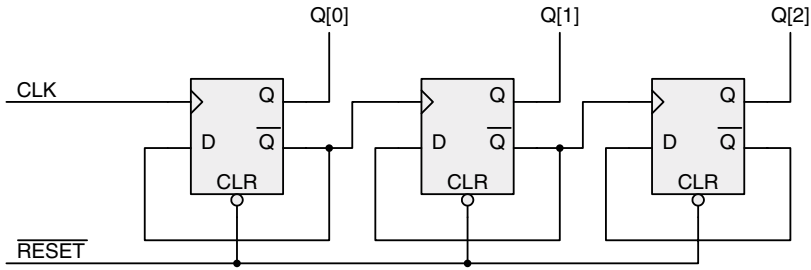

FIGURE 1.13 Three-bit ripple counter.

TABLE 1.10 Enhanced Flop Truth Table

Clock	D	EN	$\overline{CLR}$	$\overline{SET}$	Q	$\overline{Q}$
0	X	X	X	X	Q_{static}	$\overline{Q_{static}}$
↑	0	0	1	1	Q_{static}	$\overline{Q_{static}}$
↑	0	1	1	1	0	1
↑	1	1	1	1	1	0
↑	X	X	0	1	0	1
↑	X	X	1	0	1	0
↑	X	X	0	0	?	?
1	X	X	X	X	Q_{static}	$\overline{Q_{static}}$

showing electrical connectivity between intersecting wires by means of a junction dot. Wires that cross without a dot at their intersection are not electrically connected.

The ripple counter's operation is illustrated in Fig. 1.14. Each bit starts out at zero if $\overline{RESET}$ is asserted. Counting begins on the first rising edge of CLK following the de-assertion of $\overline{RESET}$. The LSB, Q[0], increments from 0 to 1, because its D input is driven by the complementary output, which is 1. The complementary output transitions to 0, which does not trigger the Q[1] rising-edge flop, but IT does set up the conditions for a trigger after the next CLK rising edge. When CLK rises again, Q[0] transitions back to 0, and $\overline{Q[0]}$ transitions to 1, forming a rising edge to trigger Q[1], which loads a 1. This sequence continues until the count value reaches 7, at which point the counter rolls over to zero, and the sequence begins again.

An undesirable characteristic of the ripple counter is that it takes longer for a new count value to stabilize as the number of bits in the counter increases. Because each flop's output clocks the next flop in the sequence, it can take some time for all flops to be updated following the CLK rising edge. Slow systems may not find this burdensome, but the added ripple delay is unacceptable in most high-speed applications. Ways around this problem will be discussed shortly.

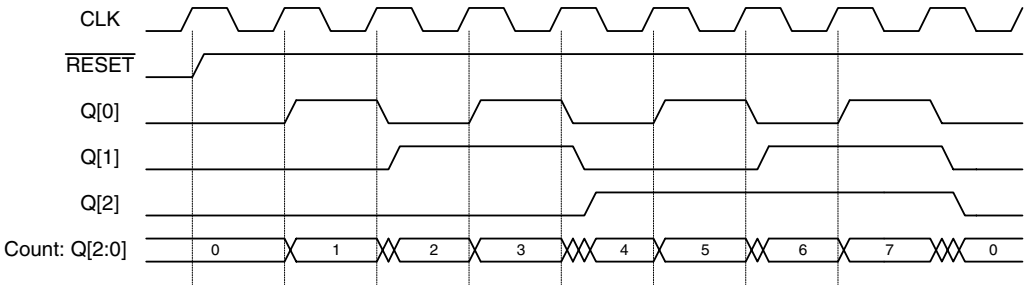

FIGURE 1.14 Ripple counter timing diagram.

A relative of the flop is the D-type *latch*, which is also capable of retaining its state indefinitely. A latch has a D input, a Q output, and an enable (EN) signal. Whereas a flop transfers its input to its output only on the active clock edge, a latch continuously transfers D to Q while EN is active. Latches are level sensitive, whereas flops are edge sensitive. A latch retains its state while EN is inactive. Table 1.11 shows the latch's truth table. Latches are simpler than flops and are unsuited to many applications in which flops are used. Latches would not substitute for flops in the preceding ripple counter example because, while the enable input is high, a continuous loop would be formed between the complementary output and input. This would result in rapid, uncontrolled oscillation at each latch during the time that the enable is held high.

TABLE 1.11 D-Latch Truth Table

EN	D	Q
0	X	Q_0
1	0	0
1	1	1

Latches are available as discrete logic elements and can also be assembled from simpler logic gates. The Boolean equation for a latch requires feeding back the output as follows:

$$Q = (EN\&D) + (\overline{EN}\&Q)$$

When EN is high, D is passed to Q. Q then feeds back to the second AND function, which maintains the state when EN is low. Latches are used in designs based on older technology that was conceived when the latch's simplicity yielded a cost savings or performance advantage. Most state-full elements today are flops unless there is a specific benefit to using a latch.

1.9 SYNCHRONOUS LOGIC

It has been shown that clock signals regulate the operation of a state-full digital system by causing new values to be loaded into flops on each active clock edge. *Synchronous logic* is the general term