CHAPTER 7

MODULAR CIRCUITS

A modular circuit is a digital circuit that performs a specific function or has certain usage. The modular circuits to be introduced in this chapter are decoders, encoders, multiplexers, and de-multiplexers.

7.1 Decoders

7.1.1 Decoder Structure

A decoder is a circuit that maps an input state or a combination of input values to one and only one of a number of outputs. For a decoder with n inputs, the number of input states is 2^n. To map each input state to one distinctive output, the circuit has 2^n outputs. Such a circuit is called an n-to-2^n decoder. The first number n and the second number 2^n are the numbers of inputs and outputs respectively. Mapping may also be for a number of input states less than 2^n. The decoder is called an n-to-m decoder and $m \leq 2^n$. For example, the binary input combination is a BCD code in a 4-to-10 BCD-to-decimal decoder. Six of the input states are don't-care states that never occur. Thus the decoder maps each of the ten valid BCD codes to one of ten outputs.

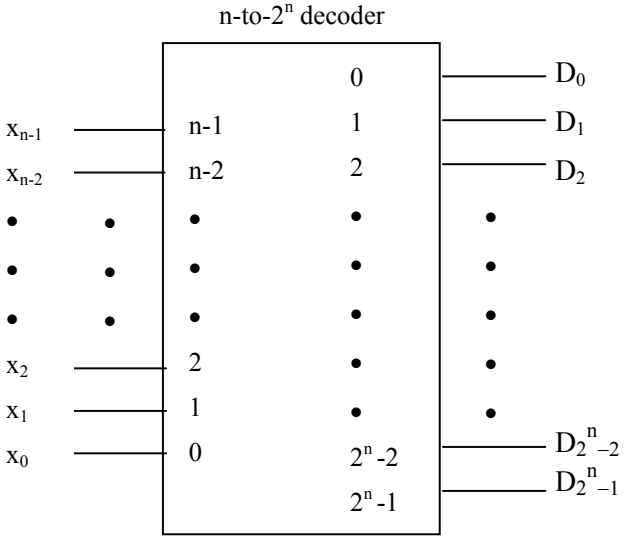


Figure 7.1 n-to-2^n decoder.

Table 7.1 Truth table for a 3-to-8 decoder.

x_2 x_1 x_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0 0 0	1	0	0	0	0	0	0	0
0 0 1	0	1	0	0	0	0	0	0
0 1 0	0	0	1	0	0	0	0	0
0 1 1	0	0	0	1	0	0	0	0
1 0 0	0	0	0	0	1	0	0	0
1 0 1	0	0	0	0	0	1	0	0
1 1 0	0	0	0	0	0	0	1	0
1 1 1	0	0	0	0	0	0	0	1

The mapping of an input state to an output is identified by a value of 1 at the mapped output. For each input state applied to a decoder, only one output is equal to 1.

All the other outputs are equal to 0. The output with a value of 1 is said to be asserted. Those with a value of 0 are said to be de-asserted. An n-to-2^n decoder is shown in Figure 7.1. To identify the mapping of an input state to an output, each input x_i is given a weight of 2^i where i = 0, 1, ..., n-2, n-1, so that a combination of binary input values can be converted to a decimal number. The weights are labeled at the inputs using the values of i, not 2^i. The outputs are numbered in decimal from 0 to 2^{n-1}. When a combination of binary input values is applied to the decoder, the output with a decimal number equal to the binary inputs is equal to 1. The truth table of a 3-to-8 decoder is given in Table 7.1. As shown in Table 7.1, the weights of x_2, x_1, and x_0 are 2^2, 2^1, and 2^0 respectively. As an example, D_3 is equal to 1 or asserted if $x_2x_1x_0 = 011$.

To build a decoder with discrete gates, the 3-to-8 decoder is used as an example. Each of the eight outputs D_0, D_1, ..., D_7 can be considered as an output independent of the other seven. It is obvious that each output is a function of only one minterm. Thus

$$D_i = m_i$$

All eight minterms of the three inputs are the eight outputs of the decoder. Since each minterm can be implemented by an AND gate provided the inputs are double-rail, an n-to-m decoder consists of an array of m AND gates.

One of the applications of decoders is to identify a location in a memory. The address of a memory location is specified by the inputs of a decoder. The location is pointed by the asserted output of a decoder

7.1.2 Decoders with Active-Low Outputs

Decoders are sometimes built with NAND gates instead of AND gates. Thus all the outputs are inverted. An input state is mapped to a distinct output of 0. This output is said to be asserted. All the other outputs are de-asserted and each has a value of 1. Inversions are shown by adding a bubble to each output in Figure 7.2. The outputs are also said to be active-low. Decoders built with AND gates have active-high outputs. The truth table for a 3-to-8 decoder with active-low outputs is given in Table 7.2.

For an active-high-output decoder, each output is a minterm. When the output is active-low, the same output that was a minterm becomes a maxterm for the same input state. That is

$$D_i = M_i$$

7.1.3 Decoders with Enable Control

Enable, or sometimes called strobe, is an input signal used to activate (enable) or de-activate (disable) a decoder. When the enable signal is asserted, a decoder is activated.

One of the decoder outputs is asserted, which has a value of 1 if the decoder has active-high outputs and a value of 0 for a decoder with active-low outputs. All the other outputs are de-asserted. When the enable signal is de-asserted, a decoder is disabled and does not work as a decoder. All the decoder outputs are de-asserted. A 2-to-4 decoder with active-high outputs and an active-high enable signal EN is given in Figure 7.3 as an example. Its corresponding truth table is given in Table 7.3. A don't-care value is denoted by "d" in Table 7.3. The outputs of a decoder are minterms of the inputs only if EN = 1. The Boolean expressions for the outputs are

$$D_0 = EN\ (x_1' x_0')$$
$$D_1 = EN\ (x_1' x_0)$$
$$D_2 = EN\ (x_1 x_0')$$
$$D_3 = EN\ (x_1 x_0)$$

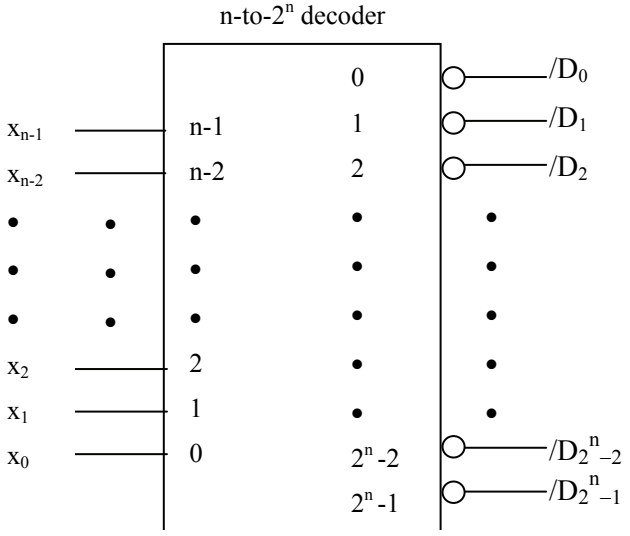


Figure 7.2 n-to-2^n decoder with active-low outputs.

Table 7.2 Truth table for a 3-to-8 decoder with active-low output.

$x_2\ x_1\ x_0$	$/D_0$	$/D_1$	$/D_2$	$/D_3$	$/D_4$	$/D_5$	$/D_6$	$/D_7$
0 0 0	0	1	1	1	1	1	1	1
0 0 1	1	0	1	1	1	1	1	1
0 1 0	1	1	0	1	1	1	1	1
0 1 1	1	1	1	0	1	1	1	1
1 0 0	1	1	1	1	0	1	1	1
1 0 1	1	1	1	1	1	0	1	1
1 1 0	1	1	1	1	1	1	0	1
1 1 1	1	1	1	1	1	1	1	0

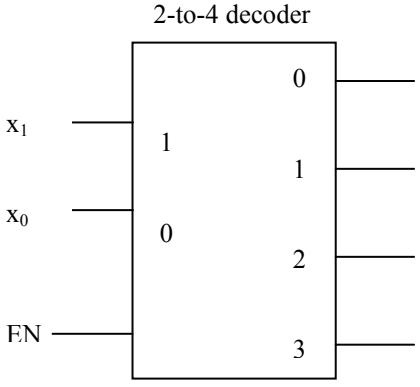


Figure 7.3 2-to-4 decoder with enable control.

Table 7.3 Truth table for Figure 7.3.

