11 COUNTERS

11.1 INTRODUCTION

What to Expect

Counters are a type of sequential circuit that are designed to count input pulses (normally from a clock) and then activate some output when a specific count is reached. Counters are commonly used as timers; thus, all digital clocks, microwave ovens, and other digital timing displays use some sort of counting circuit. Counters, however, have many other uses including devices as diverse as speedometers and frequency dividers. The following topics are included in this chapter.

- Comparing synchronous and asynchronous counters
- Developing and using up, down, ring, and modulus counters
- Creating a frequency divider using a counter
- Listing some of the more common counter ICs
- Describing the common types of read-only memory ICs
- Describing the function and use of random access memory

11.2 COUNTERS

11.2.1 *Introduction*

Counters are a type of sequential circuit designed to count input pulses (normally from a clock) and then activate some output when a specific count is reached. Counters are commonly used as timers; thus, all digital clocks, microwave ovens, and other digital timing displays use some sort of counting circuit. However, counters can be used in many diverse applications. For example, a speed gauge is a counter. By attaching some sort of sensor to a rotating shaft and counting the number of revolutions for 60 seconds the Rotations Per Minute (RPM) is determined. Counters are also commonly used as frequency dividers. If a high frequency is applied to the input of a counter, but only the "tens" count is output, then the input frequency will be divided by ten. As one final example, counters can also be used to control sequential circuits or processes; each count can either activate or deactivate some part of a circuit that controls one of the sequential processes.

11.2.2 *Asynchronous Counters*

One of the simplest counters possible is an asynchronous two-bit counter. This can be built with a two *JK flip-flops* in sequence, as shown in Figure 11.1.

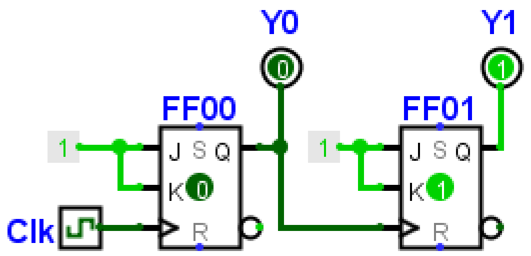

Figure 11.1: Asynchronous 2-Bit Counter

The J-K inputs on both flip-flops are tied high and the clock is wired into *FF00*. On every positive-going clock pulse the *Q* output for *FF00* toggles and that is wired to the clock input of *FF01*, toggling that output. This type of circuit is frequently called a "ripple" counter since the clock pulse must ripple through all of the flip-flops.

In Figure 11.1, assume both flip-flops start with the output low (or 00 out). On the first clock pulse, *Q_FF00* will go high, and that will send a high signal to the clock input of *FF01* which activates *Q_FF01*. At this point, the *Q* outputs for both flip-flops are high (or 11 out). On the next clock pulse, *Q_FF00* will go low, but *Q_FF01* will not change since it only toggles when the clock goes from low to high. At this point, the output is 01. On the next clock pulse, *Q_FF00* will go high and *Q_FF01* will toggle low: 10. The next clock pulse toggles *Q_FF00* to low but *Q_FF01* does not change: 00. Then the cycle repeats. This simple circuit counts: 00, 11, 10, 01. (Note, *Q_FF00* is the low-order bit.) This counter is counting backwards, but it is a trivial exercise to add the functionality needed to reverse that count.

An asynchronous three-bit counter looks much like the asynchronous two-bit counter, except that a third stage is added.

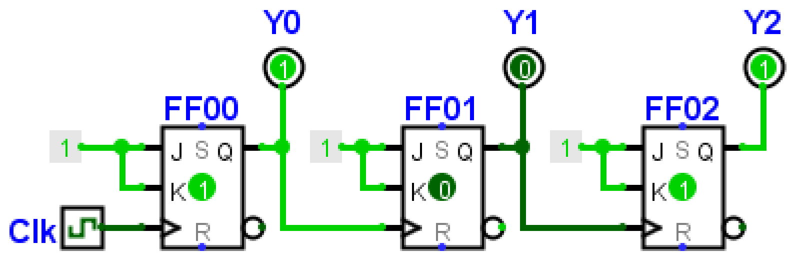

Figure 11.2: Asynchronous 3-Bit Counter

In Figure 11.2, the ripple of the clock pulse from one flip-flop to the next is more evident than in the two-bit counter. However, the overall operation of this counter is very similar to the two-bit counter.

