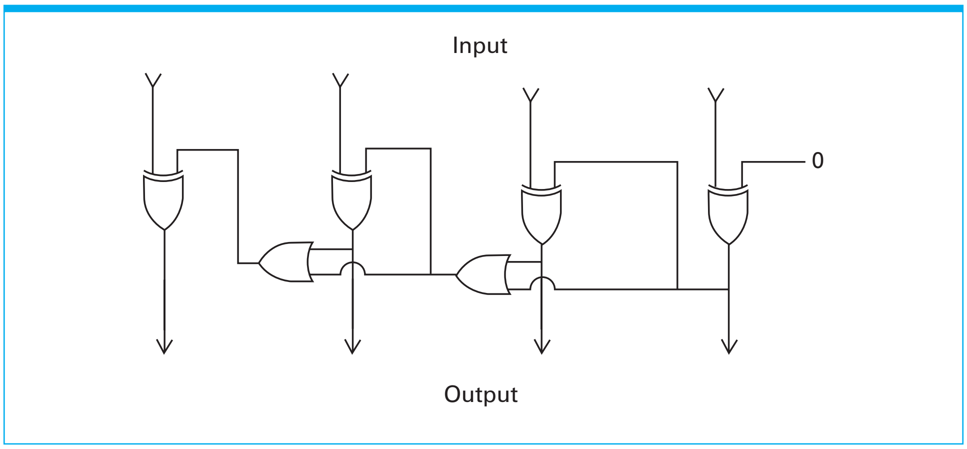
appendix B

Circuits to Manipulate Two's Complement Representations

This appendix presents circuits for negating and adding values represented in two's complement notation. We begin with the circuit in Figure B.1 that converts a four-bit two's complement representation to the representation for the negative of that value. For example, given the two's complement representation of 3, the circuit produces the representation for −3. It does this by following the same algorithm as presented in the text. That is, it copies the pattern from right to left until a 1 has been copied and then complements each remaining bit as it is moved from the input to the output. Because one input of the rightmost XOR gate is fixed at 0, this gate will merely pass its other input to the output. However, this output is also passed to the left as one of the inputs to the next XOR gate. If this output is 1, the next XOR gate will complement its input bit as it passes to the output. Moreover, this 1 will also be passed to the left through the OR gate to affect the next gate as well. In this manner, the first 1 that is copied to the output will also be passed to the left, where it will cause all the remaining bits to be complemented as they are moved to the output.

Figure B.1 A circuit that negates a two's complement pattern



Next, let us consider the process of adding two values represented in two's complement notation. In particular, when solving the problem

```
+ 0110
+ 1011
```

we proceed from right to left in a column-by-column manner, executing the same algorithm for each column. Thus once we obtain a circuit for adding one column of such a problem, we can construct a circuit for adding many columns merely by repeating the single-column circuit.

The algorithm for adding a single column in a multiple-column addition problem is to add the two values in the current column, add that sum to any carry from the previous column, write the least significant bit of this sum in the answer, and transfer any carry to the next column. The circuit in Figure B.2 follows this same algorithm. The upper XOR gate determines the sum of the two input bits. The lower XOR gate adds this sum to the value carried from the previous column. The two AND gates together with the OR gate pass any carry to the left. In particular, a carry of 1 will be produced if the original two input bits in this column were 1 or if the sum of these bits and the carry were both 1.

Figure B.2 A circuit to add a single column in a multiple-column addition problem

