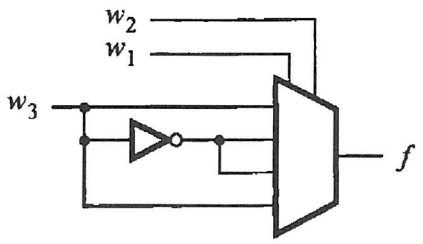
w_1	w_2	w_3	f	
0	0	0	0	w_3
0	0	1	1	
0	1	0	1	$\overline{w}_3$
0	1	1	0	
1	0	0	1	$\overline{w}_3$
1	0	1	0	
1	1	0	0	w_3
1	1	1	1	

(a) Truth table



(b) Circuit

Figure 6.10 Three-input XOR implemented with a 4-to-1 multiplexer.

6.1.2 MULTIPLEXER SYNTHESIS USING SHANNON'S EXPANSION

Figures 6.8 through 6.10 illustrate how truth tables can be interpreted to implement logic functions using multiplexers. In each case the inputs to the multiplexers are the constants 0 and 1, or some variable or its complement. Besides using such simple inputs, it is possible to connect more complex circuits as inputs to a multiplexer, allowing functions to be synthesized using a combination of multiplexers and other logic gates. Suppose that we want to implement the three-input majority function in Figure 6.8 using a 2-to-1 multiplexer in this way. Figure 6.11 shows an intuitive way of realizing this function. The truth table can be modified as shown on the right. If $w_1 = 0$, then $f = w_2w_3$, and if $w_1 = 1$, then $f = w_2 + w_3$. Using w_1 as the select input for a 2-to-1 multiplexer leads to the circuit in Figure 6.11*b*.

This implementation can be derived using algebraic manipulation as follows. The function in Figure 6.11*a* is expressed in sum-of-products form as

$$f = \overline{w}_1w_2w_3 + w_1\overline{w}_2w_3 + w_1w_2\overline{w}_3 + w_1w_2w_3$$

It can be manipulated into

$$\begin{aligned} f &= \overline{w}_1(w_2w_3) + w_1(\overline{w}_2w_3 + w_2\overline{w}_3 + w_2w_3) \\ &= \overline{w}_1(w_2w_3) + w_1(w_2 + w_3) \end{aligned}$$

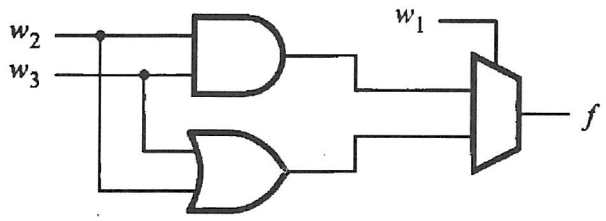
which corresponds to the circuit in Figure 6.11*b*.

Multiplexer implementations of logic functions require that a given function be decomposed in terms of the variables that are used as the select inputs. This can be accomplished by means of a theorem proposed by Claude Shannon [1].

w_1	w_2	w_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

w_1	f
0	w_2w_3
1	$w_2 + w_3$

(a) Truth table



(b) Circuit

Figure 6.11 The three-input majority function implemented using a 2-to-1 multiplexer.

Shannon's Expansion Theorem

Any Boolean function $f(w_1, \ldots, w_n)$ can be written in the form

$$f(w_1, w_2, \ldots, w_n) = \overline{w}_1 \cdot f(0, w_2, \ldots, w_n) + w_1 \cdot f(1, w_2, \ldots, w_n)$$

This expansion can be done in terms of any of the n variables. We will leave the proof of the theorem as an exercise for the reader (see problem 6.9).

To illustrate its use, we can apply the theorem to the three-input majority function, which can be written as

$$f(w_1, w_2, w_3) = w_1w_2 + w_1w_3 + w_2w_3$$

Expanding this function in terms of w_1 gives

$$f = \overline{w}_1(w_2w_3) + w_1(w_2 + w_3)$$

which is the expression that we derived above.

For the three-input XOR function, we have

$$\begin{aligned} f &= w_1 \oplus w_2 \oplus w_3 \\ &= \overline{w}_1 \cdot (w_2 \oplus w_3) + w_1 \cdot \overline{(w_2 \oplus w_3)} \end{aligned}$$

which gives the circuit in Figure 6.9*b*.

In Shannon's expansion the term $f(0, w_2, \ldots, w_n)$ is called the *cofactor* of f with respect to $\overline{w}_1$; it is denoted in shorthand notation as $f_{\overline{w}_1}$. Similarly, the term $f(1, w_2, \ldots, w_n)$ is called the cofactor of f with respect to w_1, written f_{w_1}. Hence we can write

$$f = \overline{w}_1 f_{\overline{w}_1} + w_1 f_{w_1}$$

In general, if the expansion is done with respect to variable w_i, then f_{w_i} denotes $f(w_1, \ldots, w_{i-1}, 1, w_{i+1}, \ldots, w_n)$ and

$$f(w_1, \ldots, w_n) = \overline{w}_i f_{\overline{w}_i} + w_i f_{w_i}$$

The complexity of the logic expression may vary, depending on which variable, w_i, is used, as illustrated in Example 6.5.