More stages can be added so to any desired number of outputs. Asynchronous (or ripple) counters are very easy to build and require very few parts. Unfortunately, they suffer from two rather important flaws:

- Propagation Delay. As the clock pulse ripples through the various flip-flops, it is slightly delayed by each due to the simple physical switching of the circuitry within the flip-flop. Propagation delay cannot be prevented and as the number of stages increases the delay becomes more pronounced. At some point, one clock pulse will still be winding its way through all of the flip-flops when the next clock pulse hits the first stage and this makes the counter unstable.

- Glitches. If a three-bit counter is needed, there will be a very brief moment while the clock pulse ripples through the flip-flops that the output will be wrong. For example, the circuit should go from 111 to 000, but it will actually go from 111 to 110 then 100 then 000 as the "low" ripples through the flip-flops. These glitches are very short, but they may be enough to introduce errors into a circuit.

Figure 11.3 is a four-bit asynchronous counter.

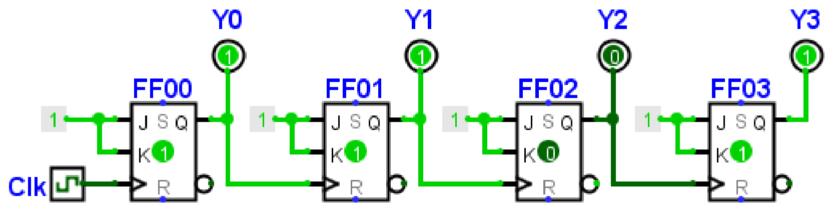

Figure 11.3: Asynchronous 4-Bit Counter

Figure 11.4 is the timing diagram obtained when the counter in Figure 11.3 is executing.

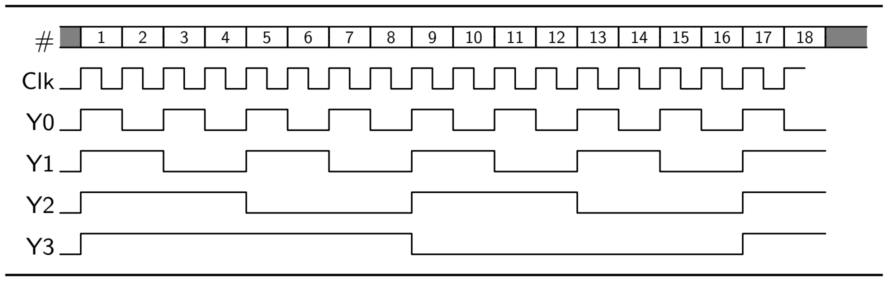

Figure 11.4: 4-Bit Asynchronous Counter Timing Diagram

Notice that *Y0* counts at half the frequency of the clock and then *Y1* counts at half that frequency and so forth. Each stage that is added will count at half the frequency of the previous stage.

11.2.3 *Synchronous Counters*

Both problems with ripple counters can be corrected with a synchronous counter, where the same clock pulse is applied to every flip-flop at one time. Here is the logic diagram for a synchronous two-bit counter:

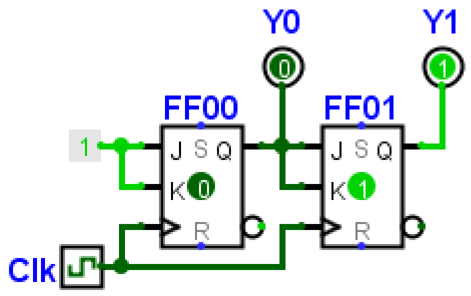

Figure 11.5: Synchronous 2-Bit Counter

Notice in this circuit, the clock is applied to both flip-flops and control is exercised by applying the output of one stage to both J and K inputs of the next stage, which effectively enables/disables that stage. When Q_FF00 is high, for example, then the next clock pulse will make FF01 change states.

