Overflow: UB

Unsigned binary:

Add 2 non-negative numbers: result is greater than or equal to each number

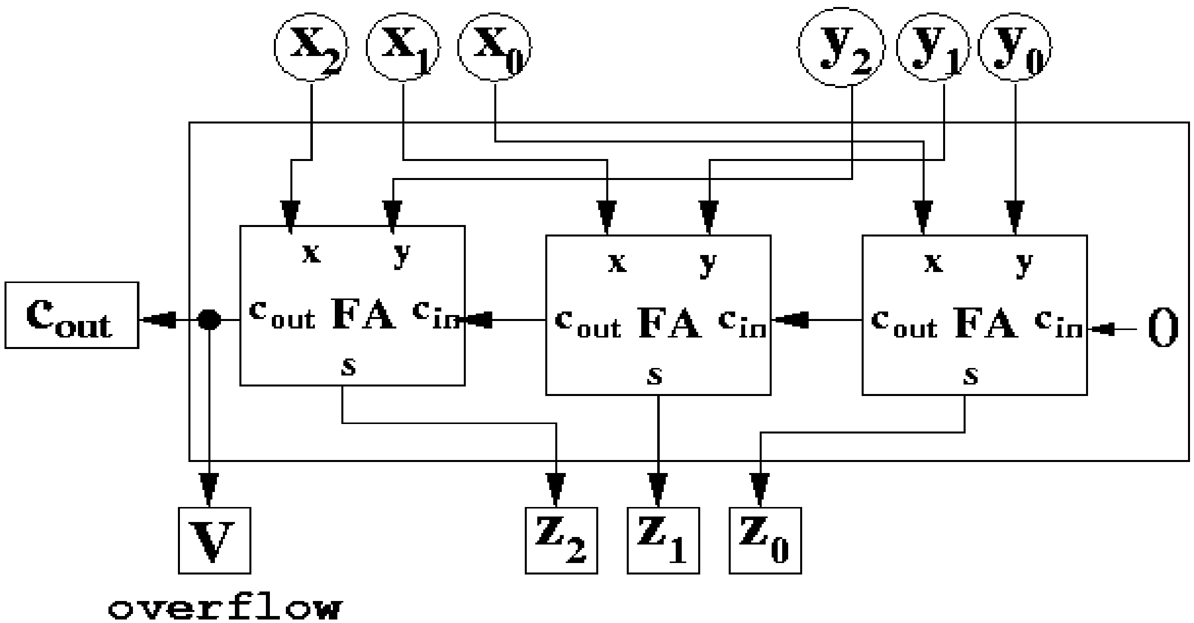
x + y >= x

x + y >= y

Overflow occurs when result is larger than maximum number (2^k - 1 for k bits)

Can detect overflow just by checking if carry out from most significant bit is 1

Ripple-carry circuit with overflow detection:



"V" is used to denote overflow bit ("O" is too close to "0")

Overflow: 2C

If x and y have opposite signs, then the result can't overflow:

magnitude of the result will be less than the magnitude of the larger number

$| x + y | <= \max (|x|, |y|)$

Overflow can only occur when the numbers both have the same sign.

If the sign of the result is different, then overflow must have occurred.

For example, if x and y both have sign bit 0 (positive), and the result has sign bit 1 (negative), then overflow must have occurred.

Add 2 k-bit numbers:

$$\begin{array}{r} x_{k-1}\ldots x_0 \\ +\ y_{k-1}\ldots y_0 \\ \hline s_{k-1}\ldots s_0 \end{array}$$

One way to express whether overflow occurs:

$V = x_{k-1}y_{k-1}\backslash s_{k-1} + \backslash x_{k-1}\backslash y_{k-1}s_{k-1}$

Either both sign bits of x and y are 1 and the sign bit of s is 0, or the sign bits are both 0 and the sign bit of s is 1

Simpler formula:

$V = c_{k-1}\ \mathrm{XOR}\ c_{k-2}$

The overflow bit is equal to the XOR of the carry-in to the leftmost bit with the carry-out from the leftmost bit.

Overflow: 2C

$V = c_{k-1}$ XOR c_{k-2} Why does this work?

Case 1: 0 carried in, 1 carried out

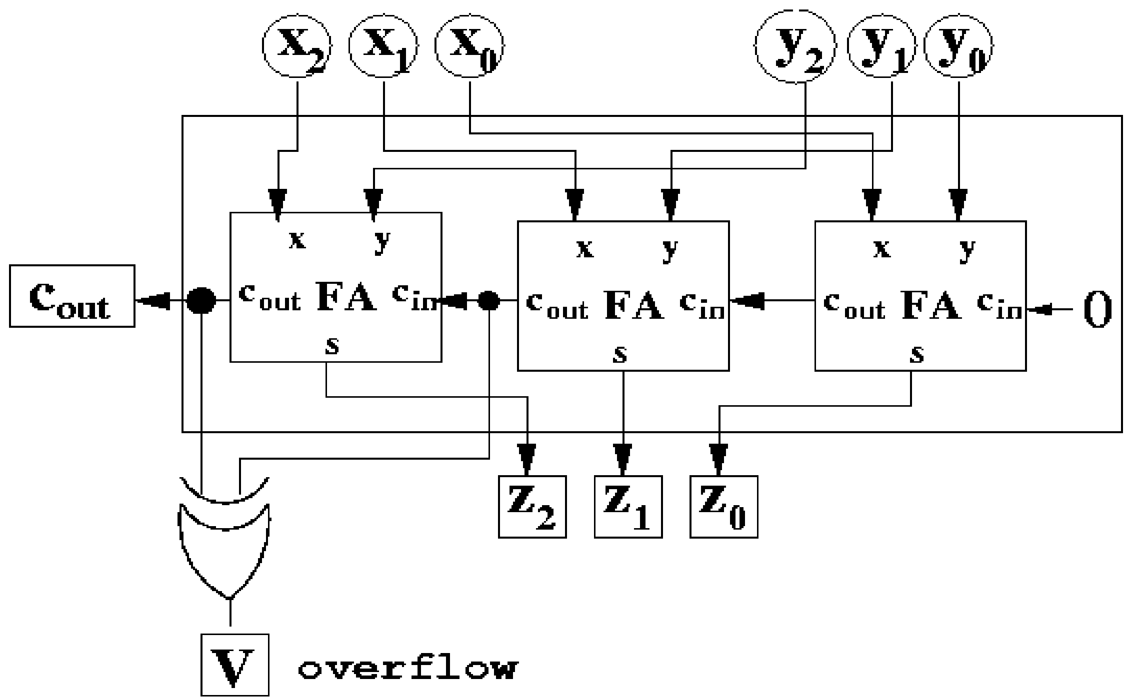
This occurs only when both x_{k-1} and y_{k-1} are 1, but then s_{k-1} is 0,

so the result is non-negative even though both x and y are negative.

Case 2: 1 carried in, 0 carried out

This occurs only when both x_{k-1} and y_{k-1} are 0, but then s_{k-1} is 1,

so the result is negative even though both x and y are non-negative.



Adder with overflow detection