 1 if its negative to make it a 2s complement.



The picture above corresponds to a 1 bit full adder.

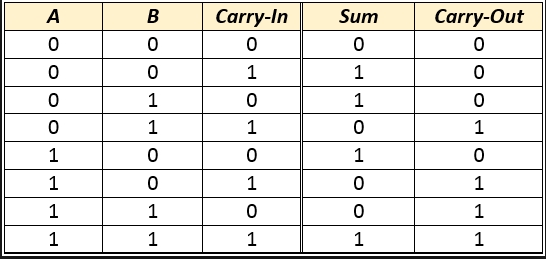
Realization of a full adder.

We are using 2 XOR gates, 2 AND gates and also an OR gate.

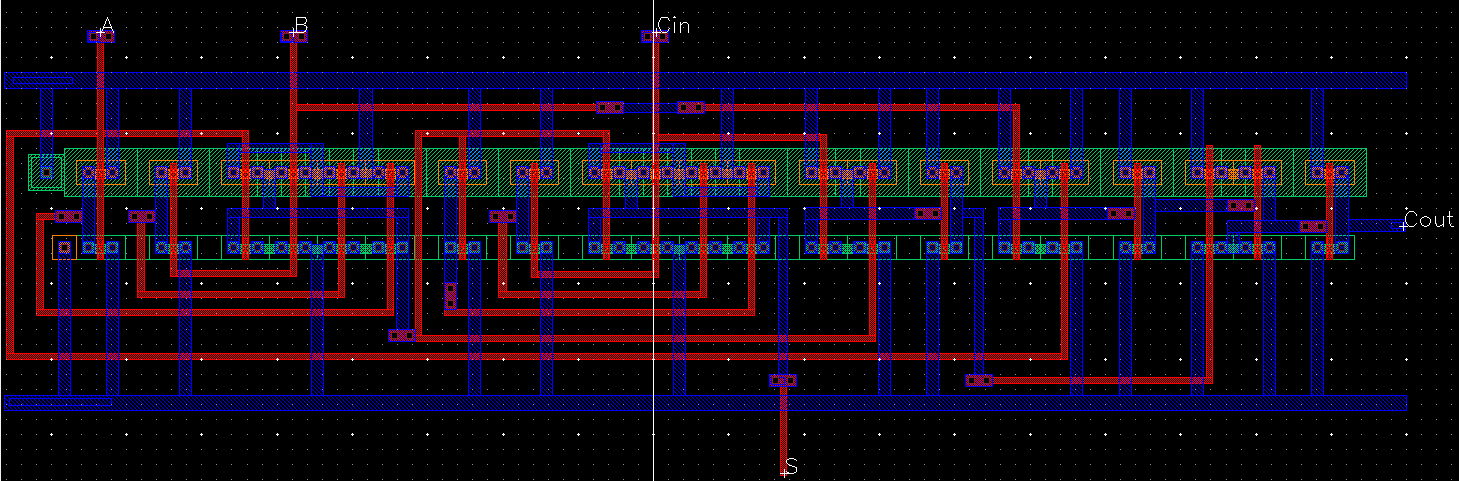
3 inputs : A, B, Cin

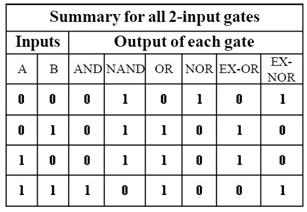
2 Outputs: (S)um and (C)arry(out).