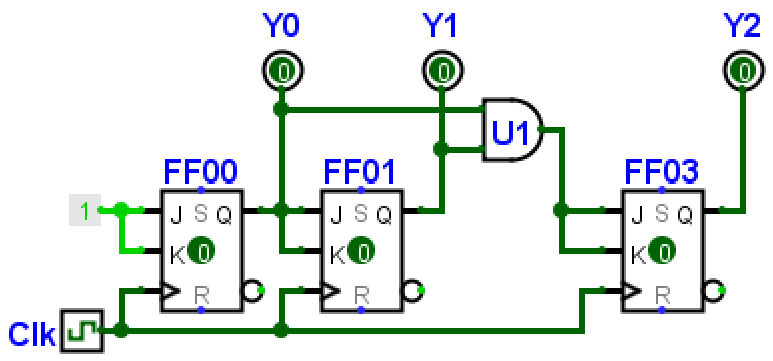

Figure 11.6: Synchronous 3-Bit Counter

A three-bit synchronous counter applies the clock pulse to each flip-flop. Notice, though, that the output from the first two stages must routed through an AND gate so *FF03* will only change when *Q_FF00* and *Q_FF01* are high. This circuit would then count properly from 000 to 111.

In general, synchronous counters become more complex as the number of stages increases since it must include logic for every stage to determine when that stage should activate. This complexity results in greater power usage (every additional gate requires power) and heat generation; however, synchronous counters do not have the propagation delay problems found in asynchronous counters.

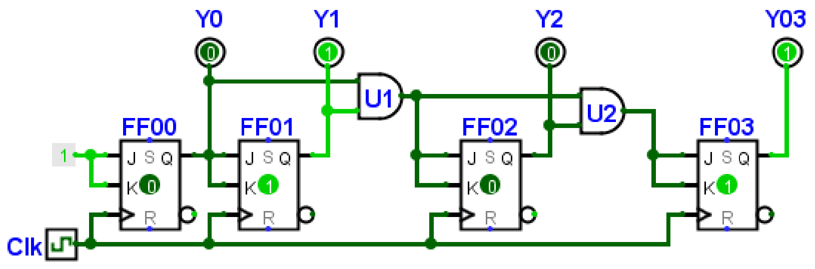

Figure 11.7: Synchronous 4-Bit Up Counter

Figure 11.8 is the timing diagram for the circuit illustrated in Figure 11.7.

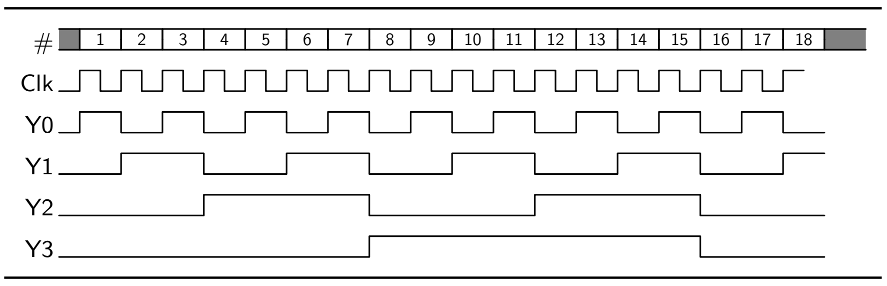

Figure 11.8: 4-Bit Synchronous Counter Timing Diagram

The timing diagram for the asynchronous counter in Figure 11.4 is the same as that for an synchronous counter in Figure 11.8 since the output of the two counters are identical. The only difference in the counters is in how the count is obtained, and designers would normally opt for a synchronous IC since that is a more efficient circuit.

11.2.3.1 *Synchronous Down Counters*

It is possible to create a counter that counts down rather than up by using the Q' outputs of the flip-flops to trigger the next stage. Figure 11.9 illustrates a four-bit down counter.

