ECE120: Introduction to Computer Engineering

Notes Set 1.3 Overflow Conditions

This set of notes discusses the overflow conditions for unsigned and 2's complement addition. For both types, we formally prove that the conditions that we state are correct. Many of our faculty want our students to learn to construct formal proofs, so we plan to begin exposing you to this process in our classes. Prof. Lumetta is a fan of Prof. George Polya's educational theories with regard to proof techniques, and in particular the idea that one builds up a repertoire of approaches by seeing the approaches used in practice.

1.3.1 Implication and Mathematical Notation

Some of you may not have been exposed to basics of mathematical logic, so let's start with a brief introduction to implication. We'll use variables p and q to represent statements that can be either true or false. For example, p might represent the statement, "Jan is an ECE student," while q might represent the statement, "Jan works hard." The **logical complement** or **negation** of a statement p, written for example as "not p," has the opposite truth value: if p is true, not p is false, and if p is false, not p is true.

An **implication** is a logical relationship between two statements. The implication itself is also a logical statement, and may be true or false. In English, for example, we might say, "If p, q." In mathematics, the same implication is usually written as either "q if p" or "$p \rightarrow q$," and the latter is read as, "p implies q." Using our example values for p and q, we can see that $p \rightarrow q$ is true: "Jan is an ECE student" does in fact imply that "Jan works hard!"

The implication $p \rightarrow q$ is only considered false if p is true and q is false. In all other cases, the implication is true. This definition can be a little confusing at first, so let's use another example to see why. Let p represent the statement "Entity X is a flying pig," and let q represent the statement, "Entity X obeys air traffic control regulations." Here the implication $p \rightarrow q$ is again true: flying pigs do not exist, so p is false, and thus "$p \rightarrow q$" is true—for any value of statement q!

Given an implication "$p \rightarrow q$," we say that the **converse** of the implication is the statement "$q \rightarrow p$," which is also an implication. In mathematics, the converse of $p \rightarrow q$ is sometimes written as "q only if p." The converse of an implication may or may not have the same truth value as the implication itself. Finally, we frequently use the shorthand notation, "p if and only if q," (or, even shorter, "p iff q") to mean "$p \rightarrow q$ *and* $q \rightarrow p$." This last statement is true only when both implications are true.

1.3.2 Overflow for Unsigned Addition

Let's say that we add two N-bit unsigned numbers, A and B. The N-bit unsigned representation can represent integers in the range $[0, 2^N - 1]$. Recall that we say that the addition operation has overflowed if the number represented by the N-bit pattern produced for the sum does not actually represent the number $A + B$.

For clarity, let's name the bits of A by writing the number as $a_{N-1}a_{N-2}...a_1a_0$. Similarly, let's write B as $b_{N-1}b_{N-2}...b_1b_0$. Name the sum $C = A + B$. The sum that comes out of the add unit has only N bits, but recall that we claimed in class that the overflow condition for unsigned addition is given by the **carry** out of the most significant bit. So let's write the sum as $c_Nc_{N-1}c_{N-2}...c_1c_0$, realizing that c_N is the carry out and not actually part of the sum produced by the add unit.

Theorem: Addition of two N-bit unsigned numbers $A = a_{N-1}a_{N-2}...a_1a_0$ and $B = b_{N-1}b_{N-2}...b_1b_0$ to produce sum $C = A + B = c_Nc_{N-1}c_{N-2}...c_1c_0$, overflows if and only if the carry out c_N of the addition is a 1 bit.

Proof: Let's start with the "if" direction. In other words, $c_N = 1$ implies overflow. Recall that unsigned addition is the same as base 2 addition, except that we discard bits beyond c_{N-1} from the sum C. The bit c_N has place value 2^N, so, when $c_N = 1$ we can write that the correct sum $C \geq 2^N$. But no value that large can be represented using the N-bit unsigned representation, so we have an overflow.

The other direction ("only if") is slightly more complex: we need to show that overflow implies that $c_N = 1$. We use a range-based argument for this purpose. Overflow means that the sum C is outside the representable range $[0, 2^N - 1]$. Adding two non-negative numbers cannot produce a negative number, so the sum can't be smaller than 0. Overflow thus implies that $C \geq 2^N$.

Does that argument complete the proof? No, because some numbers, such as 2^{N+1}, are larger than 2^N, but do not have a 1 bit in the Nth position when written in binary. We need to make use of the constraints on A and B implied by the possible range of the representation.

In particular, given that A and B are represented as N-bit unsigned values, we can write

$$\begin{aligned} 0 \leq \quad A \quad &\leq 2^N - 1 \\ 0 \leq \quad B \quad &\leq 2^N - 1 \end{aligned}$$

We add these two inequalities and replace $A + B$ with C to obtain

$$0 \leq \quad C \quad \leq 2^{N+1} - 2$$

Combining the new inequality with the one implied by the overflow condition, we obtain

$$2^N \leq \quad C \quad \leq 2^{N+1} - 2$$

All of the numbers in the range allowed by this inequality have $c_N = 1$, completing our proof.

1.3.3 Overflow for 2's Complement Addition

Understanding overflow for 2's complement addition is somewhat trickier, which is why the problem is a good one for you to think about on your own first. Our operands, A and B, are now two N-bit 2's complement numbers. The N-bit 2's complement representation can represent integers in the range $[-2^{N-1}, 2^{N-1} - 1]$. Let's start by ruling out a case that we can show never leads to overflow.