EN	$x_1\ x_0$	$D_0\ D_1\ D_2\ D_3$
0	d d	0 0 0 0
1	0 0	1 0 0 0
1	0 1	0 1 0 0
1	1 0	0 0 1 0
1	1 1	0 0 0 1

7.1.4 Expansion of Decoders

Multiple smaller size decoders sometimes are used to construct a large size decoder when the latter is not available. For example, the 3-to-8 decoder in Table 7.1 can be constructed using two 2-to-4 decoders with enable control. The circuit is shown in Figure 7.4(a). Note that only one of the two decoders is enabled for each input state. The asserted output of the 3-to-8 decoder is one of the four outputs of the enable decoder. By enabling one decoder and disabling the other, only one of the eight outputs is asserted. The circuit in Figure 7.4(b) is identical to the one in Figure 7.4(a). However, the output mapping is different from that of Figure 7.4(a) because the inputs are connected differently.

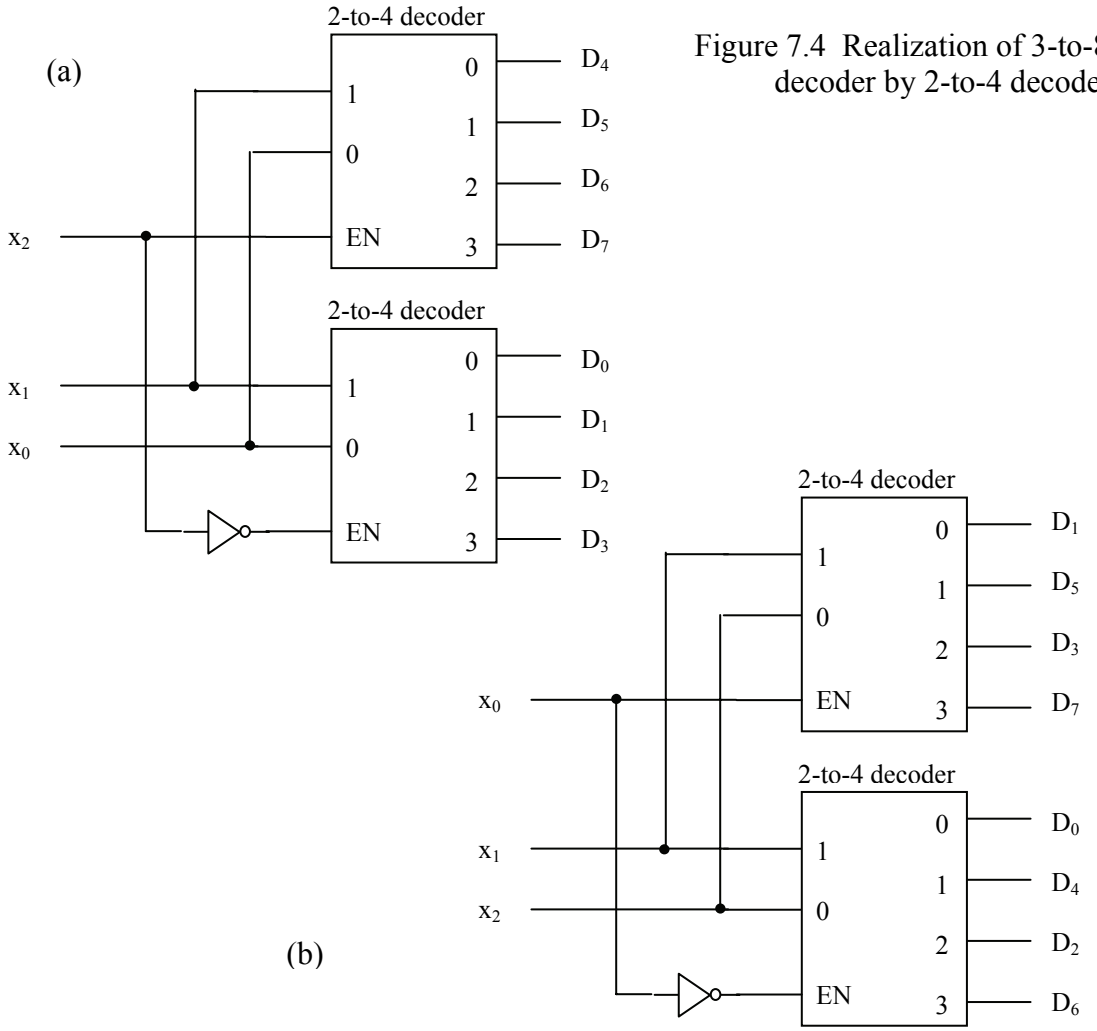


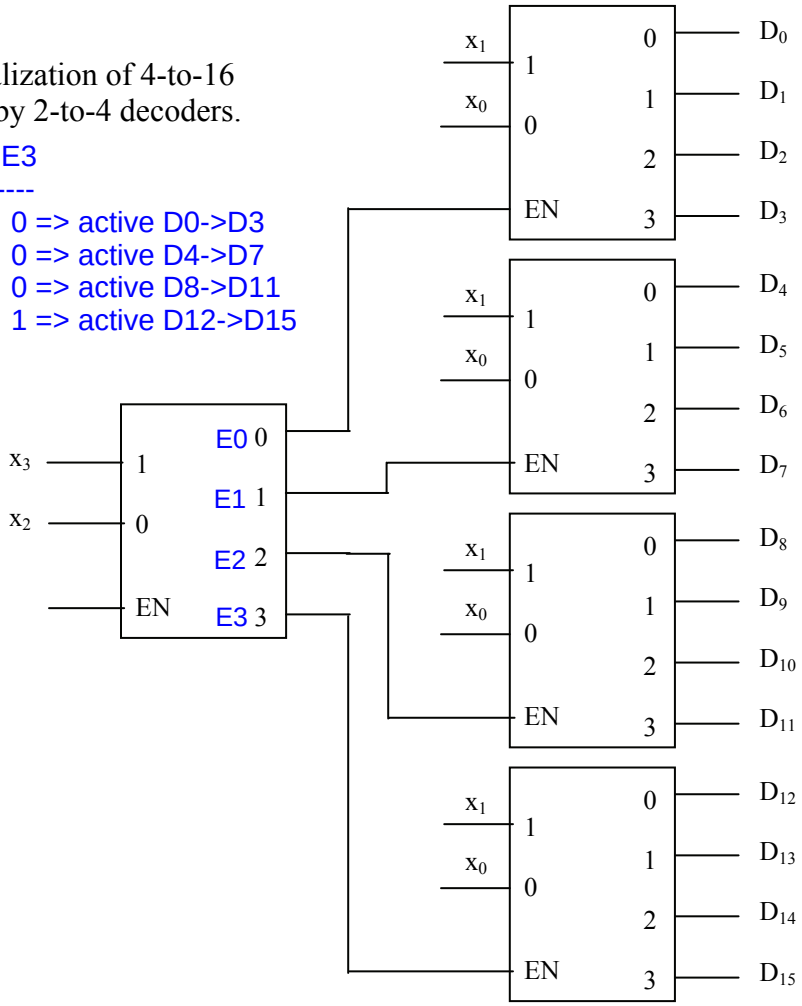
Figure 7.4 Realization of 3-to-8 decoder by 2-to-4 decoders.

Figure 7.5 is the realization of a 4-to-16 decoder using five 2-to-4 decoders. The inputs to the 4-to-16 decoder are $x_3x_2x_1x_0$. The decoders are arranged in the form of a 2-level tree. The decoder outputs in the first level are used to enable one decoder and disable three others in the second level. All the outputs of the three disabled decoders are

de-asserted. Three of the four outputs of the enabled 2-to-4 decoder are also de-asserted. Therefore only one of the sixteen outputs in the second level is asserted. The enable control of the decoder in the first level is the enable control for the 4-to-16 decoder. When this enable input is de-asserted, all sixteen outputs in the second level are de-asserted. The circuit can be expanded to a 6-to-64 decoder by adding a third level of sixteen 2-to-4 decoders. These sixteen decoders are controlled by the outputs of the decoders in the second level. Each decoder output in the second level is used to enable or disable a 4-to-2 decoder in the third level.

Figure 7.5 Realization of 4-to-16 decoder by 2-to-4 decoders.

x3	x2	E0	E1	E2	E3	
0	0	1	0	0		0 => active D0->D3
0	1	0	1	0		0 => active D4->D7
1	0	0	0	1		0 => active D8->D11
1	1	0	0	0		1 => active D12->D15



7.1.5 Implementation of Functions Using Decoders

Since the (active-high) decoder outputs are minterms of the decoder inputs, decoders can be used to implement functions. Implementation using a 3-to-8 decoder for a 3-variable function F(A,B,C) is shown in Figure 7.6.

$$F(A, B, C) = \Sigma\, m(0, 3, 5)$$

The implementation in Figure 7.6 is very straightforward. Minterms are generated by connecting A, B, and C to the decoder inputs. The decoder outputs are connected to an external OR-gate according to the minterm list of F.