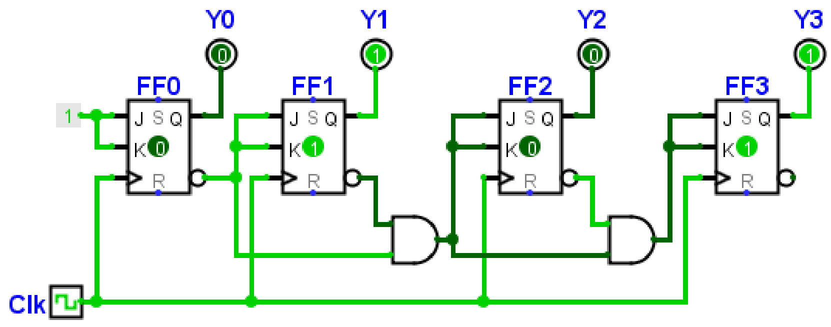

Figure 11.9: Synchronous 4-Bit Down Counter

Figure 11.10 is the timing diagram for the circuit illustrated in Figure 11.9.

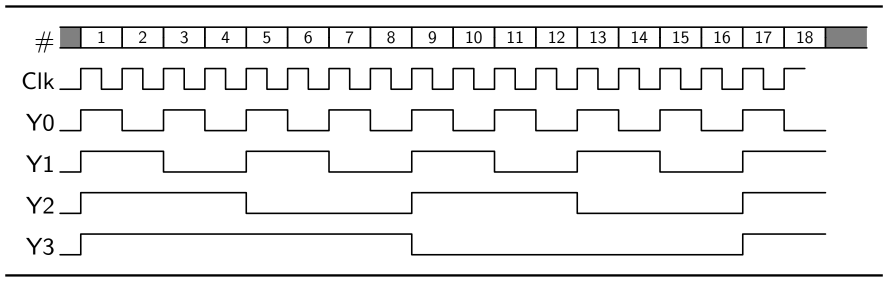

Figure 11.10: 4-Bit Synchronous Down Counter Timing Diagram

11.2.4 *Ring Counters*

A ring counter is a special kind of counter where only one output is active at a time. As an example, a four-bit ring counter may output this type of pattern:

1000 — 0100 — 0010 — 0001

Notice the high output cycles through each of the bit positions and then recycles. Ring counters are useful as controllers for processes where one process must follow another in a sequential manner. Each of the bits from the ring counter can be used to activate a different part of the overall process; thus ensuring the process runs in proper sequence. The circuit illustrated in Figure 11.11 is a 4-bit ring counter.

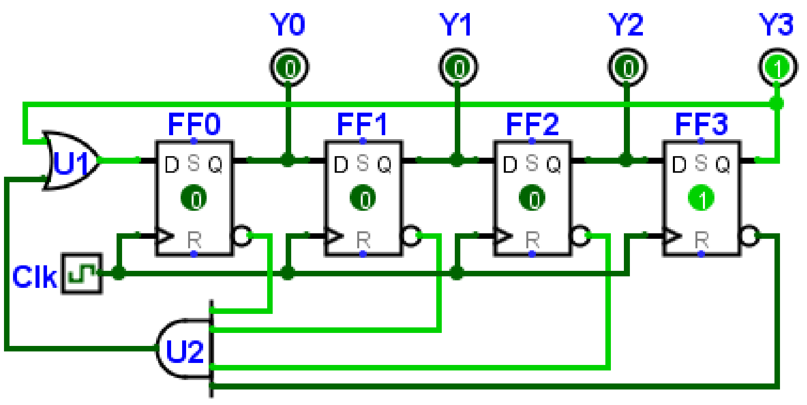

Figure 11.11: 4-Bit Ring Counter

When the circuit is initialized none of the flip-flops are active. Because Q' is high for all flip-flops, AND gate *U2* is active and that sends a high through *U1* and into the data port of *FF0*. The purpose of *U2* is to initialize *FF0* for the first count and then *U2* is never activated again. On each clock pulse the next flip-flop in sequence is activated. When *FF3* is active that output is fed back through *U1* to *FF0*, completing the ring and re-starting the process.

Figure 11.12 is the timing diagram for the ring counter illustrated in Figure 11.11.

