corresponding to the input values `1010` – `1111`. But these input values are illegal in BCD, so these outputs are simply ignored.

It is common for decoders to have an additional input that is used to enable the output. The truth table in Table 5.2 shows a decoder with a 3-bit input, an enable line, and an 8-bit (2^3) output. The output is 0 whenever $enable = 0$. When $enable = 1$, the i^{th}

$enable$	x_2	x_1	x_0	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Table 5.2: Truth table for a 3×8 decoder with $enable$. If $enable = 0$, $y = 0$. If $enable = 1$, $x = i \Rightarrow y_i = 1$ and $y_j = 0$ for all $j \neq i$.

output bit is 1 if and only if the binary value of the input is equal to i. For example, when $enable = 1$ and $x = 011_2$, $y = 00001000_2$. That is,

$$\begin{aligned} y_3 &= x_2' \cdot x_1 \cdot x_0 \\ &= m_3 \end{aligned} \tag{5.7}$$

This clearly generalizes such that we can give the following description of a decoder:

1. For n input bits (excluding an $enable$ bit) there are 2^n output bits.
2. The i^{th} output bit is equal to the i^{th} minterm for the n input bits.

The 3×8 decoder specified in Table 5.2 can be implemented with 4-input AND gates as shown in Figure 5.6.

Decoders are more versatile than it might seem at first glance. Each possible input can be seen as a minterm. Since each output is one only when a particular minterm evaluates to one, a decoder can be viewed as a "minterm generator." We know that any logical expression can be represented as the OR of minterms, so it follows that we can implement any logical expression by ORing the output(s) of a decoder.

For example, let us rewrite Equation 5.1 for the Sum expression of a full adder using minterm notation (see Section 4.3.2):

$$Sum_i(Carry_i, x_i, y_i) = m_1 + m_2 + m_4 + m_7 \tag{5.8}$$

And for the Carry expression:

$$Carry_{i+1}(Carry_i, x_i, y_i) = m_3 + m_5 + m_6 + m_7 \tag{5.9}$$

where the subscripts on x, y, and $Carry$ refer to the bit slice and the subscripts on m are part of the minterm notation. We can implement a full adder with a 3×8 decoder and two 4-input OR gates, as shown in Figure 5.7.