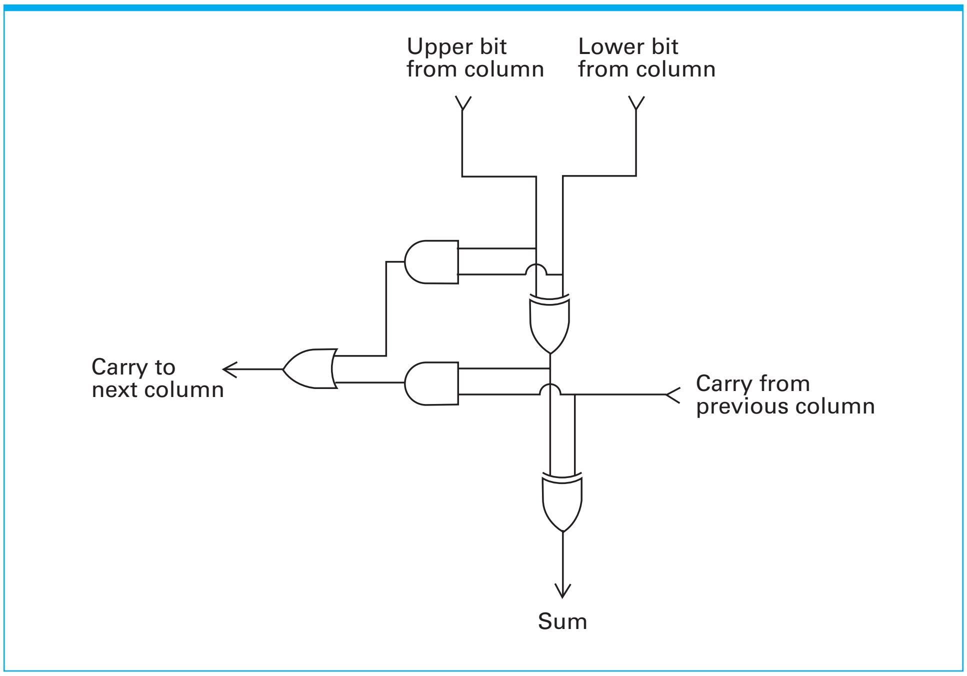
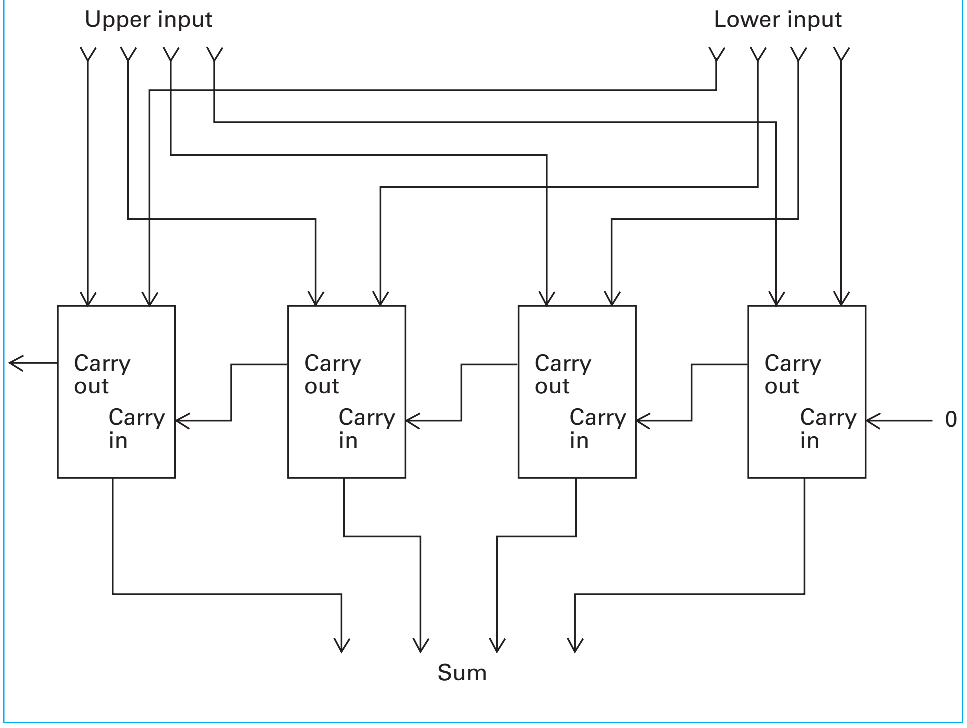


Figure B.3 shows how copies of this single-column circuit can be used to produce a circuit that computes the sum of two values represented in a four-bit two's complement system. Each rectangle represents a copy of the single-column addition circuit. Note that the carry value given to the rightmost rectangle is always 0 because there is no carry from a previous column. In a similar manner, the carry produced from the leftmost rectangle is ignored.

The circuit in Figure B.3 is known as a *ripple adder* because the carry information must propagate, or ripple, from the rightmost to the leftmost column. Although simple in composition, such circuits are slower to perform their functions than more clever versions, such as the lookahead carry adder, which minimize this column-to-column propagation. Thus the circuit in Figure B.3, although sufficient for our purposes, is not the circuit that is used in today's machines.

Figure B.3 A circuit for adding two values in a two's complement notation using four copies of the circuit in Figure B.2


Upper input
Lower input
Carry out
Carry in
Carry out
Carry in
Carry out
Carry in
Carry out
Carry in
0
Sum