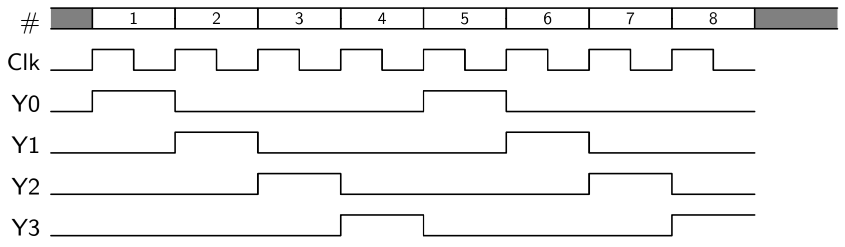

Figure 11.12: 4-Bit Ring Counter Timing Diagram

On each clock pulse a different bit is toggled high, proceeding around the four-bit nibble in a ring pattern.

11.2.4.1 *Johnson Counters*

One common modification of a ring counter is called a Johnson, or "Twisted Tail," ring counter. In this case, the counter outputs this type of pattern.

1000 – 1100 – 1110 – 1111 – 0111 – 0011 – 0001 – 0000

The circuit illustrated in Figure 11.13 is a 4-bit Johnson counter.

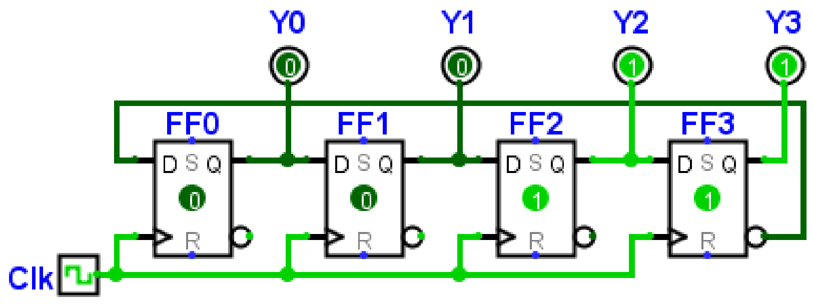

Figure 11.13: 4-Bit Johnson Counter

This is very similar to the ring counter except the feedback loop is from Q' rather than Q of $FF3$ and because of that, the AND gate initialization is no longer needed.

Figure 11.14 is the timing diagram for the Johnson counter illustrated in Figure 11.13.

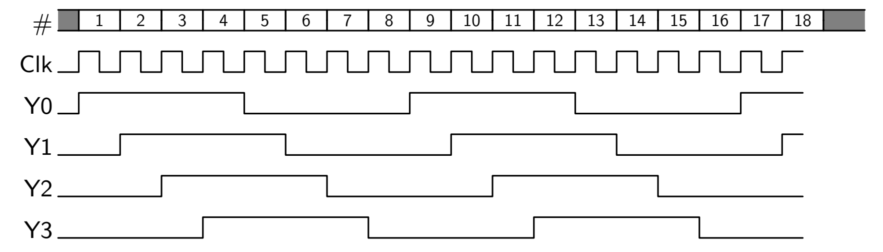

Figure 11.14: 4-Bit Johnson Counter Timing Diagram

11.2.5 *Modulus Counters*

Each of the counters created so far have had one significant flaw, they only count up to a number that is a power of two. A two-bit counter counts from zero to three, a three-bit counter counts from zero to seven, a four-bit counter counts from zero to 15, and so forth. With a bit more work a counter can be created that stops at some other number. These types of counters are called "modulus counters" and an example of one of the most common modulus counters is a decade counter that counts from zero to nine. The circuit illusted in Figure 11.15 is a decade counter.