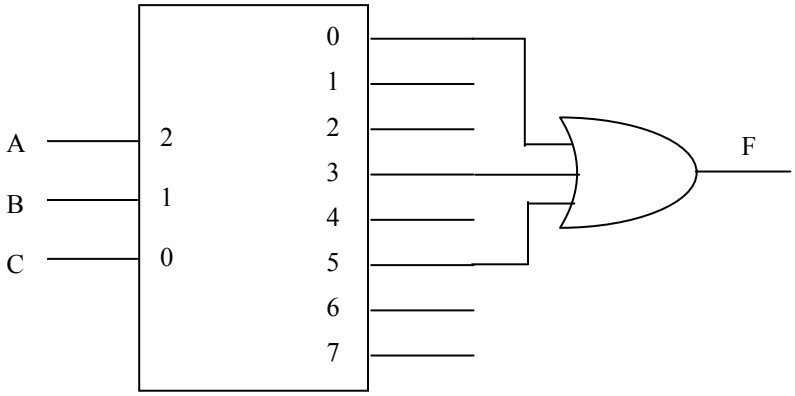


Figure 7.6 Function realization using decoder and OR gate.

F can also be realized using a decoder and an external NOR gate. As shown in Figure 7.7, imagine that the bubble at the NOR gate output was separable. Then the signal at the separated OR gate output or before the separated bubble is F'. Because

$$F'(A,B,C) = \pi M(0, 3, 5) = \Sigma m(1, 2, 4, 6, 7)$$

D_1, D_2, D_4, D_6, and D_7 are connected to the separated OR gate inputs to produce F' at the OR output. The bubble could now be attached to the OR gate to produce F at the NOR gate output. The implementation by a decoder and a NOR gates is equivalent to a 2-level AND-NOR or AOI circuit.

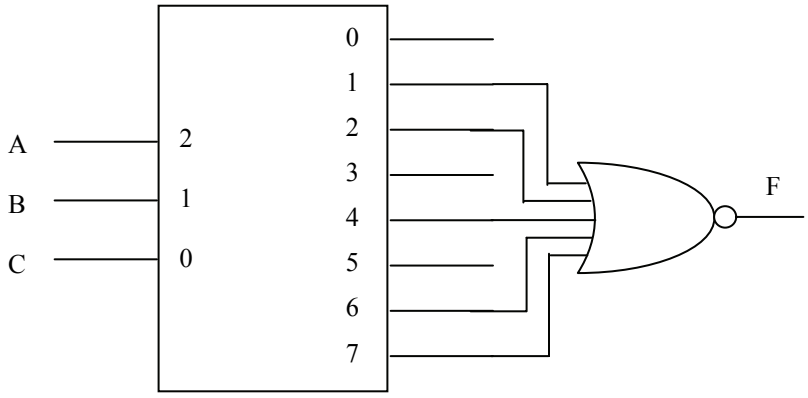


Figure 7.7 Function realization using decoder and NOR gate.

Neither AND gates nor NAND gates can be used as external gates. The inputs of any two minterms to an AND gate will generate an output of 0. If two minterms are inputted to a NAND gate, the output is always equal to 1.

Decoder with active-low outputs can also be used in realizing functions. However, the outputs are maxterms, not minterms, of the decoder inputs. The added external gates can no longer be OR gates and NOR gates. NAND gates and AND gates should be used instead.

The 3-variable function realized in Figures 7.6 and 7.7 using a decoder with active-high outputs is again used as an example.

$$F(A,B,C) = \Sigma\, m(0, 3, 5) = \pi\, M(1, 2, 4, 6, 7)$$

The function can be realized simply by connecting the five maxterms of the function to an external AND gate, which is shown in Figure 7.8.

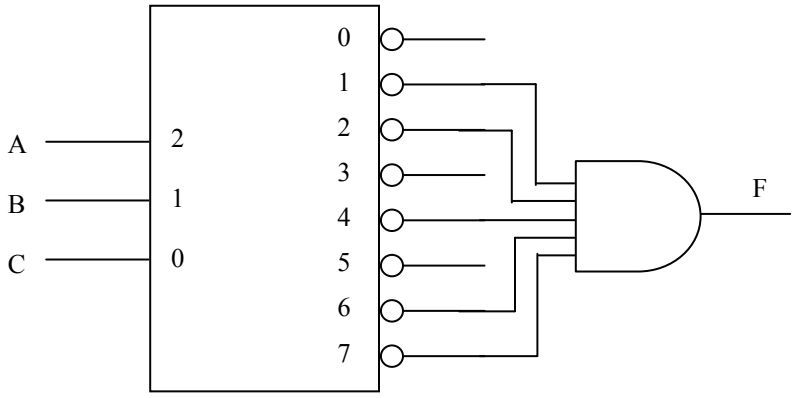


Figure 7.8 Function realization using decoder with active-low outputs and AND gate.

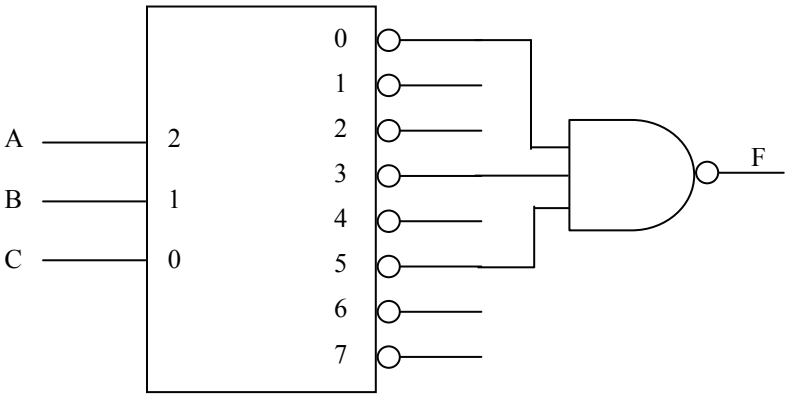


Figure 7.9 Function realization using decoder with active-low outputs and NAND gate.

An external NAND gate can also be used in function realization. Because the decoder with active-high outputs in Figure 7.6 is an AND array, the circuit is a 2-level AND-OR circuit. The 2-level AND-OR circuit can be converted to a 2-level NAND-NAND circuit by replacing the decoder in Figure 7.6 with an active-low-output decoder, which is a NAND array, and by changing the OR gate to a NAND gate. The equivalent realization is shown in Figure 7.9.

7.2 Encoders

7.2.1 Encoder Structure

An encoder is a circuit that reverses the function of a decoder. The roles of inputs and outputs of a decoder will exchange in an encoder. For a set of standard inputs to an encoder, one and only one input is asserted. The asserted input is equal to 1 for active-high inputs. All the other inputs have a value of 0. An encoder with n inputs and m outputs is called an n-to-m encoder. For an n-to-m encoder, the minimum m is the smallest integer equal to or greater than $\log_2(n)$. For example, if n = 4, $\log_2(n) = m = 2$. If n = 10, $\log_2(n) = 3.32$, and m = 4. A 4-to-2 encoder is shown in Figure 7.10. Its truth table is given in Table 7.4.

The truth table for the 4-to-2 encoder lists only the four standard sets of inputs, The assumption is that the other twelve input states never occur. With D_3 as the most significant variable, the minterm lists for x_1 and x_0 are

$$x_1 = \Sigma\, m(4, 8) + d(0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15)$$

$$x_0 = \Sigma\, m(2, 8) + d(0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15)$$

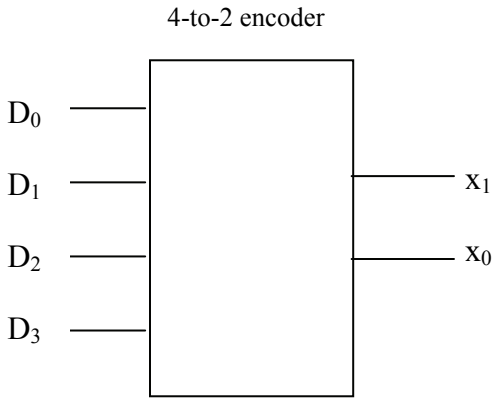


Figure 7.10 4-to-2 encoder.

Table 7.4 Truth table for a 4-to-2 encoder.

