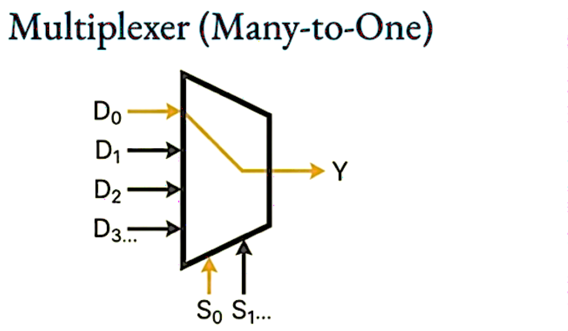
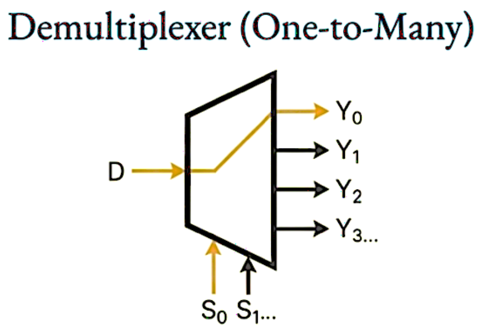
Multiplexing and Demultiplexing

Mastering the principles of multiplexing and demultiplexing is fundamental to digital design. This pair of components provides the essential capability to select, route, and distribute data, forming the backbone of communication and control within digital systems.



Function: Selects one of many inputs to route to a single output.
Key Application: Consolidating data lines, logic function generation.

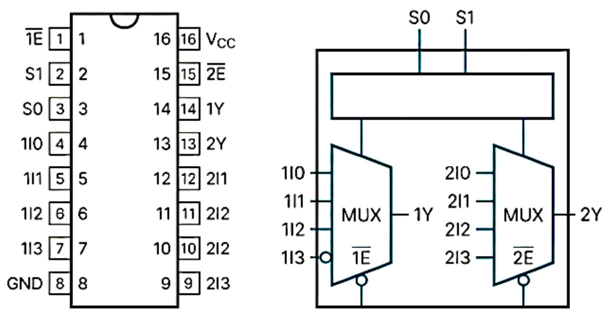


Function: Distributes a single input to one of many selected outputs.
Key Application: Directing data to specific destinations, decoder functionality.

Multiplexers are available as standard integrated circuits (Is). Here are two common examples from the 74HC family of logic chips.

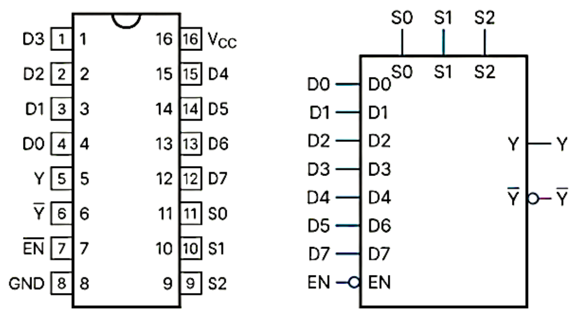
74HC153 - Dual 4-Input MUX

Contains two independent 4-input multiplexers in a single package. It features common select lines (S0, S1) and separate active-LOW enable inputs ($\overline{1E}$, $\overline{2E}$) for each MUX.



74HC151 - 8-Input MUX

A single 8-input multiplexer. It requires three select lines (S0, S1, S2) to choose from one of the eight inputs (2^3=8). Provides both the output (Y) and its complement ($\overline{Y}$) and has an active-LOW enable input (EN).



❑ Multiplexers (Data Selectors)

A **multiplexer (MUX)** is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination. The basic multiplexer has several data-input lines and a single output line. It also has data-select inputs, which permit digital data on any one of the inputs to be switched to the output line. Multiplexers are also known as data selectors. A logic symbol for a 4-input multiplexer (MUX) is shown in Figure 1. Notice that there are two data-select lines because with two select bits, any one of the four data-input lines can be selected.

