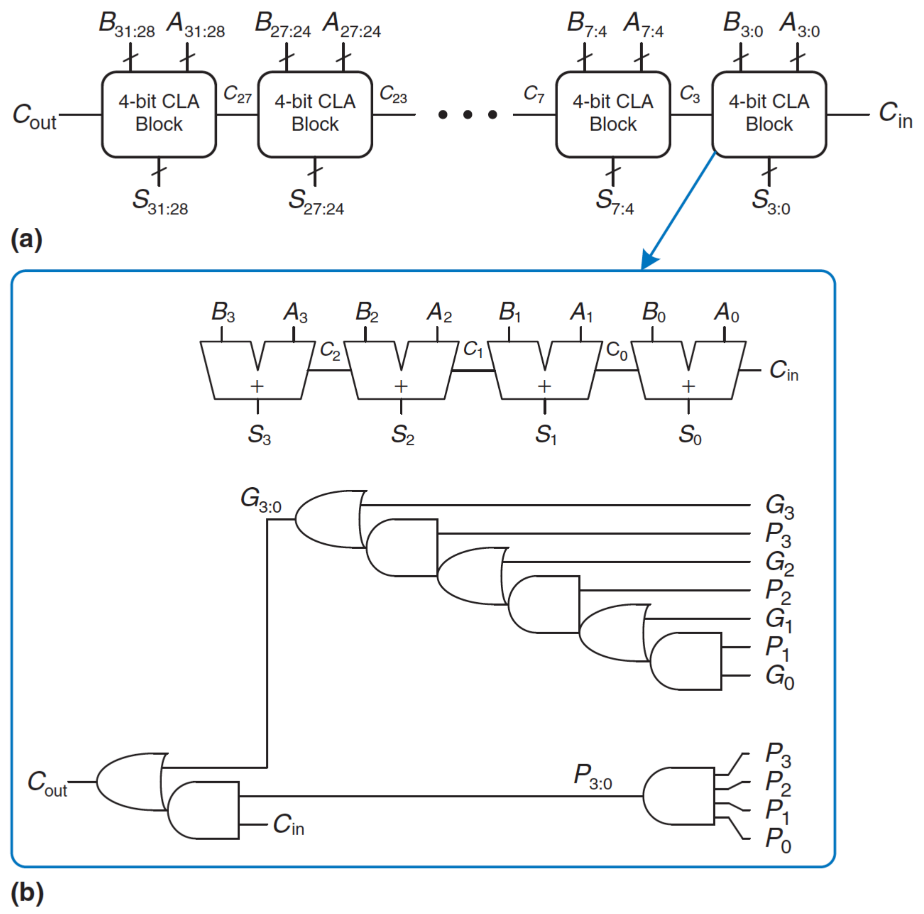
Running Time of Carry-Lookahead Adder

Running time of CLA

N = 16 bits

k = 4 bits



$$t_{CLA} = t_{pg} + t_{pg\_block} + \left(\frac{N}{k} - 1\right) t_{AND\_OR} + k t_{FA}$$

Running time of CLA

$$t_{CLA} = t_{pg} + t_{pg\_\text{block}} + \left(\frac{N}{k} - 1\right) t_{\text{AND\_OR}} + k t_{FA}$$

Not as easy to compare apples-to-apples against CSA because not entirely written in terms of T_{FA}

Completely possible to do, but requires more work than it's worth for this course

Instead, consider cost asymptotically and how it varies with block size

Running time of CLA

$$t_{CLA} = t_{pg} + t_{pg\_\mathrm{block}} + \left(\frac{N}{k} - 1\right) t_{\mathrm{AND\_OR}} + k t_{FA}$$

Same tradeoff as before:

Large k means we have more adders on critical path

Small k means carry needs to go through more blocks

Use the same trick and set k = sqrt(N) – overall time once again grows as O(sqrt(n))

For your project, CSA will generally be slightly faster

Running time of CLA vs Ripple-Carry

Compare delay of Ripple Carry and CLA.

Assume:

Full Adder Delay: 300ps

2 input gate delay: 100ps

N = 16 bits

k = 4 bits

$$t_{\text{ripple}} = N t_{FA}$$

$$t_{CLA} = t_{pg} + t_{pg\_\text{block}} + \left(\frac{N}{k} - 1\right) t_{\text{AND\_OR}} + k t_{FA}$$

Even faster adders

Idea of carry-lookahead can be repeated in hierarchical way to improve asymptotic time to O(log(n))

Your book calls these prefix adders, though some sources simply refer to them as carry-lookahead adders

We will not cover these in lecture, but they aren't *that* bad, and it is not uncommon for a few groups to implement one in JLS