UW Math Circle
October 10th, 2019

# Digital Circuits - Homework 2

## Problem 1: Build some circuits

At home, go to `nandgame.com` and practice building circuits starting from $\boxed{\text{NAND}}$. Note that they use the name INV for the negation gate we have called $\boxed{\text{NEG}}$.

## Problem 2: 2s complement

In 8-bit 2s-complement, what is the bit representation of -21? Writing out the bit representation of each number and doing manual grade-school binary addition, show that 67 + (-21) is 46.

## Problem 3: Subtraction

There are many ways to implement circuits that subtract binary numbers. In class we discussed how to subtract by *adding* the 2s-complement.

The more direct way to subtract is right-to-left gradeschool subtraction, where for each digit the subtrahend $S$ (the bottom number) and any *borrow* coming in from the right ($B_{\text{in}}$) are subtracted from the minuend $M$ (the top number) to produce a difference digit $D$, and if the minuend isn't sufficient, a borrow is propagated out to the next digit on the left ($B_{\text{out}}$). This is very similar to the addition algorithm: we need a half-subtractor for the rightmost digit that takes $M$ and $S$ and produces $D$ and $B_{\text{out}}$, and then for remaining digits full-subtractors that take $M$ and $S$ and $B_{\text{in}}$ and produce $D$ and $B_{\text{out}}$. Thus, the full-subtractor is a 3-input 2-output circuit.

| Place | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
|---|---|---|---|---|---|---|---|
| Minuend (M) | | 1 | 0 | 0 | 0 | 1 | (17) |
| Subtrahend (S) | | 0 | 1 | 0 | 1 | 1 | (11) |
| Borrow (B) | 0 | 1 | 1 | 1 | 0 | | |
| Difference (D) | | 0 | 1 | 1 | 1 | 0 | (6) |

In this example, in the $2^2$ place, the subtrahend $S = 0$ and the incoming borrow $B = 1$ are subtracted from the minuend $M = 0$. Since $0 + 1 > 0$, there must be a borrow from the next place (borrowing $2^3$, or 2 units of $2^2$). Therefore $B_{\text{out}} = 1$ and $D = 2 - (0 + 1) = 1$.

(a) Using the same table structure as the 17 - 11 = 6 example, manually subtract the 5-bit representation of 26 from the 5-bit representation of 5. Is the result what one expects for the 2s-complement representation of -21?

(b) Write the logic table for the 2-input 2-output $\boxed{\text{HALF-SUB}}$ circuit used for the rightmost digit (where there is no incoming borrow), and implement it using gates you know.

(c) Write the logic table for the 3-input 2-output full $\boxed{\text{SUB}}$ circuit, and implement it using existing gates - you may use $\boxed{\text{HALF-SUB}}$.

(d) Using $\boxed{\text{SUB}}$ and $\boxed{\text{HALF-SUB}}$, construct a circuit that subtracts two 4-bit numbers to produce a 4-bit number and a borrow bit.

## Problem 4: Three-to-two compression

A $\boxed{\text{FULL-ADD}}$ adds 3 bits (the two addends and the incoming carry $C_{\text{in}}$), so the minimum output value is 0 and the maximum is 3. Therefore each possible output value can be represented in binary using two digits, hence the two digit output $C_{\text{out}}$ holding the $2^1$ place and $S$ holding the $2^0$ place. Normally to add two binary numbers we make a chain of $\boxed{\text{FULL-ADD}}$ circuits with the carry propagating from one to the next. But this means that when adding many digit numbers, the latency grows.

Suppose we want to add *three* binary numbers of many digits without the latency growing. That is, we want the latency to be only that of a single $\boxed{\text{FULL-ADD}}$, or less. Show how to do this by producing not one but *two* output numbers, that when added equal the sum of the inputs. That is, show how to construct a circuit that takes in *three* $n$-bit numbers (call them $A$, $B$ and $C$) and outputs *two* $(n+1)$-bit numbers (call them $X$ and $Y$), such that $A+B+C = X+Y$, where the latency of your circuit is no more than that of a single $\boxed{\text{HALF-ADD}}$.

What are your $X$ and $Y$ if the input numbers are $A = 10, B = 7, C = 9$? (Convert them to binary, execute your circuit, convert the results back to decimal)?

This "three to two compression" circuit is a key to fast *multiplication*, as we shall see.