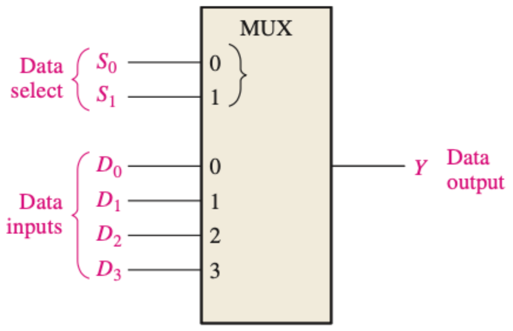


Figure 1 Logic symbol for a 1-of-4 data selector/multiplexer.

In Figure 1, a 2-bit code on the data-select (S) inputs will allow the data on the selected data input to pass through to the data output. If a binary 0 ($S_1 = 0$ and $S_0 = 0$) is applied to the data-select lines, the data on input D_0 appear on the data-output line. If a binary 1 ($S_1 = 0$ and $S_0 = 1$) is applied to the data-select lines, the data on input D_1 appear on the data output. If a binary 2 ($S_1 = 1$ and $S_0 = 0$) is applied, the data on D_2 appear on the output. If a binary 3 ($S_1 = 1$and$S_0 = 1$) is applied, the data on D_3 are switched to the output line. A summary of this operation is given in Table 1.

Table 1 Data Selection for a 1 of 4 multiplexer

Data-Select Inputs S_1	S_0	Input Selected
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Email (noorabdalkareem@uomus.edu.iq)

Now let's look at the logic circuitry required to perform this multiplexing operation. The data output is equal to the state of the *selected* data input. You can therefore, derive a logic expression for the output in terms of the data input and the select inputs.

The data output is equal to D_0 only if $S_1 = 0$ and $S_0 = 0$: $Y = D_0\bar{S}_1\bar{S}_0$.

The data output is equal to D_1 only if $S_1 = 0$ and $S_0 = 1$: $Y = D_1\bar{S}_1S_0$.

The data output is equal to D_2 only if $S_1 = 1$ and $S_0 = 0$: $Y = D_2S_1\bar{S}_0$.

The data output is equal to D_3 only if $S_1 = 1$ and $S_0 = 1$: $Y = D_3S_1S_0$.

When these terms are ORed, the total expression for the data output is

$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$$

The implementation of this equation requires four 3-input AND gates, a 4-input OR gate, and two inverters to generate the complements of S_1 and S_0, as shown in Figure 6–44. Because data can be selected from any one of the input lines, this circuit is also referred to as a **data selector**.

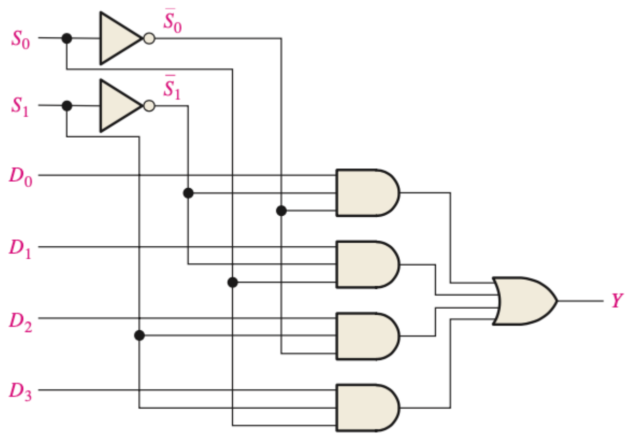


Figure 2 Logic diagram for a 4-input multiplexer.

Example 1: The data-input and data-select waveforms in Figure 3 (a) are applied to the multiplexer in Figure 2. Determine the output waveform in relation to the inputs.

