3.2.1. Shift-and-Add Multiplication

Shift-and-add multiplication is similar to the multiplication performed by paper and pencil. This method adds the multiplicand *X* to itself *Y* times, where *Y* denotes the multiplier. To multiply two numbers by paper and pencil, the algorithm is to take the digits of the multiplier one at a time from right to left, multiplying the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

As an example, consider the multiplication of two unsigned 4-bit numbers, 8 (1000) and 9 (1001).

```
Multiplicand               1000 ×
Multiplier                 1001
                           1000
                        0000
                       0000
                      1000      _
Product               1001000
```

In the case of binary multiplication, since the digits are 0 and 1, each step of the multiplication is simple. If the multiplier digit is 1, a copy of the multiplicand (1 × multiplicand) is placed in the proper positions; if the multiplier digit is 0, a number of 0 digits (0 × multiplicand) are placed in the proper positions.

Consider the multiplication of positive numbers. The first version of the multiplier circuit, which implements the shift-and-add multiplication method for two *n*-bit numbers, is shown in Figure 3.11.

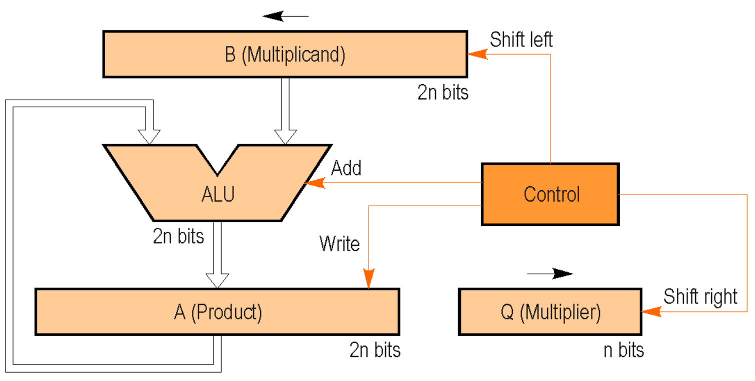

Figure 3.11. First version of the multiplier circuit.

The 2*n*-bit product register (*A*) is initialized to 0. Since the basic algorithm shifts the multiplicand register (*B*) left one position each step to align the multiplicand with the sum being accumulated in the product register, we use a 2*n*-bit multiplicand register with the multiplicand placed in the right half of the register and with 0 in the left half.

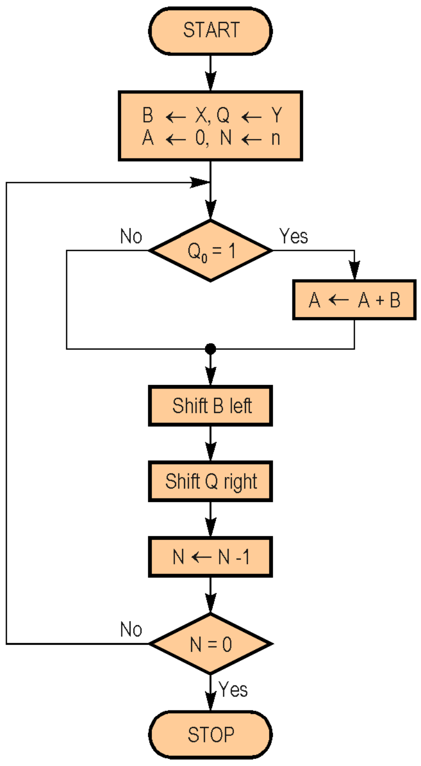

Figure 3.12. The first version of the multiplication algorithm.

Figure 3.12 shows the basic steps needed for the multiplication. The algorithm starts by loading the multiplicand into the *B* register, loading the multiplier into the *Q* register, and initializing the *A* register to 0. The counter *N* is initialized to *n*. The least significant bit of the multiplier register (Q_0) determines whether the multiplicand is added to the product register. The left shift of the multiplicand has the effect of shifting the intermediate products to the left, just as when multiplying by paper and pencil. The right shift of the multiplier prepares the next bit of the multiplier to examine in the following iteration.

Example 3.1

Using 4-bit numbers, perform the multiplication 9×12 (1001×1100).

