# Overflow Detection

In Chapters 3 and 9 we covered addition of 2's complement (signed) numbers and discussed 2 of the 3 concepts associated with it.

## Overflow CAN OCCUR

The first concept is that overflow **can** occur. Consider 4-bit numbers. These can hold values between -8 and +7. Obviously, adding 4 + 6 is too large to fit in a 4-bit quantity. The same goes for -8 + -1.

## We Can PREVENT Overflow

We also learned that we can prevent overflow by first sign extending the two operands. In the case of 4-bit numbers, if we sign extend them to 5-bit numbers then we are guaranteed that the 5-bit result will always be correct.

## We Can DETECT Overflow

There are times when we don't want to (or cannot) sign-extend. Think of a simple CPU that deals with 16-bit operands - everything in the system is 16 bits - the memory, the register file, the communication buses. In such a case we make the decision to not prevent overflow (by sign-extending everything to 17 bits). Rather, we decide we will do 16-bit addition but DETECT if overflow occurred and signal that to the rest of the system so the system can take some kind of action.

The point of HW 9 is to lead you through a guided activity so you can *discover* or *derive* a circuit to detect overflow. In fact, you are going to design *two* different circuits since there are two different ways to detect overflow.

As the figure in HW 9 shows, by looking at the inputs and outputs of the leftmost stage of your adder circuit you can detect when overflow happens. In one case (Overflow Detector 1) you look at the a, b, and s signals and you can tell. In the other case (Overflow Detector 2) you can tell simply by looking at the carry-in and the carry-out associated with the leftmost stage of your adder circuit.

So, how to proceed? You can been provided two truth tables to fill out. Perhaps the best way to fill them out is the following:

1. Choose two positive numbers that, when added don't overflow, do the addition, and fill in the row(s) of the truth tables they represent (in this case the rows will have an output value of 0 since overflow didn't occur).
2. Repeat the above but for two positive numbers that do overflow and fill in the row(s) of the truth tables they represent (in this case the rows will have an output value of 1 since overflow did occur).
3. Repeat step 1 but for two negative numbers that don't overflow.
4. Repeat step 2 but for two negative numbers that do overflow.

By now your tables will be nearing completion (actually one will be complete, the other will still have some empty rows).

Now choose additional pairs of values to add that will fill in the remaining rows of the truth table. In each case, record if overflow did occur by putting a 1 in the output column for that case. Eventually, you will have two complete tables.

Now you can write a logic equation for Overflow Detector 1 from its truth table and a logic equation for Overflow Detector 2 from its truth table.

The logic equation for Overflow Detector 1 should make sense if you think about it. HINT: if you add two positive numbers and the result looks like a negative number what can you conclude happened? A similar thing happens when you add two negative numbers and the result looks positive.

The logic equation for Overflow Detector 2 is an even simpler equation. Even though it is harder to grasp an intuitive understanding of it like with Overflow Detector 1, it is still correct.

## Conclusion

So, if you ever need to *detect* rather than *prevent* overflow, you now have two different circuits you can use to do so.