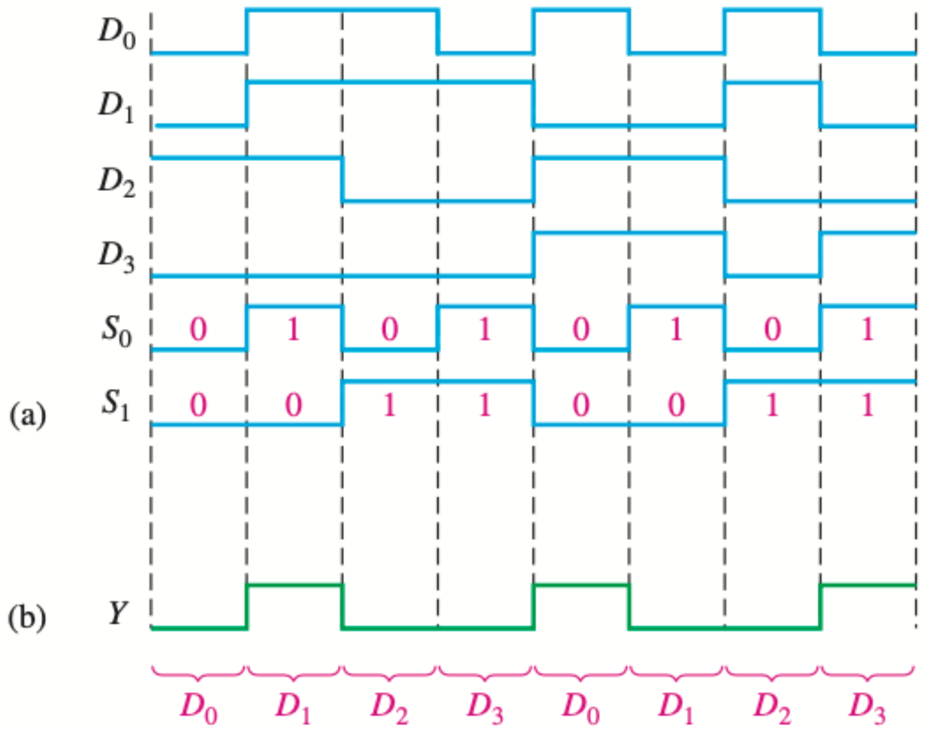


Figure 3 Data-input and data-select waveforms

Solution

The binary state of the data-select inputs during each interval determines which data input is selected. Notice that the data-select inputs go through a repetitive binary sequence 00, 01, 10, 11, 00, 01, 10, 11, and so on. The resulting output waveform is shown in Figure 3 (b).

Example 2: Use 74HC151s and any other logic necessary to multiplex 16 data lines onto a single data-output line.

Solution

An expansion of two 74HC151s is shown in Figure 4. Four bits are required to select one of 16 data inputs ($2^4 = 16$). In this application the *Enable* input is used as the most significant data-select bit. When the MSB in the data-select code is LOW, the left 74HC151 is enabled, and one of the data inputs (D_0 through D_7) is selected by the other three data- select bits. When the data-select MSB is HIGH, the right 74HC151 is enabled, and one of the data inputs (D_8 through D_{15}) is selected. The selected input data are then passed through to the negative-OR gate and onto the single output line.