Lemma: Addition of two N-bit 2's complement numbers A and B does not overflow if one of the numbers is negative and the other is not.

Proof: We again make use of the constraints implied by the fact that A and B are represented as N-bit 2's complement values. We can assume **without loss of generality**[1], or **w.l.o.g.**, that $A < 0$ and $B \geq 0$.

Combining these constraints with the range representable by N-bit 2's complement, we obtain

$$\begin{aligned} -2^{N-1} \leq \quad A \quad &< 0 \\ 0 \leq \quad B \quad &< 2^{N-1} \end{aligned}$$

We add these two inequalities and replace $A + B$ with C to obtain

$$-2^{N-1} \leq \quad C \quad < 2^{N-1}$$

But anything in the range specified by this inequality can be represented with N-bit 2's complement, and thus the addition does not overflow.

[1]This common mathematical phrasing means that we are using a problem symmetry to cut down the length of the proof discussion. In this case, the names A and B aren't particularly important, since addition is commutative ($A + B = B + A$). Thus the proof for the case in which A is negative (and B is not) is identical to the case in which B is negative (and A is not), except that all of the names are swapped. The term "without loss of generality" means that we consider the proof complete even with additional assumptions, in our case that $A < 0$ and $B \geq 0$.

We are now ready to state our main theorem. For convenience, let's use different names for the actual sum $C = A + B$ and the sum S returned from the add unit. We define S as the number represented by the bit pattern produced by the add unit. When overflow occurs, $S \neq C$, but we always have $(S = C) \bmod 2^N$.

Theorem: Addition of two N-bit 2's complement numbers A and B overflows if and only if one of the following conditions holds:

1. $A < 0$ and $B < 0$ and $S \geq 0$
2. $A \geq 0$ and $B \geq 0$ and $S < 0$

Proof: We once again start with the "if" direction. That is, if condition 1 or condition 2 holds, we have an overflow. The proofs are straightforward. Given condition 1, we can add the two inequalities $A < 0$ and $B < 0$ to obtain $C = A + B < 0$. But $S \geq 0$, so clearly $S \neq C$, thus overflow has occurred.

Similarly, if condition 2 holds, we can add the inequalities $A \geq 0$ and $B \geq 0$ to obtain $C = A + B \geq 0$. Here we have $S < 0$, so again $S \neq C$, and we have an overflow.

We must now prove the "only if" direction, showing that any overflow implies either condition 1 or condition 2. By the **contrapositive**[2] of our Lemma, we know that if an overflow occurs, either both operands are negative, or they are both positive.

Let's start with the case in which both operands are negative, so $A < 0$ and $B < 0$, and thus the real sum $C < 0$ as well. Given that A and B are represented as N-bit 2's complement, they must fall in the representable range, so we can write

$$\begin{aligned} -2^{N-1} \leq \quad & A \quad < 0 \\ -2^{N-1} \leq \quad & B \quad < 0 \end{aligned}$$

We add these two inequalities and replace $A + B$ with C to obtain

$$-2^N \leq \quad C \quad < 0$$

Given that an overflow has occurred, C must fall outside of the representable range. Given that $C < 0$, it cannot be larger than the largest possible number representable using N-bit 2's complement, so we can write

$$-2^N \leq \quad C \quad < -2^{N-1}$$

We now add 2^N to each part to obtain

$$0 \leq \quad C + 2^N \quad < 2^{N-1}$$

This range of integers falls within the representable range for N-bit 2's complement, so we can replace the middle expression with S (equal to C modulo 2^N) to find that

$$0 \leq \quad S \quad < 2^{N-1}$$

Thus, if we have an overflow and both $A < 0$ and $B < 0$, the resulting sum $S \geq 0$, and condition 1 holds.

The proof for the case in which we observe an overflow when both operands are non-negative ($A \geq 0$ and $B \geq 0$) is similar, and leads to condition 2. We again begin with inequalities for A and B:

$$\begin{aligned} 0 \leq \quad & A \quad < 2^{N-1} \\ 0 \leq \quad & B \quad < 2^{N-1} \end{aligned}$$

We add these two inequalities and replace $A + B$ with C to obtain

$$0 \leq \quad C < \quad 2^N$$

[2] If we have a statement of the form (p implies q), its contrapositive is the statement (not q implies not p). Both statements have the same truth value. In this case, we can turn our Lemma around as stated.

Given that an overflow has occurred, C must fall outside of the representable range. Given that $C \geq 0$, it cannot be smaller than the smallest possible number representable using N-bit 2's complement, so we can write

$$2^{N-1} \leq \quad C \quad < 2^N$$

We now subtract 2^N to each part to obtain

$$-2^{N-1} \leq \quad C - 2^N \quad < 0$$

This range of integers falls within the representable range for N-bit 2's complement, so we can replace the middle expression with S (equal to C modulo 2^N) to find that

$$-2^{N-1} \leq \quad S \quad < 0$$

Thus, if we have an overflow and both $A \geq 0$ and $B \geq 0$, the resulting sum $S < 0$, and condition 2 holds.

Thus overflow implies either condition 1 or condition 2, completing our proof.