Walking through these partial products takes extra logic and time, which is why multiplication and, by extension, division are considered advanced operations that are not nearly as common as addition and subtraction. Methods of implementing these functions require trade-offs between logic complexity and the time required to calculate a final result.

1.8 FLIP-FLOPS AND LATCHES

Logic alone does not a system make. Boolean equations provide the means to transform a set of inputs into deterministic results. However, these equations have no ability to store the results of previous calculations upon which new calculations can be made. The preceding adder logic continually recalculates the sum of two inputs. If either input is removed from the circuit, the sum disappears as well. A series of numbers that arrive one at a time cannot be summed, because the adder has no means of storing a running total. Digital systems operate by maintaining *state* to advance through sequential steps in an algorithm. State is the system's ability to keep a record of its progress in a particular sequence of operations. A system's state can be as simple as a counter or an accumulated sum.

State-full logic elements called *flip-flops* are able to indefinitely hold a specific state (0 or 1) until a new state is explicitly loaded into them. Flip-flops load a new state when triggered by the transition of an input *clock*. A clock is a repetitive binary signal with a defined period that is composed of 0 and 1 phases as shown in Fig. 1.10. In addition to a defined period, a clock also has a certain *duty cycle*, the ratio of the duration of its 0 and 1 phases to the overall period. An ideal clock has a 50/50 duty cycle, indicating that its period is divided evenly between the two states. Clocks regulate the operation of a digital system by allowing time for new results to be calculated by logic gates and then capturing the results in flip-flops.

There are several types of flip-flops, but the most common type in use today is the *D flip-flop*. Other types of flip-flops include RS and JK, but this discussion is restricted to D flip-flops because of their standardized usage. A D flip-flop is often called a *flop* for short, and this terminology is used throughout the book. A basic rising-edge triggered flop has two inputs and one output as shown in Fig. 1.11a. By convention, the input to a flop is labeled D, the output is labeled Q, and the clock is represented graphically by a triangle. When the clock transitions from 0 to 1, the state at the D input is propagated to the Q output and stored until the next rising edge. State-full logic is often described through the use of a timing diagram, a drawing of logic state versus time. Figure 1.11b shows a basic flop timing diagram in which the clock's rising edge triggers a change in the flop's state. Prior to the rising edge, the flop has its initial state, Q_0, and an arbitrary 0 or 1 input is applied as D_0. The rising edge loads D_0 into the flop, which is reflected at the output. Once triggered, the flop's input can change without affecting the output until the next rising edge. Therefore, the input is labeled as "don't care," or "xxx" following the clock's rising edge.

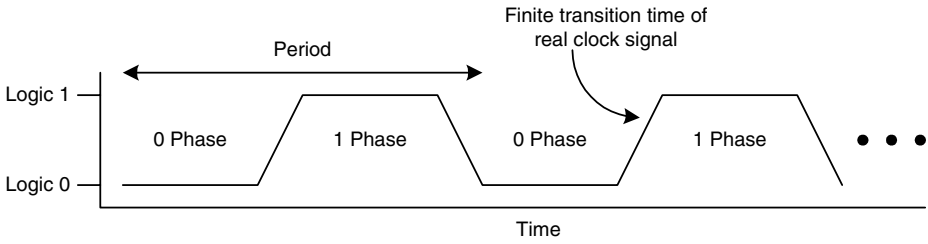

FIGURE 1.10 Digital clock signal.

Rising-edge flops are the norm, although some flops are falling-edge triggered. A falling-edge triggered flop is indicated by placing an inversion bubble at the clock input as shown in Fig. 1.12. Operation is the same, with the exception that the polarity of the clock is inverted. The remainder of this discussion assumes rising-edge triggered flops unless explicitly stated otherwise.

There are several common feature enhancements to the basic flop, including clock-enable, set, and clear inputs and a complementary output. Clock enable is used as a triggering qualifier each time a rising clock edge is detected. The D input is loaded only if clock enable is set to its active state. Inputs in general are defined by device manufacturers to be either active-low or active-high. An active-low signal is effective when set to 0, and an active-high signal is effective when set to 1. Signals are assumed to be active-high unless otherwise indicated. Active-low inputs are commonly indicated by the same inversion bubble used to indicate a falling-edge clock. When a signal is driven to its active state, it is said to be *asserted*. A signal is *de-asserted* when driven to its inactive state. Set and clear inputs explicitly force a flop to a 1 or 0 state, respectively. Such inputs are often used to initialize a digital system to a known state when it is first turned on. Otherwise, the flop powers up in a random state, which can cause problems for certain logic. Set and clear inputs can be either *synchronous* or *asynchronous*. Synchronous inputs take effect only on the rising clock edge, while asynchronous inputs take effect immediately upon being asserted. A complementary output is simply an inverted copy of the main output.

A truth table for a flop enhanced with the features just discussed is shown in Table 1.10. The truth table assumes a synchronous, active-high clock enable (EN) and synchronous, active-low set and clear inputs. The rising edge of the clock is indicated by the $\uparrow$ symbol. When the clock is at either static value, the outputs of the flop remain in their existing states. When the clock rises, the D, EN, $\overline{\text{CLR}}$, and $\overline{\text{SET}}$ inputs are sampled and acted on accordingly. As a general rule, conflicting information such as asserting $\overline{\text{CLR}}$ and $\overline{\text{SET}}$ at the same time should be avoided, because unknown results may arise. The exact behavior in this case depends on the specific flop implementation and may vary by manufacturer.

A basic application of flops is a binary ripple counter. Multiple flops can be cascaded as shown in Fig. 1.13 such that each complementary output is fed back to that flop's input and also used to clock the next flop. The current count value is represented by the noninverted flop outputs with the first flop representing the LSB. A three-bit counter is shown with an active-low reset input so that the counter can be cleared to begin at zero. The counter circuit diagram uses the standard convention of

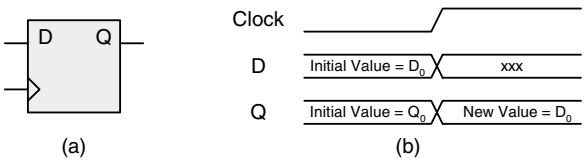

FIGURE 1.11 Rising-edge triggered flop.

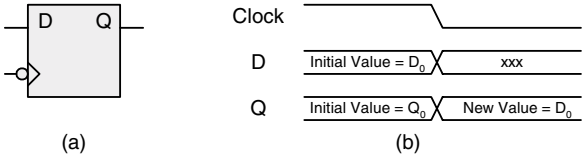

FIGURE 1.12 Falling-edge triggered flop.