11 COUNTERS AND REGISTERS

Counters and *registers* belong to the category of MSI sequential logic circuits. They have similar architecture, as both counters and registers comprise a cascaded arrangement of more than one flip flop with or without combinational logic devices. Both constitute very important building blocks of sequential logic, and different types of counter and register available in integrated circuit (IC) form are used in a wide range of digital systems. While counters are mainly used in counting applications, where they either measure the time interval between two unknown time instants or measure the frequency of a given signal, registers are primarily used for the temporary storage of data present at the output of a digital circuit before they are fed to another digital circuit. We are all familiar with the role of different types of register used inside a microprocessor, and also their use in microprocessor-based applications. Because of the very nature of operation of registers, they form the basis of a very important class of counters called *shift counters*.

11.1 Ripple (Asynchronous) Counter

A *ripple counter* is a cascaded arrangement of flip-flops where the output of one flip-flop drives the clock input of the following flip-flop. The number of flip-flops in the cascaded arrangement depends upon the number of different logic states that it goes through before it repeats the sequence, a parameter known as the modulus of the counter.

In a ripple counter, also called an *asynchronous counter* or a *serial counter*, the clock input is applied only to the first flip-flop, also called the input flip-flop, in the cascaded arrangement. The clock input to any subsequent flip-flop comes from the output of its immediately preceding flip-flop. For instance, the output of the first flip-flop acts as the clock input to the second flip-flop, the output of the second flip-flop feeds the clock input of the third flip-flop and so on. In general, in an arrangement of n flip-flops, the clock input to the nth flip-flop comes from the output of the (n−1)th flip-flop for n>1.

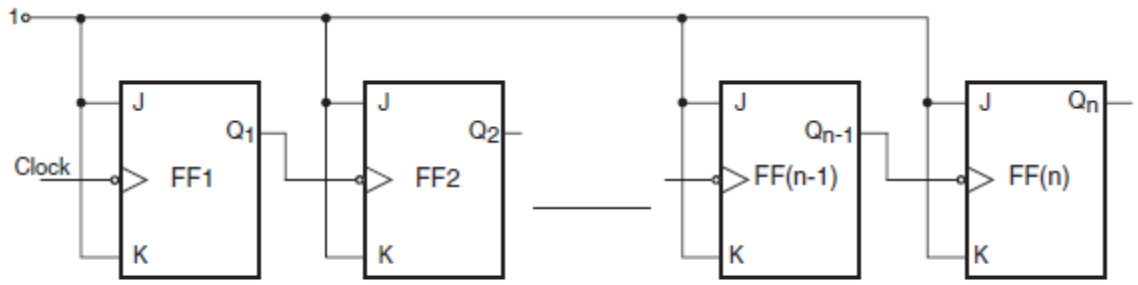


Figure 11.1 Generalized block schematic of *n*-bit binary ripple counter.

Figure 11.1 shows the generalized block schematic arrangement of an n-bit binary ripple counter.

As a natural consequence of this, not all flip-flops change state at the same time. The second flip-flop can change state only after the output of the first flip-flop has changed its state. That is, the second flip-flop would change state a certain time delay after the occurrence of the input clock pulse owing to the fact that it gets its own clock input from the output of the first flip-flop and not from the input clock. This time delay here equals the sum of propagation delays of two flip-flops, the first and the second flip-flops. In general, the nth flip-flop will change state only after a delay equal to n times the propagation delay of one flip-flop. The term ‘ripple counter’ comes from the mode in which the clock information ripples through the counter. It is also called an ‘asynchronous counter’ as different flip-flops comprising the counter do not change state in synchronization with the input clock.

A major problem with ripple counters arises from the propagation delay of the flip-flops constituting the counter. As mentioned earlier, the effective propagation delay in a ripple counter is equal to the sum of propagation delays due to different flip-flops. The situation becomes worse with increase in the number of flip-flops used to construct the counter, which is the case in larger bit counters.

11.2 Synchronous Counter

In a *synchronous counter*, also known as a *parallel counter*, all the flip-flops in the counter change state at the same time in synchronism with the input clock signal. The clock signal in this case is simultaneously applied to the clock inputs of all the flip-flops. The delay involved in this case is equal to the propagation delay of one flip-flop only, irrespective of the number of flip-flops used to construct the counter. In other words, the delay is independent of the size of the counter.