D_3	D_2	D_1	D_0	x_1	x_0
1	0	0	0	1	1
0	1	0	0	1	0
0	0	1	0	0	1
0	0	0	1	0	0

Karnaugh maps can be used to obtain the simplest sum-of-products expressions, which are

$$x_1 = D_3 + D_2$$

$$x_0 = D_3 + D_1$$

Although two outputs are sufficient to encode the four standard input states, an extra output x_2 is added to the output of the 4-to-2 encoder in Figure 7.10 so that a non-standard or don't-care input state can be recognized by a value of 0 at x_2. The diagram of

a 4-to-3 encoder is shown in Figure 7.11. Its truth table is given in Table 7.5. The last row is a condensed representation of all the other twelve non-standard input states. The encoded values of x_1 and x_0 are always 0 for a non-standard input state. The minterm list representations of the outputs and their corresponding sum-of-products expressions are given below.

$$x_2 = \Sigma\, m(1, 2, 4, 8) = D_3'D_2'D_1'D_0 + D_3'D_2'D_1D_0' + D_3'D_2D_1'D_0' + D_3D_2'D_1'D_0'$$

$$x_1 = \Sigma\, m(4, 8) = D_3'D_2D_1'D_0' + D_3D_2'D_1'D_0'$$

$$x_0 = \Sigma\, m(2, 8) = D_3'D_2'D_1D_0' + D_3D_2'D_1'D_0'$$

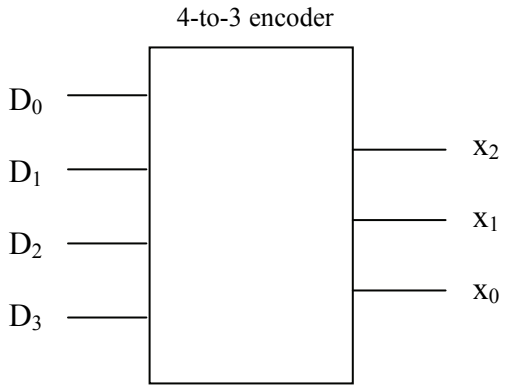


Figure 7.11 4-to-3 encoder.

Table 7.5 Truth table for a 4-to-3 encoder.

D_3 D_2 D_1 D_0	x_2 x_1 x_0
1 0 0 0	1 1 1
0 1 0 0	1 1 0
0 0 1 0	1 0 1
0 0 0 1	1 0 0
All other input states	0 0 0

7.2.2 Priority Encoders

A priority encoder is an encoder which can have more than one asserted input. In other words, a standard input state is not limited to just one input with a value of 1. Such an input state is not considered as non-standard. As a matter of fact, each of the inputs is given a priority. When there is more than one asserted input, only the asserted input with the highest priority is encoded. The priority given to the inputs of the encoder is in the order of D_3, D_2, D_1, and D_0. The diagram of a priority encoder is given in Figure 7.12. Its truth table is given in Table 7.6. As shown in the truth table, when D_3 =1, it is encoded regardless of the values of the other three inputs because it has the highest priority. When D_2 is asserted, it cannot be encoded unless D_3 is not asserted. An active-high output G is provided to show that at least one input is asserted. When no input is asserted, G is 0 and the encoded outputs are 00.

The minterm list forms of the priority encoder outputs and their simplest sum-of-products expressions are given below.

$x_1 = \Sigma\, m(4 - 7, 8 - 15) = D_3 + D_2$

$x_0 = \Sigma\, m(2, 3, 8 - 15) = D_3 + D_2'D_1$

$G = \Sigma\, m(1\text{-}15) = M_0 = D_3 + D_2 + D_1 + D_0$

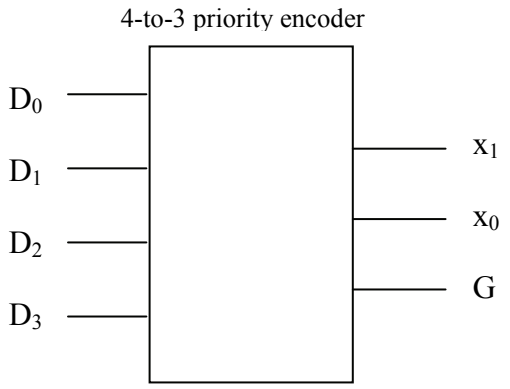


Figure 7.12 4-to-3 priority encoder.

Table 7.6 Truth table for a 4-to-3 priority encoder.

D_3	D_2	D_1	D_0	x_1	x_0	G
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	d	0	1	1
0	1	d	d	1	0	1
1	d	d	d	1	1	1

7.3 Multiplexers

7.3.1 Multiplexer Structure

A multiplexer is a combinational circuit that selects one of a number of inputs and transmits it to the output. The inputs to be selected are called data inputs. The selection of data inputs is determined by another set of inputs known as selection inputs or control inputs. Thus a multiplexer is also called a data selector. It is also abbreviated as "MUX". The logic symbol of a 2^n-to-1 multiplexer is shown in Figure 7.13. The first number, 2^n, is the number of data inputs and the second number is always 1, which refers to the only output of a multiplexer. Each data input is assigned an address. The addresses range from 0 to 2^n - 1. For 2^n data inputs, it requires n control signals to determine which one of the data input is to be selected. The control signals are weighted so that a decimal equivalent can be determined from the control inputs. The address of the data input equal to the decimal equivalent of the binary control inputs is the data to be selected.

A 4-to-1 multiplexer is shown in Figure 7.14 and Table 7.7 as an example. The output Y can be expressed as follows:

$$Y = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3 \qquad (7.1)$$

If the control signals in the above equation is expressed in terms of minterm numbers,

$$Y = m_0 I_0 + m_1 I_1 + m_2 I_2 + m_3 I_3 = \sum_{i=0}^{3} m_i I_i$$

The above equation can be generalized to a 2^n-to-1 multiplexer.

$$Y = \sum_{i=0}^{2^n - 1} m_i I_i$$

This expression is in fact a sum-of-products. Each $m_i I_i$ is a product.

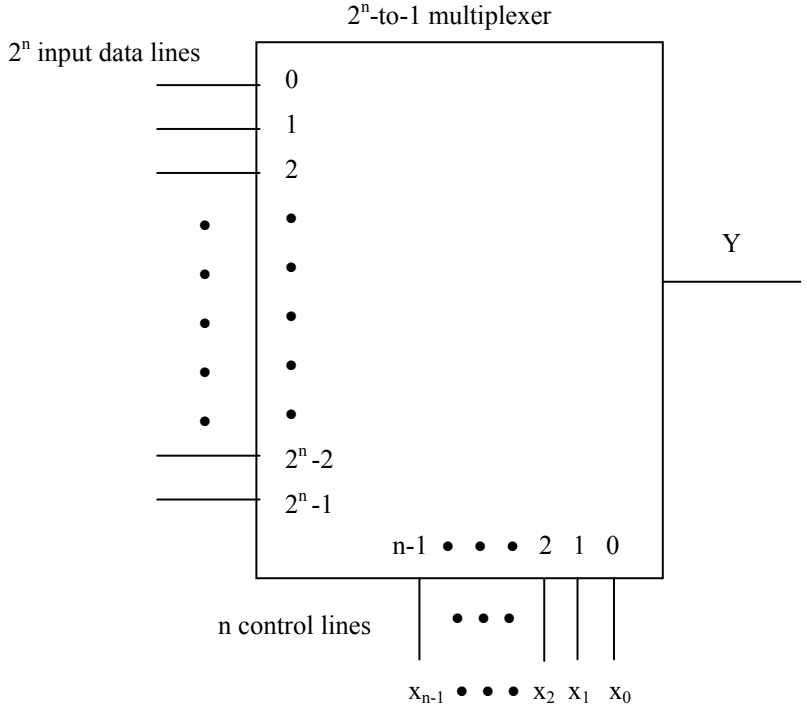


Figure 7.13 Logic diagram of a 2^n-to-1 multiplexer.

Table 7.7 Truth table for a 4-to-1 multiplexer.

A	B	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

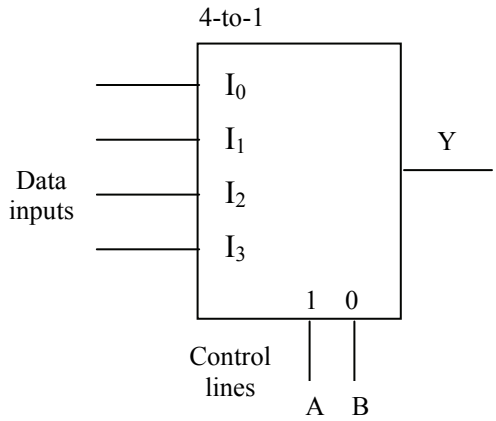


Figure 7.14 Logic diagram for a 4-to-1 multiplexer.