5.5 For the function $f = \overline{w}_1 w_3 + w_2 \overline{w}_3$, decomposition using w_1 gives

$$\begin{aligned} f &= \overline{w}_1 f_{\overline{w}_1} + w_1 f_{w_1} \\ &= \overline{w}_1 (w_3 + w_2) + w_1 (w_2 \overline{w}_3) \end{aligned}$$

Using w_2 instead of w_1 produces

$$\begin{aligned} f &= \overline{w}_2 f_{\overline{w}_2} + w_2 f_{w_2} \\ &= \overline{w}_2 (\overline{w}_1 w_3) + w_2 (\overline{w}_1 + \overline{w}_3) \end{aligned}$$

Finally, using w_3 gives

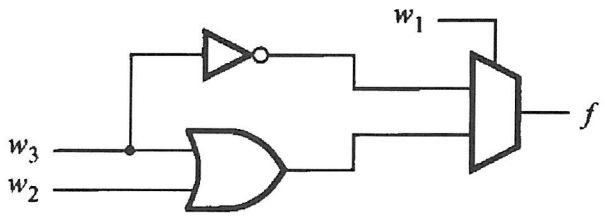
$$\begin{aligned} f &= \overline{w}_3 f_{\overline{w}_3} + w_3 f_{w_3} \\ &= \overline{w}_3 (w_2) + w_3 (\overline{w}_1) \end{aligned}$$

The results generated using w_1 and w_2 have the same cost, but the expression produced using w_3 has a lower cost. In practice, the CAD tools that perform decompositions of this type try a number of alternatives and choose the one that produces the best result.

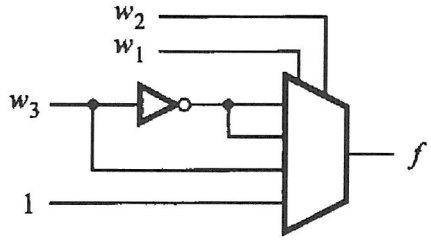
Shannon's expansion can be done in terms of more than one variable. For example, expanding a function in terms of w_1 and w_2 gives

$$\begin{aligned} f(w_1, \ldots, w_n) = \overline{w}_1 \overline{w}_2 \cdot f(0, 0, w_3, \ldots, w_n) + \overline{w}_1 w_2 \cdot f(0, 1, w_3, \ldots, w_n) \\ + w_1 \overline{w}_2 \cdot f(1, 0, w_3, \ldots, w_n) + w_1 w_2 \cdot f(1, 1, w_3, \ldots, w_n) \end{aligned}$$

This expansion gives a form that can be implemented using a 4-to-1 multiplexer. If Shannon's expansion is done in terms of all n variables, then the result is the canonical sum-of-products form, which was defined in section 2.6.1.



(a) Using a 2-to-1 multiplexer



(b) Using a 4-to-1 multiplexer

Figure 6.12 The circuits synthesized in Example 6.6.

Example

Assume that we wish to implement the function

$$f = \overline{w}_1\overline{w}_3 + w_1w_2 + w_1w_3$$

using a 2-to-1 multiplexer and any other necessary gates. Shannon's expansion using w_1 gives

$$\begin{aligned} f &= \overline{w}_1 f_{\overline{w}_1} + w_1 f_{w_1} \\ &= \overline{w}_1(\overline{w}_3) + w_1(w_2 + w_3) \end{aligned}$$

The corresponding circuit is shown in Figure 6.12*a*. Assume now that we wish to use a 4-to-1 multiplexer instead. Further decomposition using w_2 gives

$$\begin{aligned} f &= \overline{w}_1\overline{w}_2 f_{\overline{w}_1\overline{w}_2} + \overline{w}_1 w_2 f_{\overline{w}_1 w_2} + w_1\overline{w}_2 f_{w_1\overline{w}_2} + w_1w_2 f_{w_1w_2} \\ &= \overline{w}_1\overline{w}_2(\overline{w}_3) + \overline{w}_1 w_2(\overline{w}_3) + w_1\overline{w}_2(w_3) + w_1w_2(1) \end{aligned}$$

The circuit is shown in Figure 6.12*b*.

Example

Consider the three-input majority function

$$f = w_1w_2 + w_1w_3 + w_2w_3$$

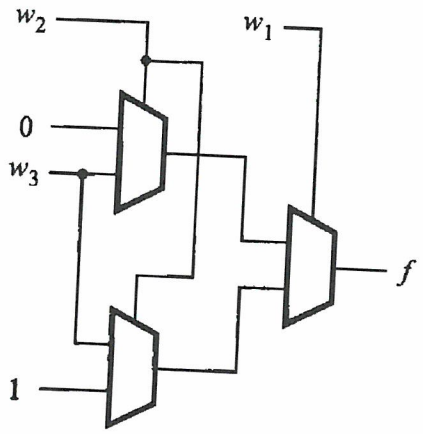


Figure 6.13 The circuit synthesized in Example 6.7.

We wish to implement this function using only 2-to-1 multiplexers. Shannon's expansion using w_1 yields

$$\begin{aligned} f &= \overline{w}_1(w_2w_3) + w_1(w_2 + w_3 + w_2w_3) \\ &= \overline{w}_1(w_2w_3) + w_1(w_2 + w_3) \end{aligned}$$

Let $g = w_2w_3$ and $h = w_2 + w_3$. Expansion of both g and h using w_2 gives

$$\begin{aligned} g &= \overline{w}_2(0) + w_2(w_3) \\ h &= \overline{w}_2(w_3) + w_2(1) \end{aligned}$$

The corresponding circuit is shown in Figure 6.13. It is equivalent to the 4-to-1 multiplexer circuit derived using a truth table in Figure 6.8.