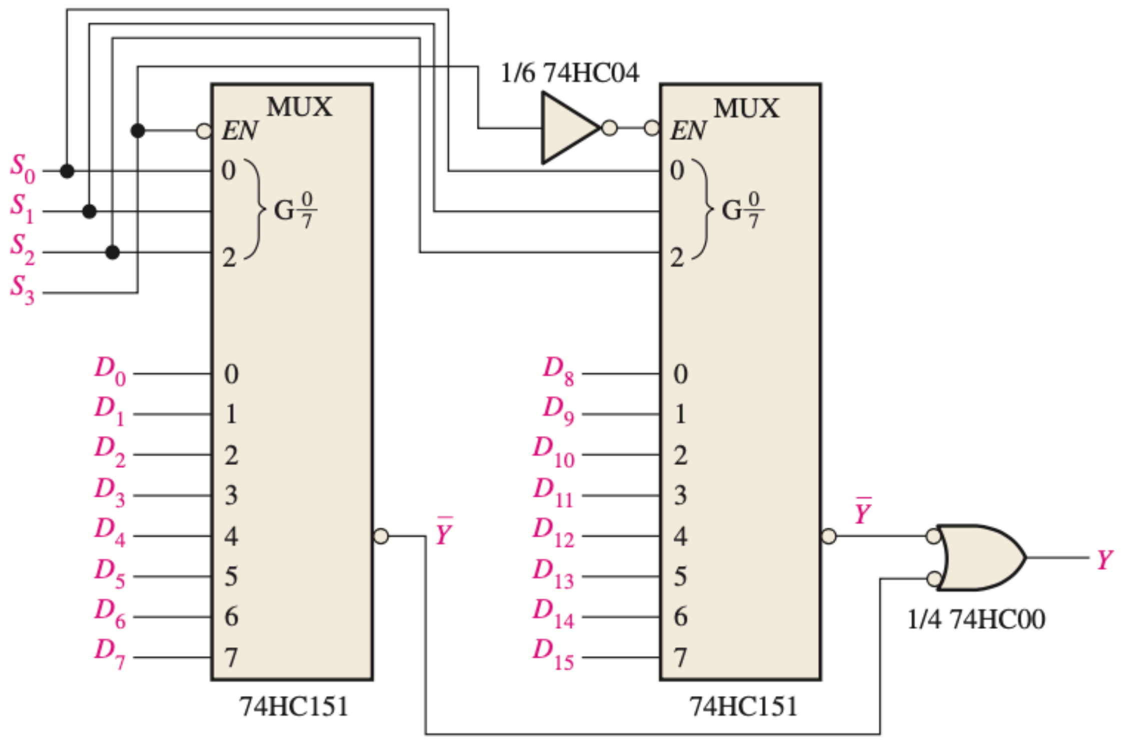


Figure 4 16-input multiplexer.

Applications:

1- A 7-Segment Display Multiplexer

Figure 5 shows a simplified method of multiplexing BCD numbers to a 7-segment display. In this example, 2-digit numbers are displayed on the 7-segment readout by the use of a single BCD-to-7-segment decoder. This basic method of display multiplexing can be extended to displays with any number of digits. The 74HC157 is a quad 2-input multiplexer.

The basic operation is as follows. Two BCD digits ($A_3A_2A_1A_0$ and $B_3B_2B_1B_0$) are applied to the multiplexer inputs. A square wave is applied to the data-select line, and when it is LOW, the *A* bits ($A_3A_2A_1A_0$) are passed through to the inputs of the 74HC47 BCD-to-7- segment decoder. The LOW on the data-select also puts a LOW on the A_1 input of the 74HC139 2-line-to-4-line decoder, thus activating its 0 output and enabling the *A*-digit display by effectively connecting its common terminal to ground. The *A* digit is now *on* and the *B* digit is *off.*