7.3.2 Design of Multiplexers

Although a multiplexer can be realized using the expression derived in the previous sub-section, a decoder is used as an example to show how its outputs are used to transmit or block a signal. The design of a 4-to-1 multiplexer is shown in Figure 7.15. The inputs to the 2-to-4 decoder are the address (control lines of the multiplexer) of the data input to be selected for transmission. This address is decoded into an output of 1. The other three outputs are equal to 0. When a decoder output of 0 is ANDed with a data input, the data input is blocked and prohibited from transmitting to the AND gate output. The blocking will produce a value of 0 at the AND gate output. When a data input is ANDed with a decoder output of 1, the data will be directed to the AND gate output. The decoder outputs allow only one data input to be transmitted and block the other three.

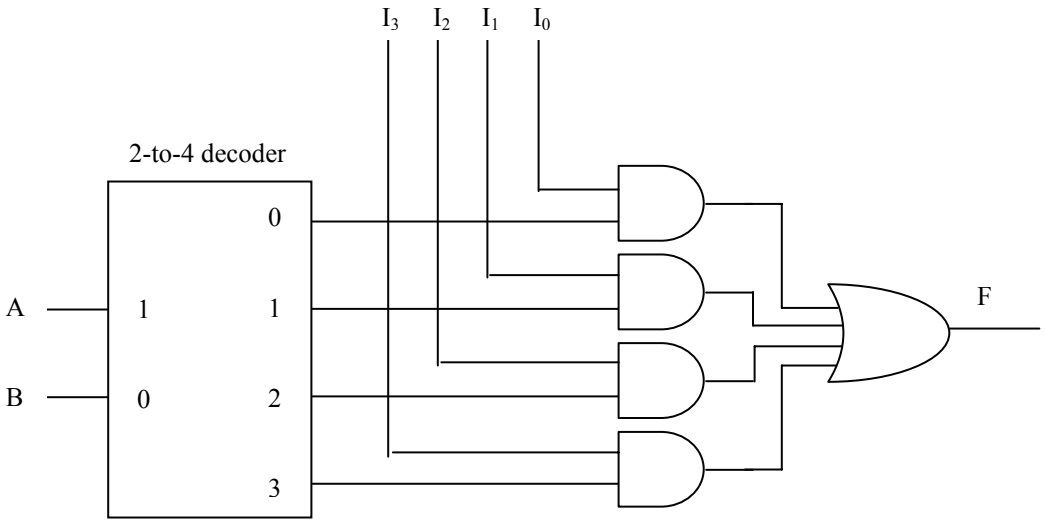
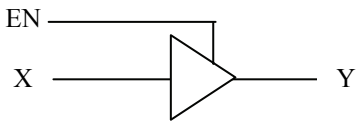


Figure 7.15 Design of a 4-to-1 multiplexer using a 2-to-4 decoder.

The 4-to-1 multiplexer is re-designed using tri-state buffers instead of AND gates and OR gates. A tri-state buffer is shown in Figure 7.16. The input is X and the out Y. EN is an active-high enable signal. When EN = 1, the buffer is enabled and Y = X. When EN = 0, the buffer is a high impedance (Z) and behaves as an open-circuit. The input and output are disconnected. The three states are 0, 1, and high Z.

Figure 7.16 Logic diagram of a tri-state buffer.



The design of a 4-to-1 multiplexer using a 2-to-4 decoder and four tri-state buffers is illustrated in Figure 7.17. The four decoder outputs are used to enable or disable the four tri-state buffers. The data to be transmitted are connected to the inputs of the buffers. Since three of the four decoder outputs are 0, three of the four tri-state buffers are disabled and disconnected from the multiplexer output Y. The only asserted decoder

output enables the remaining tri-state buffer. The data input to this buffer will then be transmitted to the output of the buffer, which is Y.

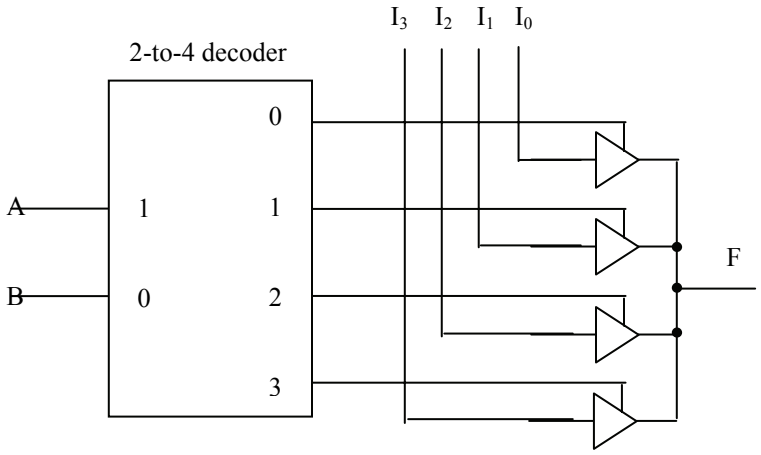


Figure 7.17 Design of a multiplexer using a decoder and four tri-state buffers.

7.3.3 Multiplexers with Enable Control

Similar to decoders, multiplexers may be enabled or disabled. When the enable signal is asserted, a multiplexer is activated. One of the data inputs is routed to the output. When the enable signal is de-asserted, a multiplexer is disabled. The output is de-asserted and has a value of 0 for active-high output, or a value of 1 for active-low output. The standard expression for a 2^n-to-1 multiplexers can be modified as follows. EN is the enable input.

$$Y = EN \left(\sum_{i=0}^{2^n - 1} m_i I_i \right)$$

For an active-low enable input /EN,

$$Y = (/EN)' \left(\sum_{i=0}^{2^n - 1} m_i I_i \right)$$

7.3.4 Expansion of Multiplexers

Similar to decoders, multiple smaller size multiplexers can also be used to realize a large size multiplexer when the latter is not available. Realization can be in tree configuration. Figure 7.18 shows the construction of a 16-to-1 multiplexer using five 4-

to-1 multiplexers. The control signals are $x_3x_2x_1x_0$. One data input from each of the four multiplexers in the first level is selected. These four data inputs are then screened by the multiplexer in the second level, which allows only one of the data inputs to be routed to the output.

Multiplexers can also be realized by taking advantage of the enable controls of multiplexers. Figure 7.19 shows the realization of an 8-to-1 multiplexer using two 4-to-1 multiplexers with enable controls. The control signals are $x_2x_1x_0$. Instead of selecting one data input from each multiplexer in the first level, one of the two multiplexers in Figure 7.19 is always disabled so that no data input can be selected. Only the selected data input from the enabled multiplexer can be routed to the output Y. Instead of a multiplexer, a 2-input OR gate can be used in the second level to select either one of the two outputs from the two 4-to-1 multiplexers. Because one of the two 4-to-1 multiplexer output is always de-asserted or 0, it is impossible for both inputs to the OR gate to be one. It is a don't-care situation. Therefore OR is equivalent to XOR.

