In More Depth: Carry Save Adders

Exercises B.33 and B.34 motivate an organization that uses the 1-bit adder in Figure B.5.2 on page B-27 in a way it was not intended. Although this piece of hardware is simple and fast, the problem comes from trying to get the CarryIn signal calculated in a timely fashion across several adders.

We can think of the adder instead as a hardware device that can add three inputs together (a_i, b_i, c_i) and produce two outputs (s, c_{i+1}). When adding two numbers together, there is little we can do with this observation. When we are adding more than two operands, it is possible to reduce the cost of the carry. The idea is to form two independent sums, called S′ (sum bits) and C′ (carry bits). At the end of the process, we need to add C′ and S′ together using a normal adder. This technique of delaying carry propagation until the end of a sum of numbers is called *carry save addition*. The block drawing on the lower right of Figure B.14.1 shows the organization, with two levels of carry save adders connected by a single normal adder.

B.31 [10] <§B.6> Calculate the delays to add four 16-bit numbers using full carry-lookahead adders versus carry save with a carry lookahead adder forming the final sum. (The time unit T in Exercises B.28 and B.34 is the same.)

B.32 [20] <§B.6> Perhaps the most likely case of adding many numbers at once in a computer would be when trying to multiply more quickly by using many adders to add many numbers in a single clock cycle. Compared to the multiply algorithm in Figure 3.6 on page 178, a carry save scheme with many adders could multiply more than 10 times faster.

This exercise estimates the cost and speed of a combinational multiplier to multiply two positive 16-bit numbers. Assume that you have 16 intermediate terms M15, M14, . . . , M0, called *partial products*, that contain the multiplicand ANDed with multiplier bits m15, m14, . . . , m0.

The idea is to use carry save adders to reduce the n operands into $2n/3$ in parallel groups of three, and do this repeatedly until you get two large numbers to add together with a traditional adder.

First, show the block organization of the 16-bit carry save adders to add these 16 terms, as shown on the right in Figure B.14.1. Then calculate the delays to add these 16 numbers. Compare this time to the iterative multiplication scheme in Figure 3.6 on page 178 but only assume 16 iterations using a 16-bit adder that has full carry lookahead whose speed was calculated in Exercise 3.29.