The diagram in Figure 2.5 uses a common convention: two lines that cross do not indicate an electrical connection unless a solid dot appears. The idea is similar to the way vertices and edges are drawn in a graph: two edges that cross do not indicate a vertex is present unless a dot (or circle) is drawn. In a circuit diagram, two lines that cross without a dot correspond to a situation in which there is no physical connection; we can imagine that the wires are positioned so an air gap exists between them (i.e., the wires do not touch). To help indicate that there is no connection, the lines are drawn with a slight space around the crossing point.

Now that we have seen an example of how a gate can be created out of transistors, we do not need to consider individual transistors again. Throughout the rest of the chapter, we will discuss gates without referring to their internal mechanisms. Later chapters discuss larger, more complex mechanisms that are composed of gates.

2.7 Symbols Used For Logic Gates

When they design circuits, engineers think about interconnecting logic gates rather than interconnecting transistors. Each gate is represented by a symbol, and engineers draw diagrams that show the interconnections among gates. Figure 2.6 shows the symbols for *nand*, *nor*, *inverter*, *and*, *or*, and *xor* gates. The figure follows standard terminology by using the term *inverter* for a gate that performs the Boolean *not* operation.

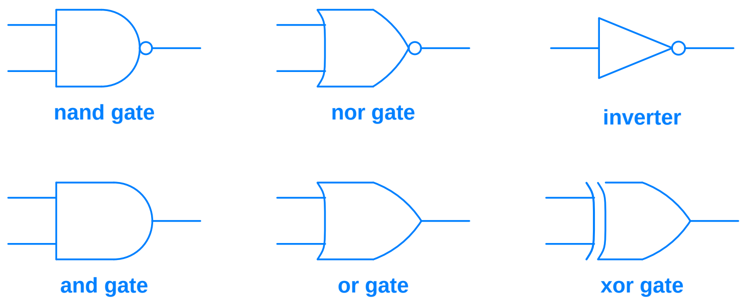

Figure 2.6 The symbols for commonly used gates. Inputs for each gate are shown on the left, and the output of the gate is shown on the right.

2.8 Example Interconnection Of Gates

The electronic parts that implement gates are classified as *Transistor-Transistor Logic* (*TTL*) because the output transistors in each gate are designed to connect directly to input transistors in other gates. In fact, an output can connect to several inputs†. For example, suppose a circuit is needed in which the output is true if a disk is spinning and the user presses a power-down button. Logically, the output is a Boolean *and* of two

†The technology limits the number of inputs that can be connected to a single output; we use the term *fanout* to specify the number of inputs that an output supplies.