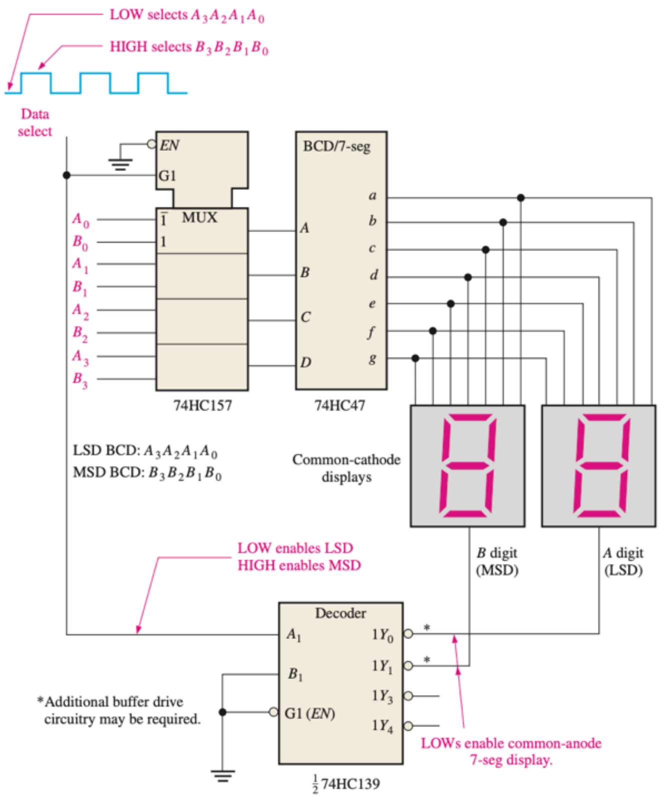


Figure 5 Simplified 7-segment display multiplexing logic

When the data-select line goes HIGH, the *B* bits ($B_3B_2B_1B_0$) are passed through to the inputs of the BCD-to-7-segment decoder. Also, the 74HC139 decoder's 1 output is activated, thus enabling the *B*-digit display. The *B* digit is now *on* and the *A* digit is *off.* The cycle repeats at the frequency of the data-select square wave. This frequency must be high enough to prevent visual flicker as the digit displays are multiplexed.

2- **A Logic Function Generator**

A useful application of the data selector/multiplexer is in the generation of combinational logic functions in sum-of-products form. When used in this way, the device can replace discrete gates, can often greatly reduce the number of ICs, and can make design changes much easier.

To illustrate, a 74HC151 8-input data selector/multiplexer can be used to implement any specified 3-variable logic function if the variables are connected to the data-select inputs and each data input is set to the logic level required in the truth table for that function. For example, if the function is a 1 when the variable combination is $\overline{A_2}A_1\overline{A_0}$, the 2 input (selected by 010) is connected to a HIGH. This HIGH is passed through to the output when this particular combination of variables occurs on the data-select lines. Example 6–16 will help clarify this application.

Example 3 Implement the logic function specified in Table below by using a 74HC151 8-input data selector/multiplexer. Compare this method with a discreet logic gate implementation.

Inputs			Output
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Solution

Notice from the truth table that *Y* is a 1 for the following input variable combinations: 001, 011, 101, and 110. For all other combinations, *Y* is 0. For this function to be implemented with the data selector, the data input selected by each of the above-mentioned combinations must be connected to a HIGH (5 V). All the other data inputs must be connected to a LOW (ground), as shown in Figure 6.

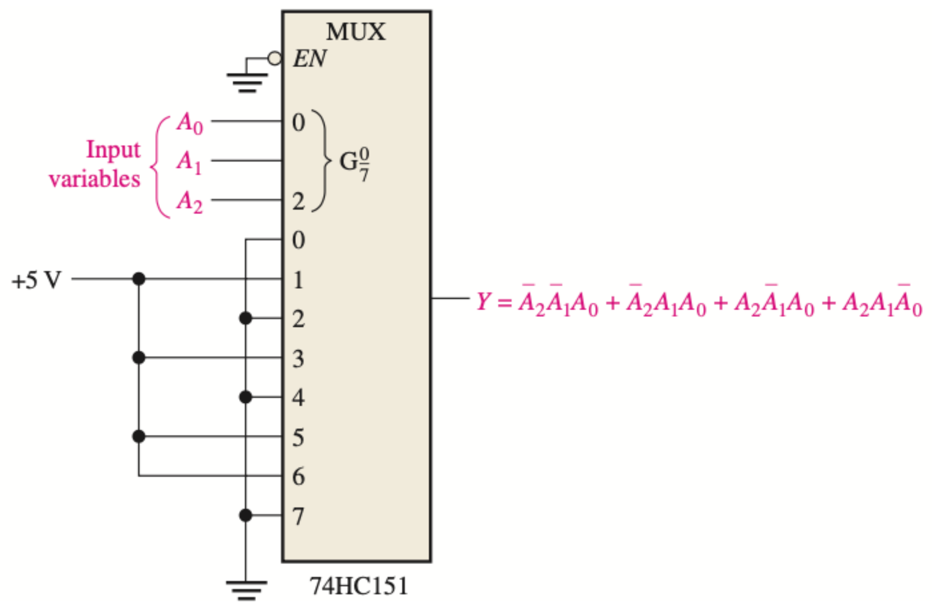

MUX
EN
Input variables
A_0
A_1
A_2
0
2
$G\frac{0}{7}$
+5 V
0
1
2
3
4
5
6
7
$Y = \bar{A}_2\bar{A}_1A_0 + \bar{A}_2A_1A_0 + A_2\bar{A}_1A_0 + A_2A_1\bar{A}_0$
74HC151

Figure 6 Data selector/multiplexer connected as a 3-variable logic function generator.