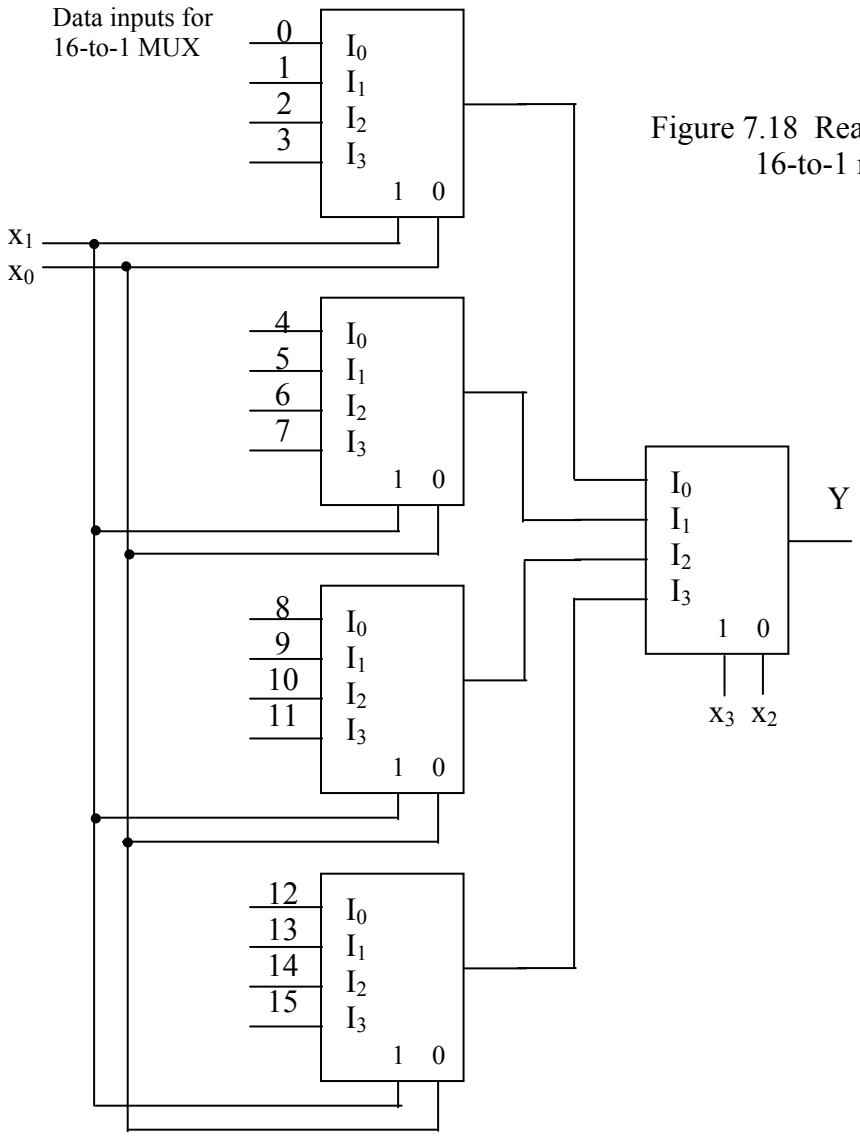


Figure 7.18 Realization of 16-to-1 multiplexer.

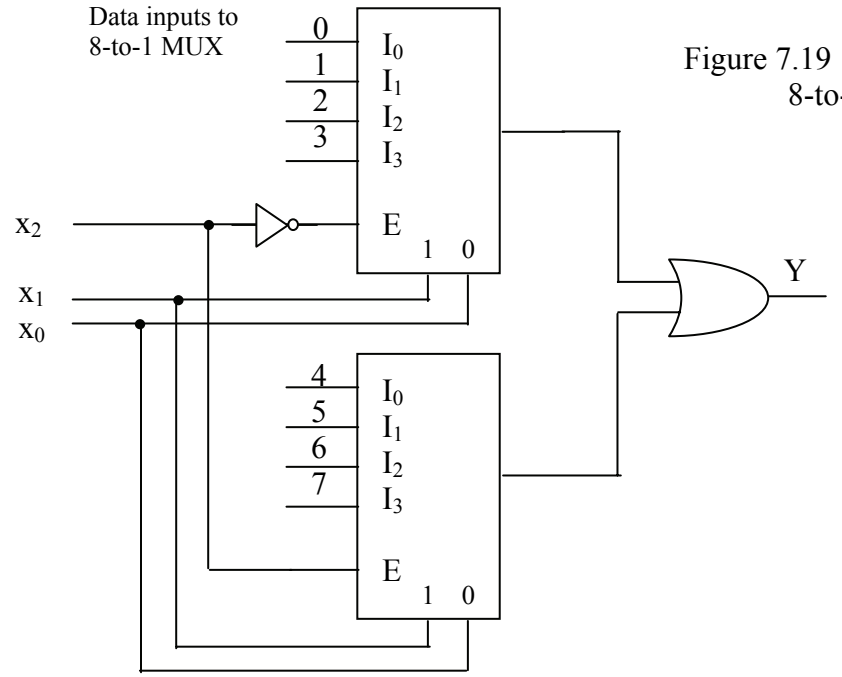


Figure 7.19 Realization of 8-to-1 multiplexer.

7.3.5 Implementation of Functions Using Multiplexers

It is shown in Section 7.3.1 that the general expression for a multiplexer is a sum-of-products expression. Therefore multiplexers can also be used to implement Boolean functions if variables and sub-functions are properly assigned to the data and control

Table 7.8 Combined table for F(A,B,C) and an 8-to-1 multiplexer.

A	B	C	Y	F(A,B,C)
0	0	0	I_0	0
0	0	1	I_1	0
0	1	0	I_2	1
0	1	1	I_3	1
1	0	0	I_4	1
1	0	1	I_5	0
1	1	0	I_6	0
1	1	1	I_7	1

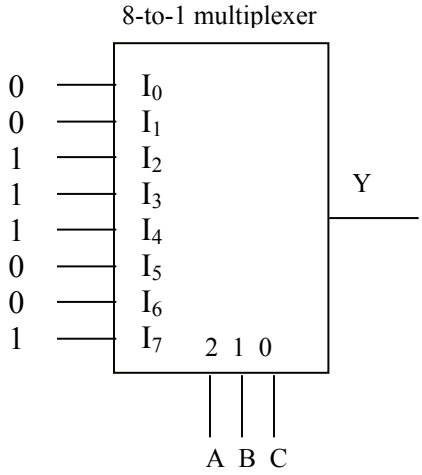


Figure 7.20 Implementation of a 3-variable function by an 8-to-1 multiplexer.

inputs. As an example, an 8-to-1 multiplexer is used to implement a 3-variable function F(A,B,C).

$$F(A,B,C) = \Sigma\, m(2, 3, 4, 7) = A'B + BC + AB'C'$$

The truth table for F and the function table of an 8-to-1 multiplexer are combined and shown as one table in Table 7.8. It is obvious that F can be implemented by placing the minterm coefficients at the data inputs and by connecting the variables to the control inputs, which is shown in Figure 7.20.

In general, an n-variable function can be implemented by a 2^n-to-1 multiplexer by assigning the variables to the control inputs and the minterm coefficients to the data inputs. The size of a multiplexer is doubled when the number of variables is incremented by one.

A 2^n-to-1 multiplexer can also be used to realize a function of more than n variables. As an example, a 4-to-1 multiplexer is used to realize the function in Table 7.8 Since there are only two control inputs for a 4-to-1 multiplexer, only two of the three variables are selected and connected to the control inputs. A and B are selected as shown in Figure 7.21. The standard form of the multiplexer output is

$$Y = A'B'(I_0) + A'B(I_1) + AB'(I_2) + AB(I_3) \qquad (7.1)$$

If F is expanded with A and B, the expression is

$$F = A'B'F_{AB=00} + A'B\,F_{AB=01} + AB'\,F_{AB=10} + AB\,F_{AB=11} \qquad (7.2)$$

It is obvious Equations (7.1) and (7.2) are equal if

$$I_0 = F_{AB=00}, \quad I_1 = F_{AB=01}, \quad I_2 = F_{AB=10}, \quad I_3 = F_{AB=11}$$

From the sum-of-products expression for f, the data inputs or the sub-functions can be ready determined. The implementation is shown in Figure 7.21.

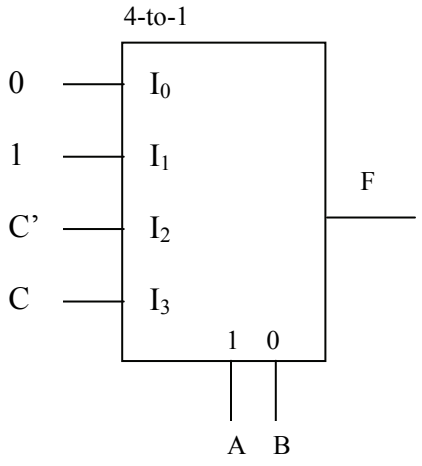


Figure 7.21 Realization of a 3-variable function by a 4-to-1 multiplexer.

$I_0 = F_{AB=00} = 0$

$I_1 = F_{AB=01} = 1$

$I_2 = F_{AB=10} = C'$

$I_3 = F_{AB=11} = C$

To implement an n-variable function using a 2^m-to-1 multiplexer, m variables are first selected as control signals of the multiplexers. Then find all the 2^m sub-functions with the control signals as expansion variables. They are the data inputs of the multiplexer.

❖ Example 7.1

A 4-variable function is implemented by an 8-to-1 multiplexer in this example. The minterm list form of the function is

$$F(A,B,C,D) = \Sigma\, m(0, 2, 3, 5, 7, 10, 14, 15)$$

Figure 7.22 shows all the eight sub-functions of F with the arbitrary selection of A, C, and D as control signals. A is the most significant and D the least significant.

