

bit Q_{n+1} are initially cleared to 0 and the sequence counter SC is set to a number n equal to the number of bits in the multiplier. The two bits of the multiplier in Q_n and Q_{n+1} are inspected. If the two bits are equal to 10, it means that the first 1 in a string of 1's has been encountered. This requires a subtraction of the multiplicand from the partial product in AC . If the two bits are equal to 01, it means that the first 0 in a string of 0's has been encountered. This requires the addition of the multiplicand to the partial product in AC . When the two bits are equal, the partial product does not change. An overflow cannot occur because the addition and subtraction of the multiplicand follow each other. As a consequence, the two numbers that are added always have opposite signs, a condition that excludes an overflow. The next step is to shift right the partial product and the multiplier (including bit Q_{n+1}). This is an arithmetic shift right (ashr) operation which shifts AC and QR to the right and leaves the sign bit in AC unchanged (see Sec. 4-6). The sequence counter is decremented and the computational loop is repeated n times.

A numerical example of Booth algorithm is shown in Table 10-3 for $n = 5$. It shows the step-by-step multiplication of $(-9) \times (-13) = +117$. Note that the multiplier in QR is negative and that the multiplicand in BR is also negative. The 10-bit product appears in AC and QR and is positive. The final value of Q_{n+1} is the original sign bit of the multiplier and should not be taken as part of the product.

Array Multiplier

Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift microoperations. The multiplication of two binary numbers can be done with one microoperation by means of a combinational circuit that forms the product bits all

TABLE 10-3 Example of Multiplication with Booth Algorithm

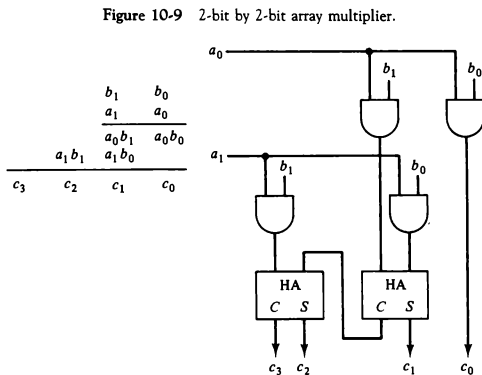
$Q_n Q_{n+1}$		$BR = 10111$ $\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
		Initial	00000	10011	0	101
1	0	Subtract BR	<u>01001</u> 01001			
		ashr	00100	11001	1	100
1	1	ashr	00010	01100	1	011
0	1	Add BR	<u>10111</u> 11001			
		ashr	11100	10110	0	010
0	0	ashr	11110	01011	0	001
1	0	Subtract BR	<u>01001</u> 00111			
		ashr	00011	10101	1	000

at once. This is a fast way of multiplying two numbers since all it takes is the time for the signals to propagate through the gates that form the multiplication array. However, an array multiplier requires a large number of gates, and for this reason it was not economical until the development of integrated circuits.

To see how an array multiplier can be implemented with a combinational circuit, consider the multiplication of two 2-bit numbers as shown in Fig. 10-9. The multiplicand bits are b_1 and b_0 , the multiplier bits are a_1 and a_0 , and the product is $c_3 c_2 c_1 c_0$. The first partial product is formed by multiplying a_0 by $b_1 b_0$. The multiplication of two bits such as a_0 and b_0 produces a 1 if both bits are 1; otherwise, it produces a 0. This is identical to an AND operation and can be implemented with an AND gate. As shown in the diagram, the first partial product is formed by means of two AND gates. The second partial product is formed by multiplying a_1 by $b_1 b_0$ and is shifted one position to the left. The two partial products are added with two half-adder (HA) circuits. Usually, there are more bits in the partial products and it will be necessary to use full-adders to produce the sum. Note that the least significant bit of the product does not have to go through an adder since it is formed by the output of the first AND gate.

A combinational circuit binary multiplier with more bits can be constructed in a similar fashion. A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier. The binary output in each level of AND gates is added in parallel with the partial product of the previous level to form a new partial product. The last level produces the product. For j multiplier bits and k multiplicand bits we need $j \times k$ AND gates and $(j - 1)$ k -bit adders to produce a product of $j + k$ bits.

As a second example, consider a multiplier circuit that multiplies a binary number of four bits with a number of three bits. Let the multiplicand be



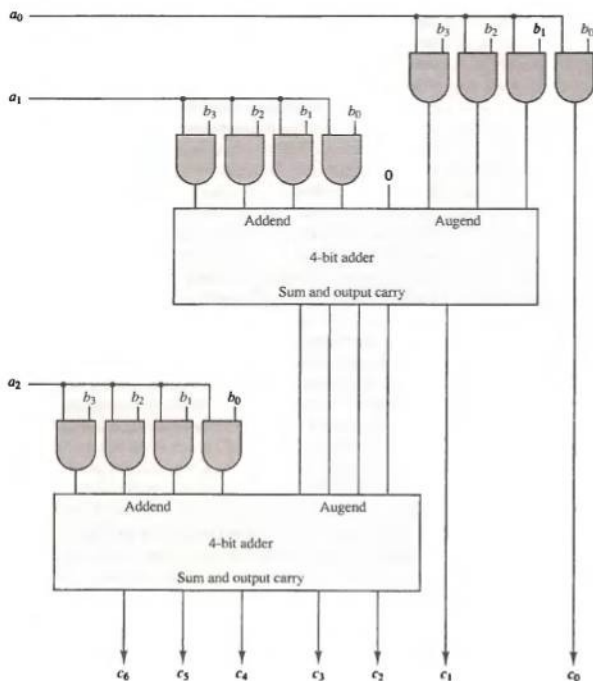


Figure 10-10 4-bit by 3-bit array multiplier.

represented by $b_3 b_2 b_1 b_0$ and the multiplier by $a_2 a_1 a_0$. Since $k = 4$ and $j = 3$, we need 12 AND gates and two 4-bit adders to produce a product of seven bits. The logic diagram of the multiplier is shown in Fig. 10-10.

10-4 Division Algorithms

Division of two fixed-point binary numbers in signed-magnitude representation is done with paper and pencil by a process of successive compare, shift, and subtract operations. Binary division is simpler than decimal division be-