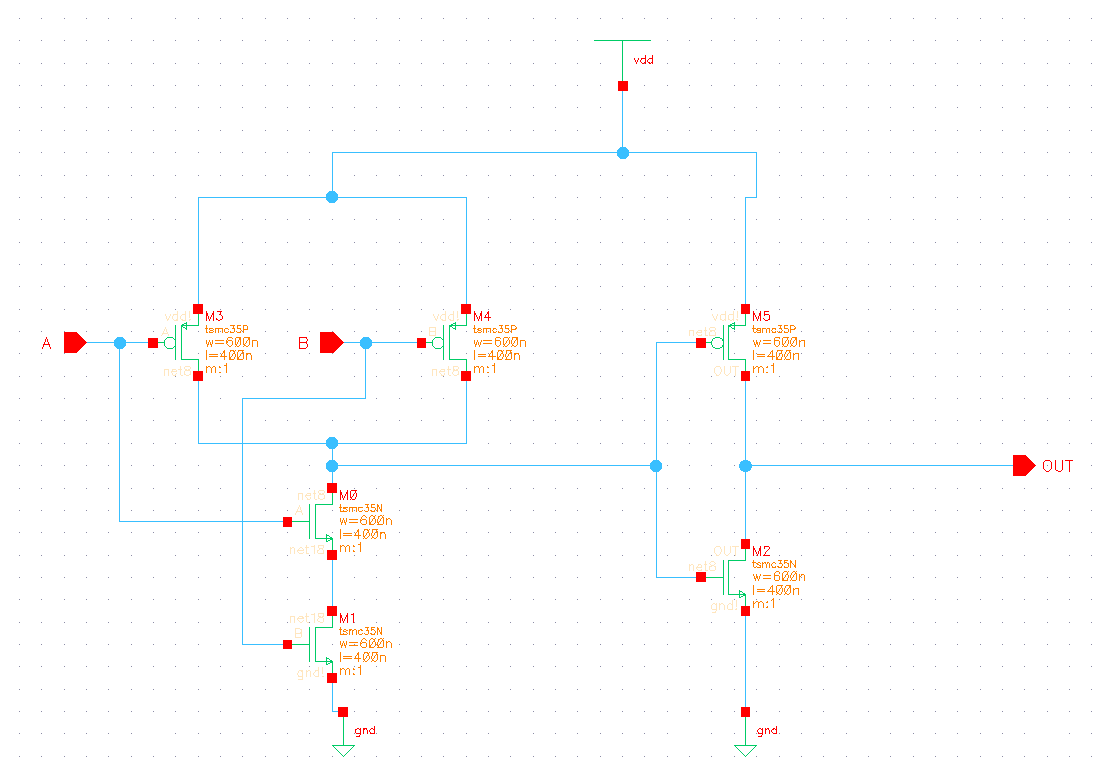
After testing the circuit this is output at the gates.



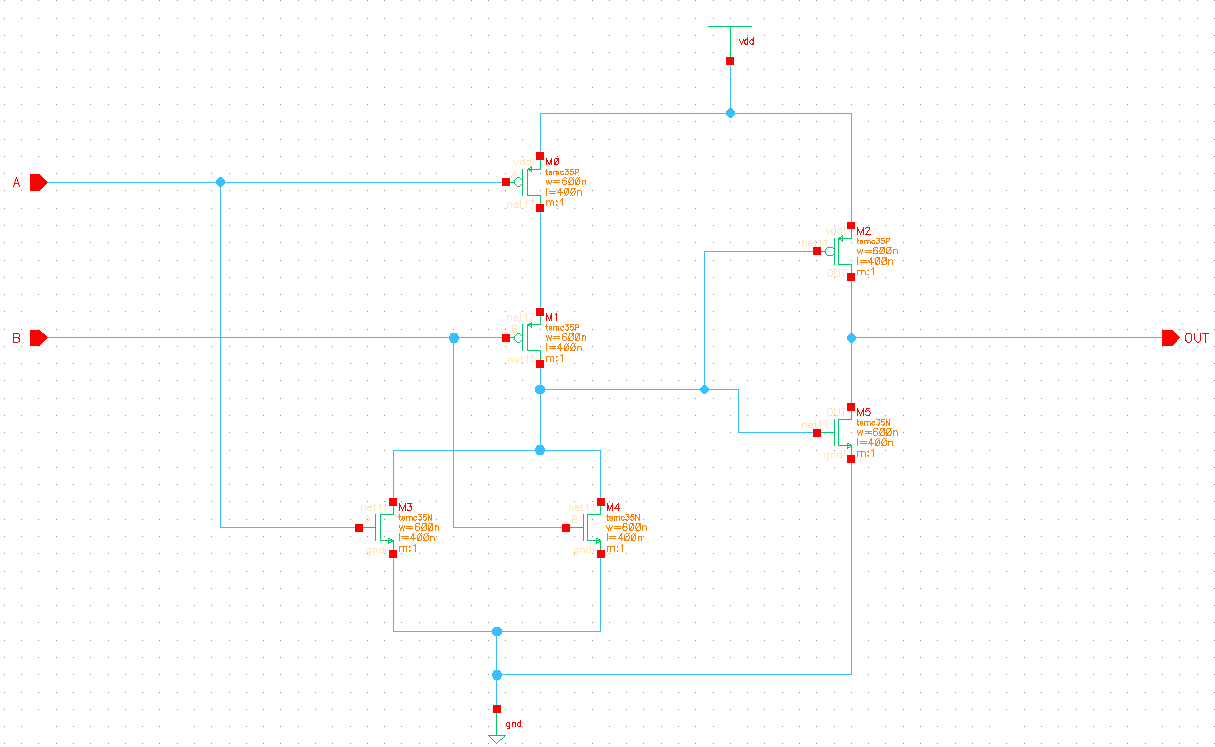
Layout of 1 Bit adder:



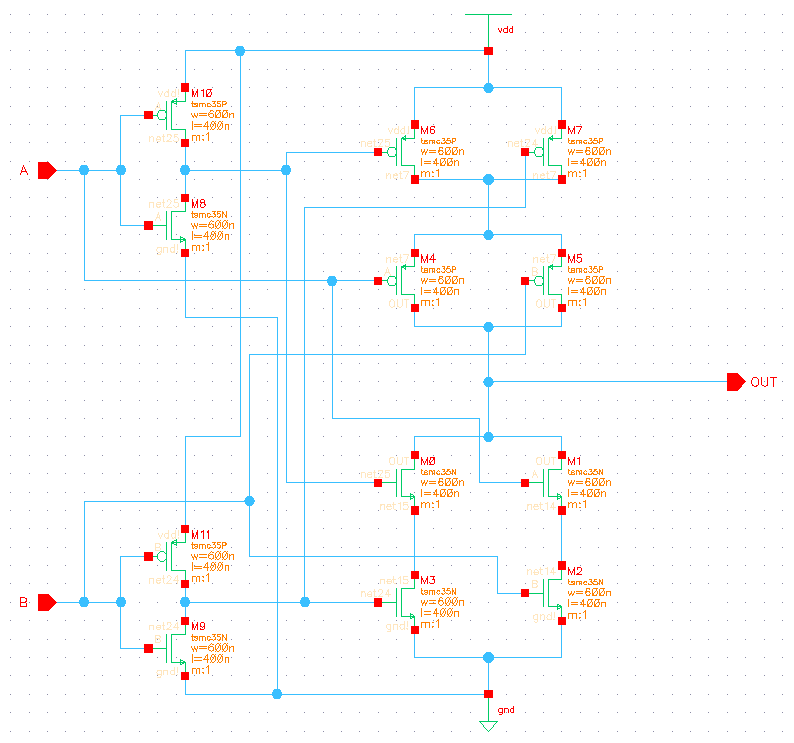




This is an AND game. We basically make a Nand Gate and then use another CMOS to invert the input to make it an AND gate.



We constructed an OR gate by making a NOR gate and then setting the output as the input to a CMOS circuit and making it an OR GATE.



To make a XOR gate was a little bit more complicated. We had to use A and A’ and B and B’. So we first needed a CMOS to invert it.

Similar to how we created the circuit diagram we have made the layout for the diagrams.

***How does it work:***

For addition

We will take the inputs as they are and add them. The mode value shall be 0.

The output will contain 6 bits, namely S0,s1,s2,s3,s4,s5,s6

S6 bit contains the sign value

S5 contains the overflow if any.

For subtraction:

We will take the inputs of A alone as they are. We will pass the inputs of B and the mode, which is 1, into an xor gate and we will get the 1s complement of B. In the following full adder we will add the contents of A and B but will also insert 1 as the Cin of the first full adder making the one's complement a 2s complement. We shall then get the outputs

In the case of the last full adders outputting resulting in 1, we will have to understand the answer is negative and hence need to be inverted to get its original value.   
  
