Table 2.2, given that A, B, and C are the inputs, and an LED is the active-low output (assume that the LED is turned on by driving a logic 0 rather than a logic 1).

TABLE 2.2 LED Driver Logic Truth Table

A	B	C	$\overline{\text{LED}}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

This LED driver truth table can be converted into the following Boolean logic equation with a Karnaugh map or simply by inspection:

$$\overline{\text{LED}} = \overline{\bar{A}BC + A\bar{B}C + AB\bar{C}}$$

After consulting a list of available 7400 logic ICs, three become attractive for our application: the 7404 inverter, 7408 AND, and 7432 OR. The LED driver logic equation requires four inverters, six two-input AND gates, and two 2-input OR gates. Four ICs are required, because a 7404 provides six inverters, a 7408 provides four AND gates, and a 7432 contains four OR gates. These four ICs can be connected according to a *schematic diagram* as shown in Fig. 2.12. A schematic diagram illustrates the electrical connectivity scheme of various components. Each component is identified by a *reference designator* consisting of a letter followed by a number. ICs are commonly identified by reference designators beginning with the letter "U". Additionally, each component has numerous pins that are numbered on the diagram. These pin numbers conform to the IC manufacturer's numbering scheme. Each of these 7400-series ICs has 14 pins. Another convention that remains from bipolar logic days is the use of the label VCC to indicate the positive voltage supply node. GND represents ground—the common, or return, voltage supply node.

All ICs require connections to a power source. In this circuit, +5 V serves as the power supply, because the 7400 family is commonly manufactured in a bipolar semiconductor process requiring a +5-V supply. The four rectangular blocks at the top of the diagram represent this power connection information. Because this schematic diagram shows individual gates, the gates' reference designators contain an alphabetic suffix to identify unique instances of gates within the same IC. Not all gates in each IC are actually used. Those that are unused are tied inactive by connecting their inputs to a valid logic level—in this case, ground. It would be equally valid to connect the inputs of unused gates to the positive supply voltage, +5 V.

This logic circuit would work, but a more efficient solution is available to those who are familiar with the capabilities of the 7400 family. The 7411 provides three 3-input AND gates, which is perfect for this application, allowing a reduction in the part count to three ICs instead of four. This cir-

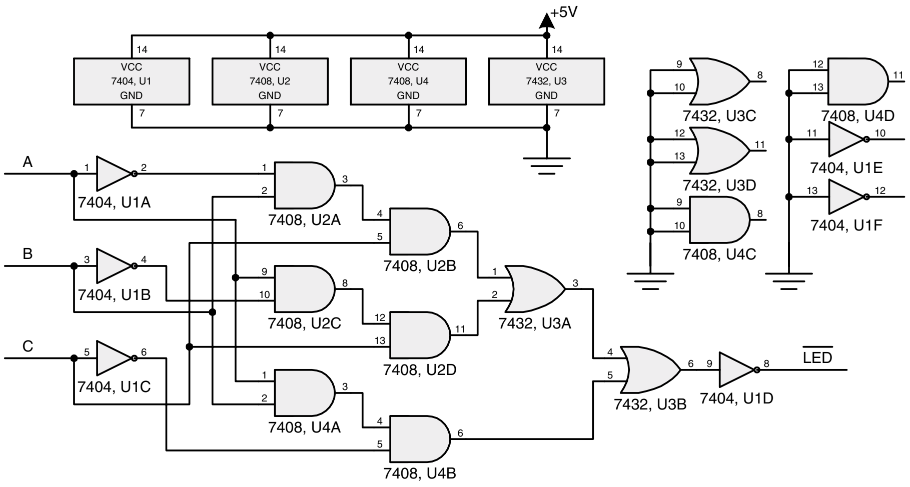

FIGURE 2.12 LED driver logic implementation.

cuit is shown in Fig. 2.13 with alternative notation to illustrate varying circuit presentation styles. Rather than drawing gates as separate elements, the complete 7400-series ICs are shown as monolithic blocks. Either notation is commonly accepted and depends on the engineer's preference.

2.5 SYNCHRONOUS LOGIC DESIGN WITH THE 7400 FAMILY

The preceding LED driver example shows how state-less logic (logic without flops and a clock) can be designed to implement an arbitrary logic equation. State-full logic is almost always required in a digital system, because it is necessary to advance one step at a time (one step each cycle) through an algorithm. Some 7400 ICs, such as counters, implement synchronous logic within the IC itself by combining Boolean logic gates and flops on the same die. Other 7400 ICs implement only flops that may be combined externally with logic to create the desired function.

An example of a synchronous logic application is a basic serial communications controller. Serial communications is the process of taking parallel data, perhaps a byte of information, and transmitting or receiving that byte at a rate of one bit per clock cycle. The obvious downside of doing this is that it will take longer to transfer the byte, because it would be faster to just send the entire byte during the same cycle. The advantage of serial communications is a reduction in the number of wires required to transfer information. Being able to string only a few wires between buildings instead of dozens usually compensates for the added serial transfer time. If the time required to serially transfer bits is too slow, the rate at which the bits are sent can be increased with some engineering work to achieve the desired throughput. Such speed improvements are beyond the scope of this presentation.

Real serial communications devices can get fairly complicated. For purposes of discussion, a fairly simplistic approach is taken. Once the decision is made to serialize a data byte, the problem arises of knowing when that byte begins and ends. *Framing* is the process of placing special patterns into the data stream to indicate the start and end of data units. Without some means to frame the individual bits as they are transmitted, the receiver would have no means of finding the first and last bits of each byte. In this example, a single *start bit* is used to mark the first bit. Once the first bit is