machine can look ahead one cycle into the future and return to idle in time for a new start bit that may be arriving. With the logical details of the state machine now complete, the state machine can be represented with the *state transition diagram* in Fig. 2.17.

A state transition diagram, often called a *bubble diagram*, shows all the states of an FSM and the logical *arcs* that dictate how one state leads to another. When implemented, the arcs are translated into the state logic to make the FSM function. With a clearly defined state transition diagram, the logic to drive the state machine can be organized as shown in Table 2.3.

TABLE 2.3 Serial Receive State Machine Logic Truth Table

Current State	din	ready_next	Next State
1	0	X	1
1	1	X	0
0	X	0	0
0	X	1	1

When in the idle state (1), a high on *din* (the start bit) must be observed to transition to the receiving state (0). Once in the receiving state, *ready_next* must be high to return to idle. This logic is represented by the Boolean equation,

$$\text{Next} = (\text{State}\&\overline{\text{Din}}) + (\overline{\text{State}}\&\text{ready\_next})$$

As with most problems, there exists more than one solution. Depending on the components available, one may choose to design the logic differently to make more efficient use of those components. As a general rule, it is desirable to limit the number of ICs used. The 7451 provides two "AND-OR-INVERT" gates, each of which implements the Boolean function,

$$Y = \overline{AB + CD}$$

This function is tantalizingly close to what is required for the state machine. It differs in that the inversion of two inputs (*state* and *din*) and a NOR function rather than an OR are necessary. Both differences can be resolved using a 7404 inverter IC, but there is a more efficient solution using the 74175 quad flop. The 74175's four flops each provide both true and inverted outputs. Therefore, a separate 7404 is not necessary. An inverted version of *din* can be obtained by passing *din* through a flip-flop before feeding the remainder of the circuit's logic. For purposes of notation, we will refer to this "flopped" *din* as *din´*. Another flop will be used for the state machine. The inverted output of the state flop will compensate for the NOR vs. OR function of the 7451. A third flop will form the ninth bit of the *ready* delay shift register when combined with a 74164 eight-bit parallel-out shift register.

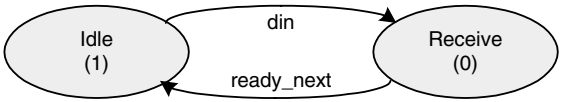

FIGURE 2.17 Serial receive state machine.

Conveniently, the 74164 contains an internal AND gate at its input to implement the idle-enable of the start bit into the shift register.

The total parts count for this serial receiver is four 7400-family ICs: two 74164 shift registers, one 7451 AND-OR-INVERT, and one 74175 quad flop. One flop and one-half of the 7451 are unused in this application. Figure 2.18 shows how these ICs are connected to implement the serial receive logic. Note that a mixed-style of IC representation is used: most ICs are shown in a single block, but the 74175 is broken into separate flops for clarity. Even if an IC is represented as a single block, it is not necessary to draw the individual pins in the order in which they physically appear. As with the previous example, the graphical representation of logic depends on individual discretion. In addition to being functionally and electrically correct, a schematic diagram should be easy to understand.

All synchronous elements, the shift registers and flops, are driven by an input clock signal, *clk*. The synchronous elements involved in the control path of the logic are also reset at the beginning of operation with the active-low *reset_* signal. *Reset_* is necessary to ensure that the state flop and the *ready_next* delay logic begin in an idle state when power is first applied. This is necessary, because flip-flops power up in a random, hence unknown, state. Once they are explicitly reset, they hold their state until the logic specifically changes their state. The shift register in the data path, U3, does not require a reset, because its contents are not used until eight valid data bits are shifted in, thereby flushing the eight bits with random power-up states. It would not hurt to connect U3's *clr_* pin to *reset_*, but this is not done to illustrate the option that is available. In certain logic implementations, adding reset capability to a flop may incur a penalty in terms of additional cost or circuit size. When a reset function is not free, it may be decided not to reset certain flops if their contents do not need to be guaranteed at power up, as is the case here.

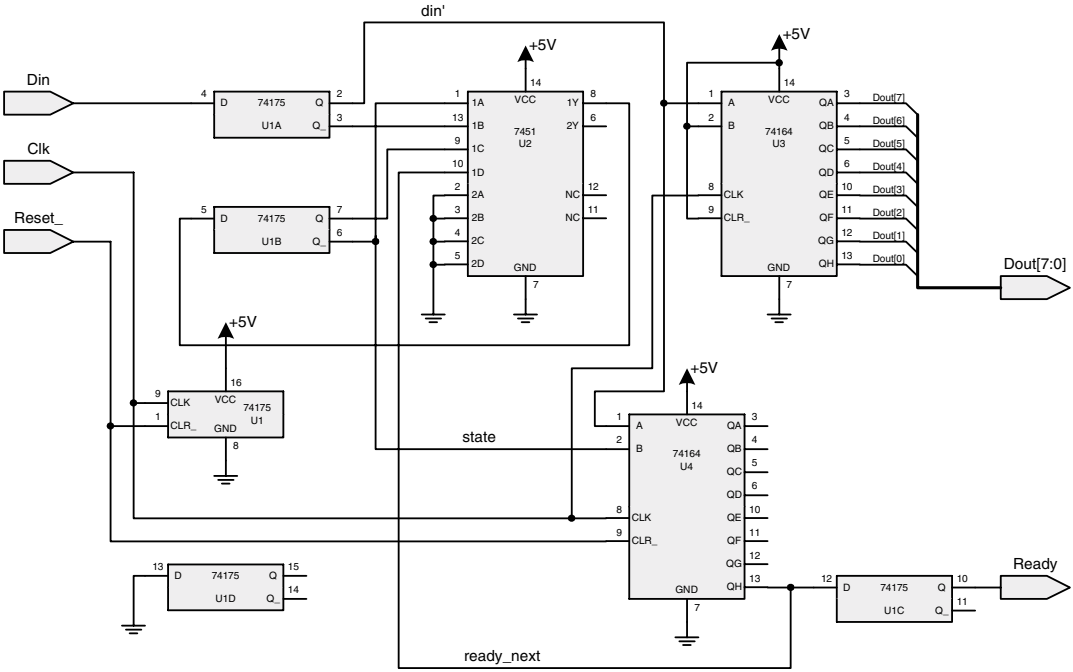

FIGURE 2.18 Serial receive logic schematic diagram.