For this we are going to be using another set of 5 full adders to invert the answer and giving us s0,s1,s2,s3,s4,s5.   
S6 will simply be the sign bit which we will get from the And gate.

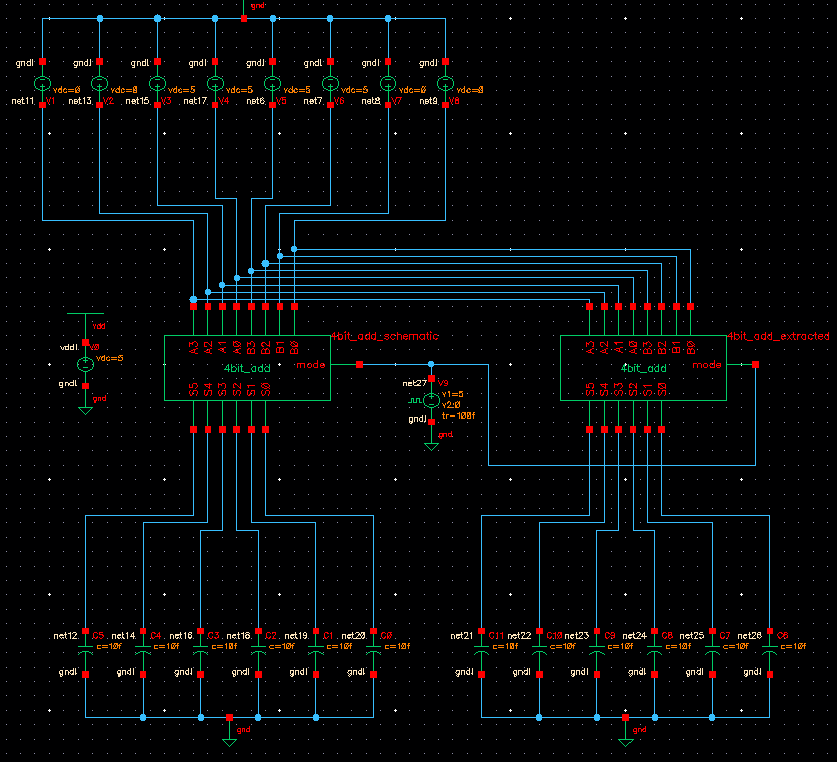
***Results:***

This is the result of simulating a 1 bit adder.

It is in the order of A,B,Cin, Sum, Carryout



Now we shall simulate the cases for the 4 bit adder.  
In the first case I will be simulating 0011 + 1100 (3+12) and 0011-1100 (3-12)



Starting from top

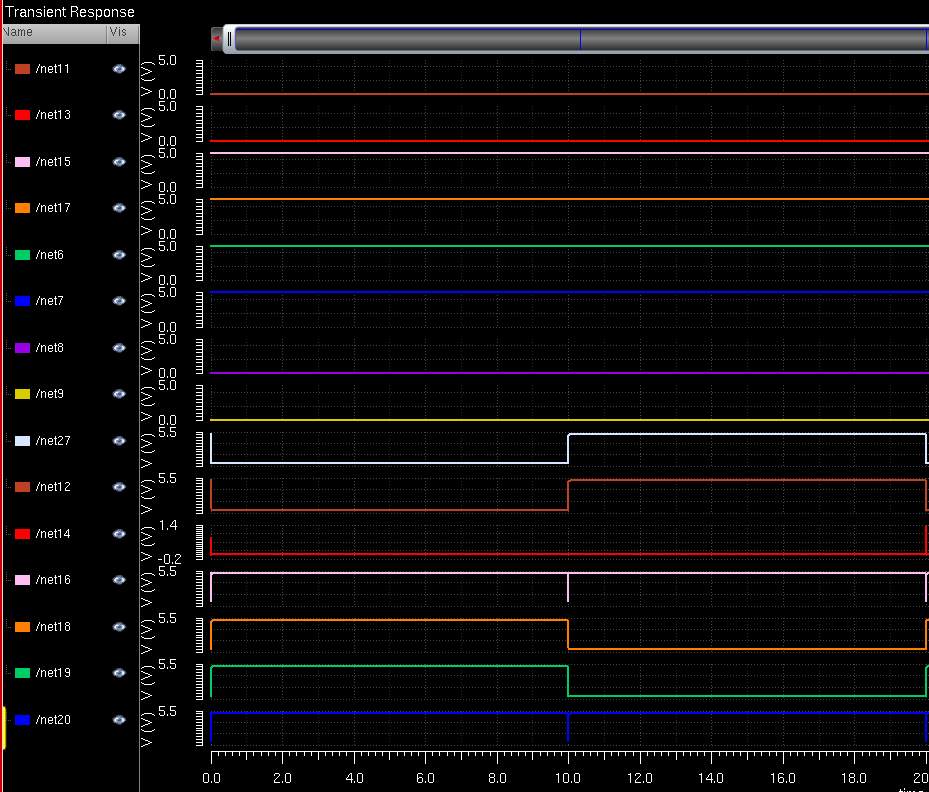
A3,A2,A1,A0 B3,B2,B1,B0, MODE, S5,S4,S3,S2,S1,S0