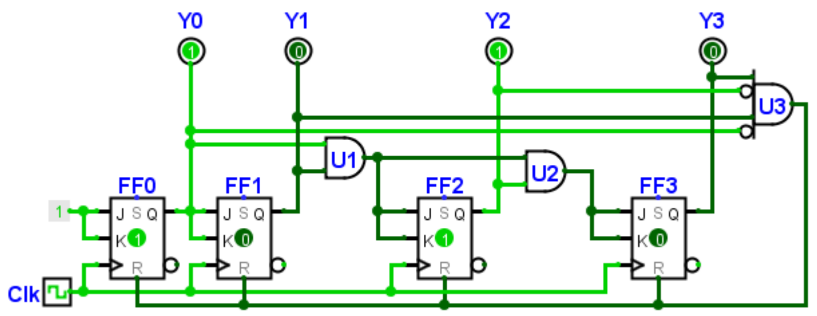

Figure 11.15: Decade Counter

This is only a four-bit counter (found in Figure 11.7) but with an additional AND gate (*U3*). The inputs for that AND gate are set such that when *Y0*-*Y3* are 1010 then the gate will activate and reset all four flip-flops.

Figure 11.16 is the timing diagram obtained from the decade counter illustrated in Figure 11.15.

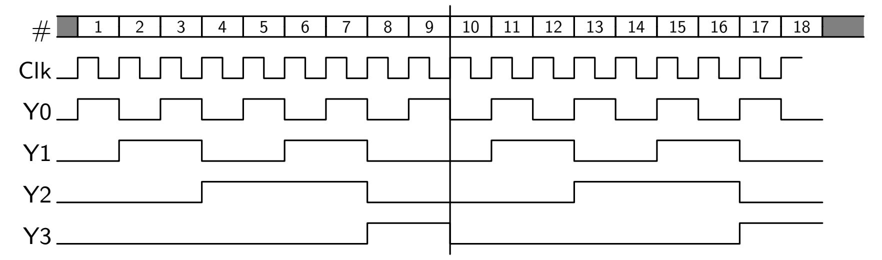

Figure 11.16: 4-Bit Decade Counter Timing Diagram

The timing diagram shows the count increasing with each clock pulse (for example, at 8 the outputs are 1000) until pulse number ten, when the bits reset to 0000 and the count starts over. A vertical line was added at count ten for reference.

11.2.6 *Up-Down Counters*

In this chapter both up and down counters have been considered; however counters can also be designed to count both up and down. These counters are, of course, more complex than the simple counters encountered so far and Figure 11.17 illustrates an up-down counter circuit.

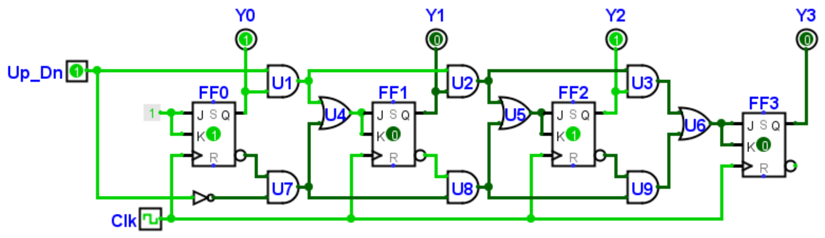

Figure 11.17: Up-Down Counter

When the *Up_Dn* bit is high the counter counts up but when that bit is low then the counter counts down. This counter is little more than a merging of the up counter in Figure 11.7 and the down counter in Figure 11.9. *U1*-*U3* transmit the *Q* outputs from each flip-flop to the next stage while *U7*-*U9* transmit the *Q′* outputs from each flip-flop to the next stage. The E bit activates one of those two banks of AND gates.

No timing diagram is provided for this circuit since that diagram would be the same as for the up counter (Figure 11.8 or the down counter (Figure 11.10), depending on the setting of the *Up_Dn* bit.

11.2.7 *Frequency Divider*

Often, a designer creates a system that may need various clock frequencies throughout its subsystems. In this case, it is desirable to have only one main pulse generator, but divide the frequency from that generator so other frequencies are available where they are needed. Also, by using a common clock source all of the frequencies are easier to synchronize when needed.

The circuit illustrated in Figure 11.18 is the same synchronous two-bit counter first considered in Figure 11.5.