$I_0 = B'$ $I_1 = B$ $I_2 = B'$ $I_3 = 1$

$I_4 = 0$ $I_5 = 0$ $I_6 = 1$ $I_7 = B$

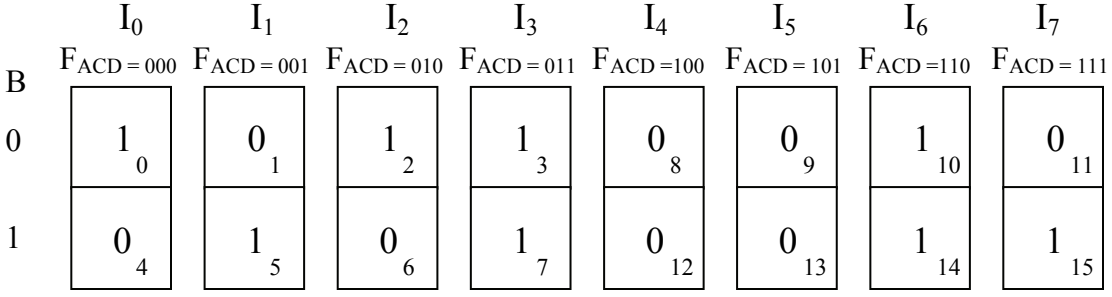


Figure 7.22 Sub-function K-maps as data inputs in the implementation of a 4-variable function by an 8-to-1 multiplexer.

❖ Example 7.2

The implementation of a 4-variable function F using a 4-to-1 multiplexer is given in this example.

$$F(A, B, C, D) = \Sigma\, m(0, 1, 3, 5, 8, 11, 14, 15) + d(9, 10, 13)$$

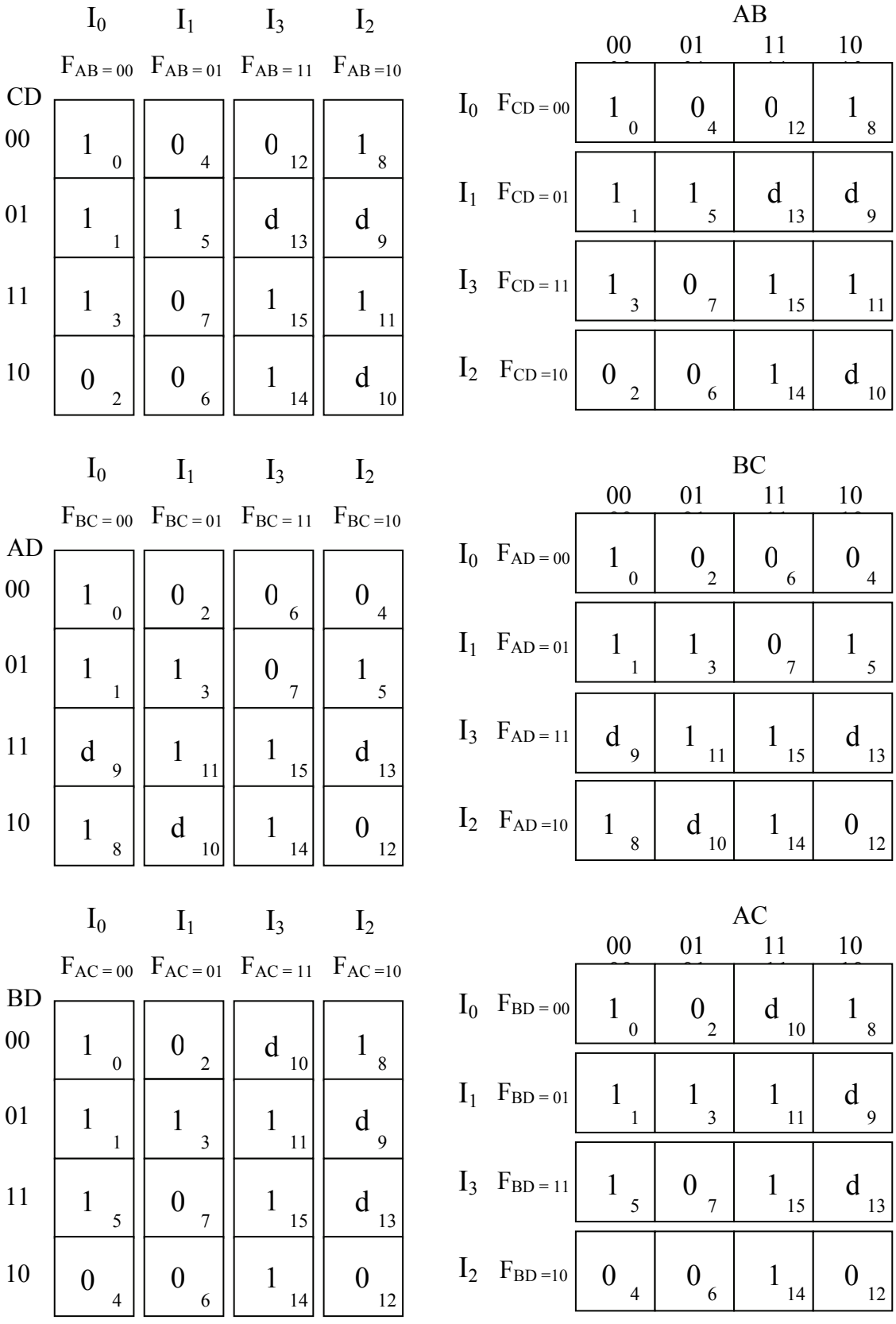


Figure 7.23 Sub-function K-maps as data inputs in the implementation of a 4-variable function by a 4-to-1 multiplexer.

Since the function has four variables and the multiplexers has only two control inputs, only two variables can be selected as control inputs. To select two variables out of four, there are ${}_4C_2 = 4! / [2! (4-2)!] = 6$ different combinations. They are AB, AC, AD, BC, BD, and CD. A good selection may translate into simpler expressions, which in turn require fewer external gates. However, it is unable to foresee which one of the six combinations is the best selection.

Figure 7.23 shows the partition for all the six different selections of control inputs. The simplest expressions for the data inputs are listed below. They are listed in the order according to the six sets of sub-function K-maps from left to right and top to bottom in Figure 7.23. It is seen that by selecting B and C as the control inputs, the expressions for the data inputs are the simplest. It does not require any external gates.

Control inputs A, B	$I_0 = C' + D$	$I_1 = C'D$
	$I_2 = 1$	$I_3 = C$
Control inputs C, D	$I_0 = B'$	$I_1 = 1$
	$I_2 = A$	$I_3 = A + B'$
Control inputs B, C	$I_0 = 1$	$I_1 = D$
	$I_2 = D$	$I_3 = A$
Control inputs A, D	$I_0 = B'C'$	$I_1 = B' + C'$
	$I_2 = B' + C$	$I_3 = 1$
Control inputs A, C	$I_0 = B' + D$	$I_1 = B'D$
	$I_2 = B'$	$I_3 = 1$
Control inputs B, D	$I_0 = C'$	$I_1 = 1$
	$I_2 = AC$	$I_3 = A + C'$

7.4 De-multiplexer

A de-multiplexer performs the reverse function of a multiplexer. There is only one data input. This input is directed to one of 2^n outputs. The n control inputs will select one of the outputs to which the data input is directed. Figure 7.24 is a de-multiplexer with a data input I and four outputs Y_0, Y_1, Y_2, Y_3. The data input I is directed to one of the four outputs. The output to which I is directed is determined by an address generated by the 2-to-4 decoder.

The Boolean expressions for the outputs are

$Y_0 = I (x_1' x_0')$

$Y_1 = I (x_1' x_0)$

$Y_2 = I (x_1 x_0')$

$Y_3 = I (x_1 x_0)$

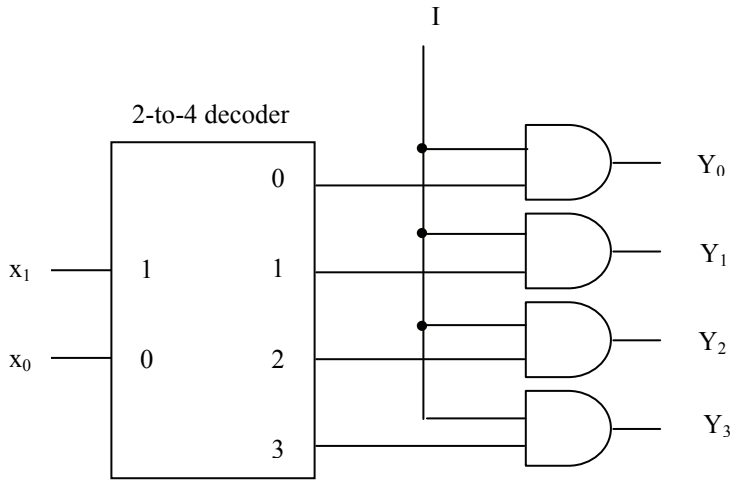


Figure 7.24 Example of a de-multiplexer.