Answer

Table 3.2 shows the value of registers for each step of the multiplication algorithm.

Table 3.2. Multiply example using the first version of the algorithm.

Step	A	Q	B	Operation
0	0000 0000	1100	0000 1001	Initialization
1	0000 0000	1100	0001 0010	Shift left B
	0000 0000	0110	0001 0010	Shift right Q
2	0000 0000	0110	0010 0100	Shift left B
	0000 0000	0011	0010 0100	Shift right Q
3	0010 0100	0011	0010 0100	Add B to A
	0010 0100	0011	0100 1000	Shift left B
	0010 0100	0001	0100 1000	Shift right Q
4	0110 1100	0001	0100 1000	Add B to A
	0110 1100	0001	1001 0000	Shift left B
	0110 1100	0000	1001 0000	Shift right Q

The original algorithm shifts the multiplicand left with zeros inserted in the new positions, so the least significant bits of the product cannot change after they are formed. Instead of shifting the multiplicand left, we can shift the product to the right. Therefore the multiplicand is fixed relative to the product, and since we are adding only *n* bits, the adder needs to be only *n* bits wide. Only the left half of the 2*n*-bit product register is changed during the addition.

Another observation is that the product register has an empty space with the size equal to that of the multiplier. As the empty space in the product register disappears, so do the bits of the multiplier. In consequence, the final version of the multiplier circuit combines the product (*A* register) with the multiplier (*Q* register). The *A* register is only *n* bits wide, and the product is formed in the *A* and *Q* registers. Figure 3.13 shows the new version of the circuit.

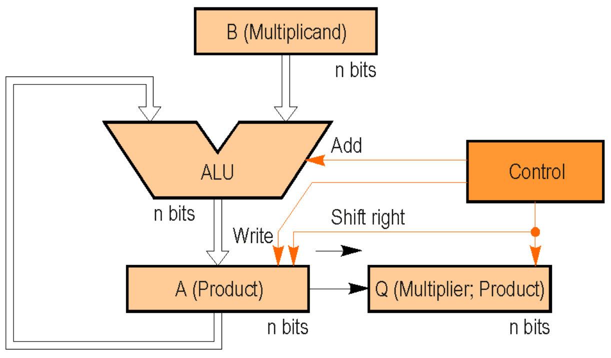

Figure 3.13. Final version of the multiplier circuit.

The final version of the multiplication algorithm is shown in Figure 3.14.

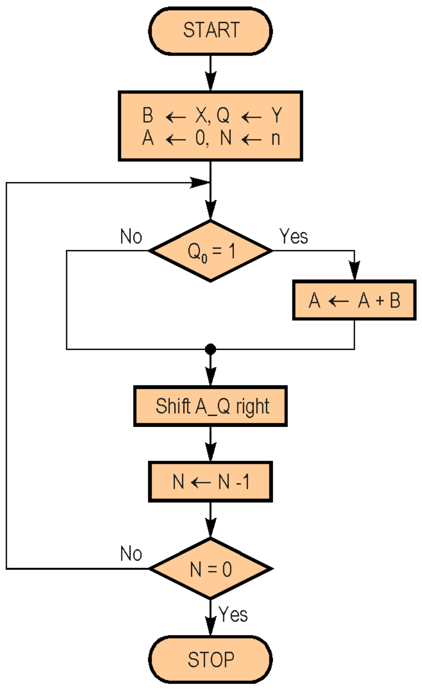

Figure 3.14. The final version of the multiplication algorithm.

Example 3.2

Perform the multiplication 9 × 12 (1001 × 1100) using the final version of the multiplication algorithm.

Answer

Table 3.3 shows the revised multiplication example for the final version of the algorithm.

Table 3.3. Multiply example using the final version of the algorithm.

Step	A	Q	B	Operation
0	0000	110$\underline{0}$	1001	Initialization
1	0000	011$\underline{0}$	1001	Shift right A_Q
2	0000	001$\underline{1}$	1001	Shift right A_Q
3	1001	001$\underline{1}$	1001	Add B to A
	0100	100$\underline{1}$	1001	Shift right A_Q
4	1101	100$\underline{1}$	1001	Add B to A
	0110	1100	1001	Shift right A_Q