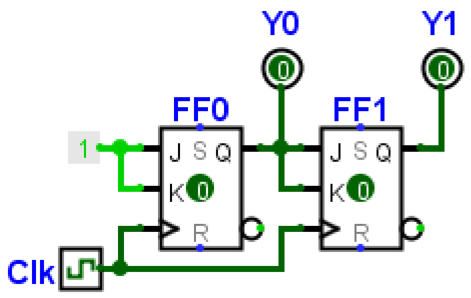

Figure 11.18: Synchronous 2-Bit Counter

The circuit in Figure 11.18 produces the timing diagram seen in Figure 11.19

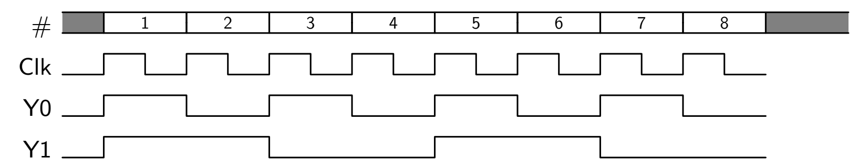

Figure 11.19: Frequency Divider

Notice that *Y0* is half of the clock's frequency and *Y1* is one-quarter of the clock's frequency. If the circuit designer used Y1 as a timer for some subcircuit then that circuit would operate at one-quarter of the main clock frequency. It is possible to use only one output port from a modulus counter, like the decade counter found in Figure 11.15, and get any sort of division of the main clock desired.

11.2.8 *Counter Integrated Circuits (IC)*

In practice, most designers do not build counter circuits since there are so many types already commercially available. When a counter is needed, a designer can select an appropriate counter from those available on the market. Here are just a few as examples of what is available:

IC	Function
74x68	dual four-bit decade counter
74x69	dual four-bit binary counter
74x90	decade counter (divide by 2 and divide by 5 sections)
74x143	decade counter/latch/decoder/seven-segment driver
74x163	synchronous four-bit binary counter
74x168	synchronous four-bit up/down decade counter
74x177	presettable binary counter/latch
74x291	four-bit universal shift register and up/down counter

Table 11.1: Counter IC's

11.3 Memory

11.3.1 *Read-Only Memory*

Read Only Memory (ROM) is an IC that contains tens-of-millions of registers (memory locations) to store information. Typically, ROM stores microcode (like bootup routines) and computer settings that do not change. There are several types of ROM: mask, programmable, and erasable.

- Mask ROMs are manufactured such that memory locations already filled with data and that cannot be altered by the end user. They are called "mask" ROMs since the manufacturing process includes applying a mask to the circuit as it is being created.

- Programmable Read-Only Memorys (PROMs) are a type of ROM that can be programmed by the end user, but it cannot be altered after that initial programming. This permits a designer to distribute some sort of program on a chip that is designed to be installed in some device and then never changed.

- Erasable PROMs can be programmed by the end user and then be later altered. An example of where an erasable PROM would be used is in a computer's Basic Input/Output System (BIOS) (the operating system that boots up a computer), where the user can alter certain "persistent" computer specifications, like the device boot order.

Two major differences between ROM and RAM are 1) ROM's ability to hold data when the computer is powered off and 2) altering the data in RAM is much faster than in an erasable PROM.

11.3.2 *Random Access Memory*

RAM is an integrated circuit that contains tens-of-millions of registers (memory locations) to store information. The RAM IC is designed to quickly store data found at its input port and then look-up and return stored data requested by the circuit. In a computer operation, RAM contains both program code and user input (for example, the *LibreOffice Writer* program along with whatever document the user is working on). There are two types of RAM: dynamic and static. Dynamic Random Access Memory (DRAM) stores bits in such a way that it must be "refreshed" (or "strobed") every few milliseconds. Static Random Access Memory (SRAM) uses flip-flops to store data so it does not need to be refreshed. One enhancement to DRAMs was to package several in a single IC and then synchronize them so they act like a single larger memory; these are called Synchronized Dynamic Random Access Memorys (SDRAMs) and they are very popular for camera and cell phone memory.

Both ROM and RAM circuits use registers, flip-flops, and other components already considered in this chapter, but by the millions rather than just four or so at a time. There are no example circuits or timing diagrams for these devices in this book; however, the lab manual that accompanies this book includes an activity for a ROM device.