In Section 7.1.3, it is shown that when a decoder is controlled by a strobe or enable signal EN, the outputs of a 2-to-4 decoder are

$D_0 = EN (x_1' x_0')$

$D_1 = EN (x_1' x_0)$

$D_2 = EN (x_1 x_0')$

$D_3 = EN (x_1 x_0)$

By comparing the output expressions for the de-multiplexer and the decoder, it is obvious that, with the enable input used as a data input, a decoder can function as a de-multiplexer.

In order to provide the feature of enabling and disabling a de-multiplexer, the de-multiplexer in Figure 7.24 can be modified by including an enable input. A de-multiplexer with an enable input is given in Figure 7.25.

$Y_0 = I \bullet EN \bullet (x_1' x_0')$

$Y_1 = I \bullet EN \bullet (x_1' x_0)$

$Y_2 = I \bullet EN \bullet (x_1 x_0')$

$Y_3 = I \bullet EN \bullet (x_1 x_0)$

When the circuit in Figure 7.25 is used as a de-multiplexer, asserting or de-asserting EN will enable or disable the de-multiplexer. If it is used as a decoder, the data input I becomes another strobe. To enable the circuit as a decoder, both I and EN must be asserted. De-asserting either I or EN, or both of them, will disable the decoder.

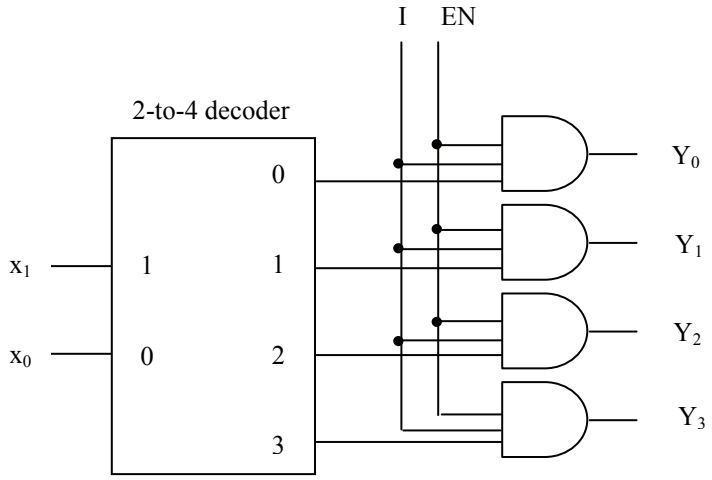


Figure 7.25 A de-multiplexer with enable input.

PROBLEMS

1. Design a 4-to-16 decoder using only 3-to-8 decoders. Assume that each 3-to-8 decoder has an active-low enable input /E.

2. Design a 3-to-8 decoder using only two 2-to-4 decoders/de-multiplexer with an active-low enable control /E and an active-high data input I.
3. Design a BCD-to-decimal decoder. The inputs to the decoder are a 4-bit BCD code $b_3b_2b_1b_0$. The decoder outputs are active-low ($/d_0$, $/d_1$, $/d_2$, $/d_3$, $/d_4$, $/d_5$, $/d_6$, $/d_7$, $/d_8$, $/d_9$). Minimize the design.

4. Find the minterm list for the function F(A,B,C) realized by the circuit in Figure P7.1.

5. Implement each of the following functions using a 3-to-8 decoder and a minimum number of 4-input NOR gates. Assume that the decoder outputs are active-high.

   (a) f(A,B,C) = AB + AC' + BC
   (b) f(A,B,C) = C $\oplus$ A'B

6. Implement f(A,B,C,D) = Σ m (0,2,3,4,5,6,8,9,10,12,14) using a 3-to-8 decoder and a minimum number of 4-input NAND gates. The decoder outputs are active-low.

7. Implement the following functions using a 3-to-8 decoder with active-low outputs and a minimum number of 4-input gates.

$F_1(A,B,C) = \Sigma\, m(1, 2, 4, 5)$
$F_2(A,B,C) = \Sigma\, m(0, 2, 3, 5, 6)$

8. Design a 4-to-2 priority encoder using only NOR gates. The priorities of the inputs are in the order of a_3, a_2, a_1, a_0. Their corresponding outputs are y_1y_0 =11, 10, 01, 00. The encoder also has an active-low output /Idle which is asserted when none of the (active-high) inputs is asserted.

9. Find the simplest sum-of-products for the output F in Figure P7.2.

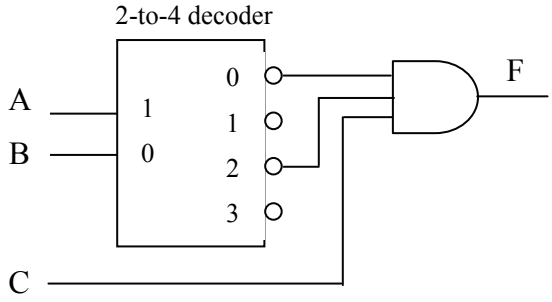


Figure P7.1

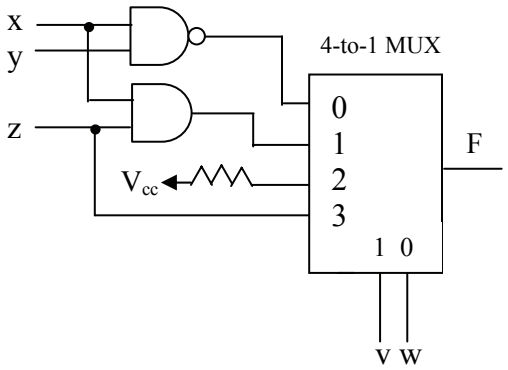
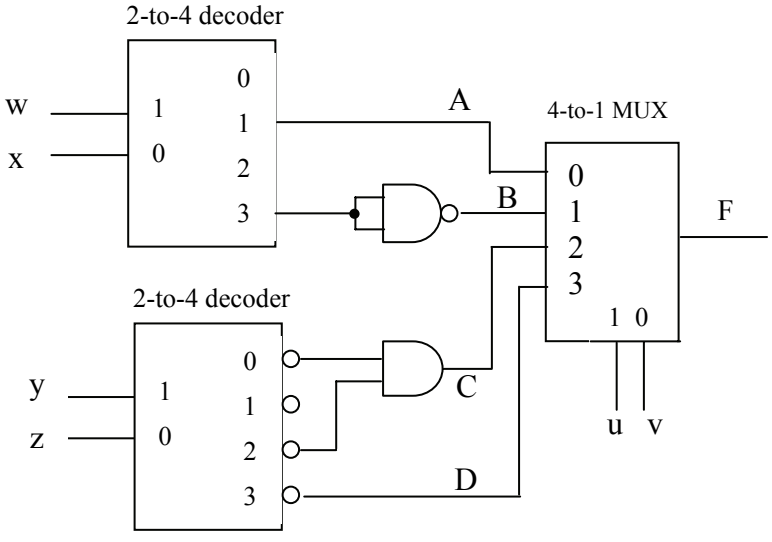


Figure P7.2

10. Determine A, B, C, D, and F as functions of u, v, w, x, y, z for the circuit in Figure P7.3.

Figure P7.3



11. Design an 8-to-1 multiplexer using only 2-to-1 multiplexers without enable inputs.

12. Find the simplest sum-of-products expression for the output of a 4-to-1 multiplexer if the control inputs to the multiplexer are A (most significant control bit) and D, and the four data inputs are

$$I_0 = C + B'E$$
$$I_1 = C$$
$$I_2 = B'C + C'E$$
$$I_3 = BC + C'E$$

13. Implement the following function using a 4-to-1 multiplexer and a minimum number of AND gates and OR gates. All variables are double-rail and the control signals of the multiplexer are B and D.

$$f(A,B,C,D,E) = \Sigma\, m\,(3,4,5,6,7,11,15,16,17,20,21,30,31)$$

14. Implement the following function using a 4-to-1 multiplexer and a minimum number of AND gates and OR gates. All variables are double-rail and the control signals of the multiplexer are D and E.

$$f(A,B,C,D,E) = \Sigma\, m\,(0,4,5,8,12,13,14,19,23,25,26,29,30) + d(2,3,6,7,16,18,20,22)$$