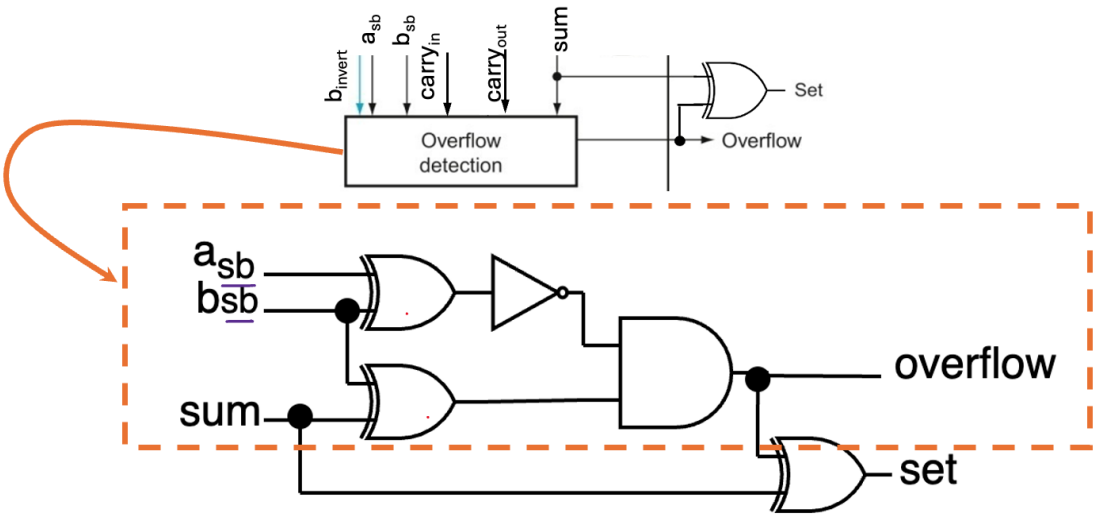
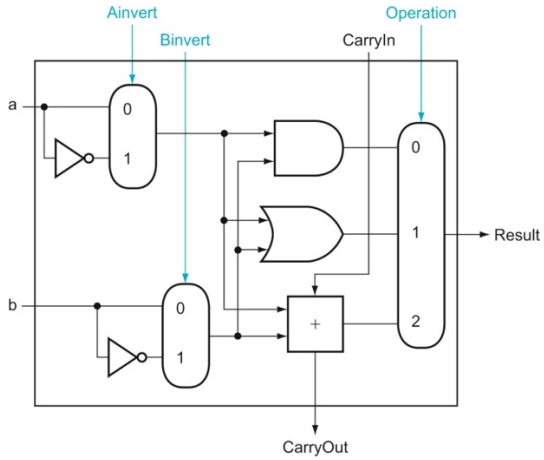
CS281 – Homework #5

1. We have discussed a few reasons for the importance of overflow detection in class. The intuition behind overflow detection is that if the signs of two numbers to be added (or subtracted) are the same, and the resultant sign bit changes then an overflow has occurred. Below is a picture that I used in class slides to show this. There is an alternative way to detect overflow that only requires the $Carry_{in}$ and $Carry_{out}$ bits. Do some research to learn about this alternative way. Draw a circuit to show how this works, and then provide a **very informal** proof (just a bulleted description) showing why using this technique is equivalent to the one that I covered in class.



2. Consider the basic ALU cell we discussed in class shown below. Describe how I could use the below to calculate a ⊕ b without making any modifications?



3. Continuing from the question above, I don't think its possible that XOR can be supported directly on the 32-bit version of the ALU we discussed in class (see picture below on left) with the solution you provided for question 2. Can you identify the changes that would need to be made to the individual ALU cells (shown below on the right) to enable the XOR operation on the 32-bit version of the ALU? Note that you cannot add any new control signals, or extend existing control signals to have more bits, but you can introduce additional logic gates or other components such as multiplexors to formulate your solution.

   Hint: Carefully examine the control signals from our class slides. For example, 0000=AND, 0001=OR, 0010=ADD, 0110=SUB, 0111=SLT, 1100=NOR, 1101=NAND. I would strongly suggest assigning control signal 1110 to XOR. Note that there are likely other ways to do this, but one solution that I see involves adding an and gate, a not gate, and a single multiplexor to the ALU Cell.