When Mode = 0

We get 001111 which is right

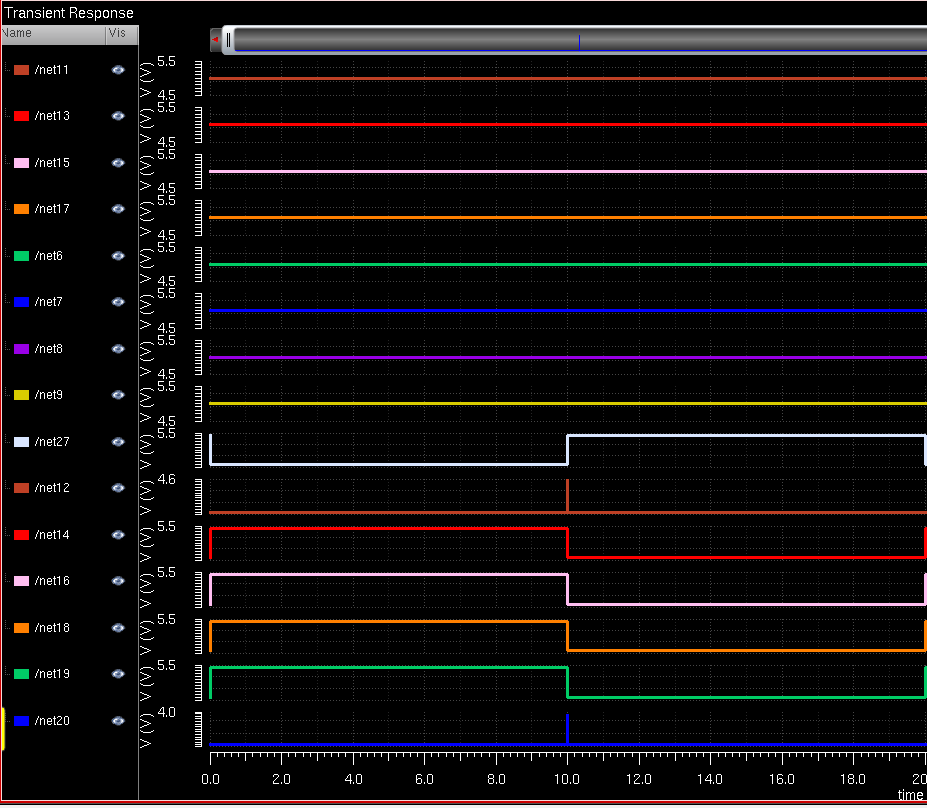
When Mode = 1

We get 101001 which is -9 which is also right



Let us test edge case 15 +15 and 15 -15 to ensure that our circuit works

1111 +1111 and 1111-1111



When mode = 0

We get 011110 = 30 (15+15)

When Mode is 1

We get 000000 (15-15)