The implementation of this function with logic gates would require four 3-input AND gates, one 4-input OR gate, and three inverters unless the expression can be simplified.

❑ Demultiplexers (Data Distributor)

A **demultiplexer (DEMUX)** basically reverses the multiplexing function. It takes digital information from one line and distributes it to a given number of output lines. For this rea- son, the demultiplexer is also known as a data distributor. As you will learn, decoders can also be used as demultiplexers. Figure 7 shows a 1-line-to-4-line demultiplexer (DEMUX) circuit. The data-input line goes to all of the AND gates. The two data-select lines enable only one gate at a time, and the data appearing on the data-input line will pass through the selected gate to the associated data-output line.

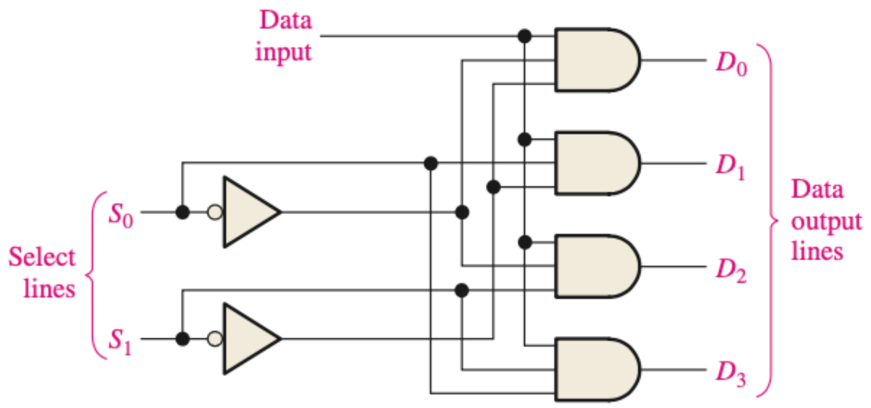


Figure 7 A 1-line-to-4-line demultiplexer.

Example 4: The serial data-input waveform (Data in) and data-select inputs (S_0 and S_1) are shown in Figure 8. Determine the data-output waveforms on D_0 through D_3 for the demultiplexer in Figure below.

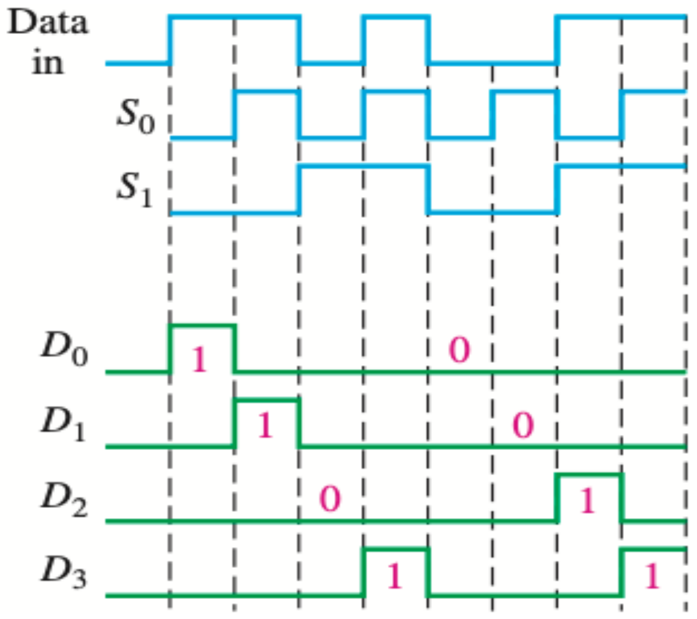


Figure 8 Waveforms for input & output

Solution: Notice that the select lines go through a binary sequence so that each successive input bit is routed to D_0, D_1, D_2, and D_3 in sequence, as shown by the output waveforms in Figure 8.