11.3 Modulus of a Counter

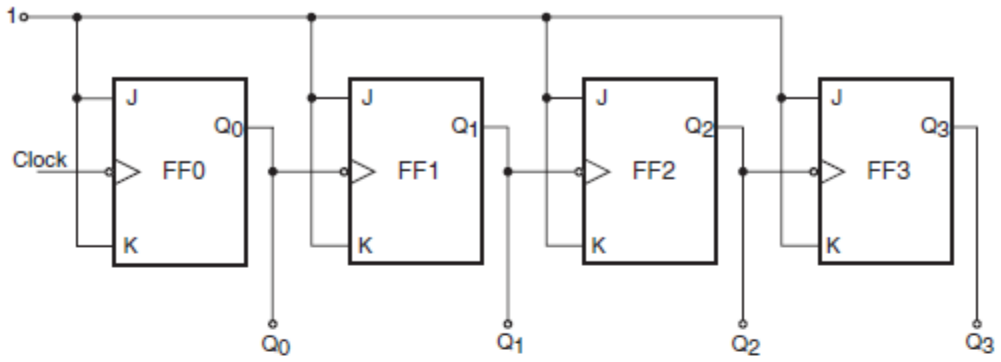
The *modulus* (MOD number) of a counter is the number of different logic states it goes through before it comes back to the initial state to repeat the count sequence. An n-bit counter that counts through all its natural states and does not skip any of the states has a modulus of 2^n. We can see that such counters have a modulus that is an integral power of 2, that is, 2, 4, 8, 16 and so on. These can be modified with the help of additional combinational logic to get a modulus of less than 2^n.

To determine the number of flip-flops required to build a counter having a given modulus, identify the smallest integer m that is either equal to or greater than the desired modulus and is also equal to an integral power of 2. For instance, if the desired modulus is 10, which is the case in a decade counter, the smallest integer greater than or equal to 10 and which is also an integral power of 2 is 16. The number of flip-flops in this case would be 4, as $16 = 2^4$. On the same lines, the number of flip-flops required to construct counters with MOD numbers of 3, 6, 14, 28 and 63 would be 2, 3, 4, 5 and 6 respectively. In general, the arrangement of a minimum number of N flip-flops can be used to construct any counter with a modulus given by the equation

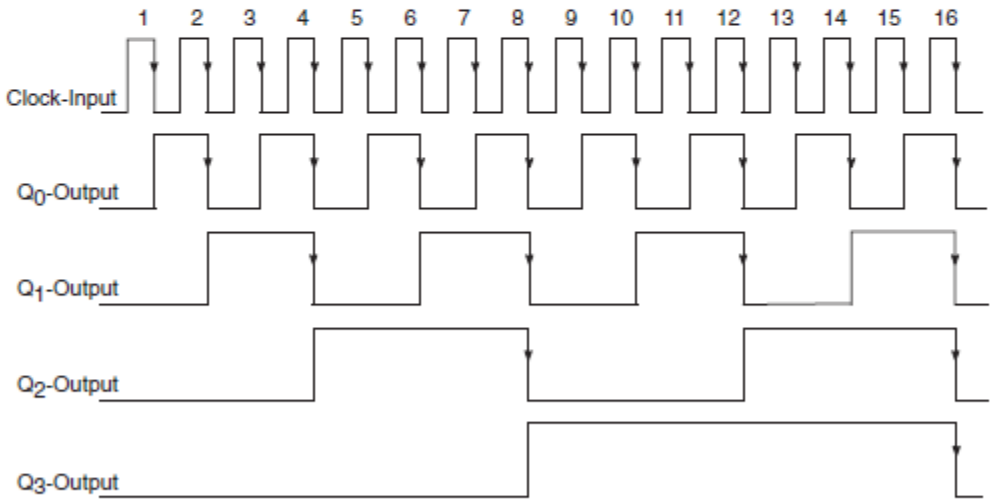
$$(2^{N-1} + 1) \leq modulus \leq 2^N \qquad (11.1)$$

11.4 Binary Ripple Counter – Operational Basics

The operation of a binary ripple counter can be best explained with the help of a typical counter of this type. Figure 11.2(a) shows a four-bit ripple counter implemented with negative edge-triggered *J-K* flip-flops wired as toggle flip-flops. The output of the first flip-flop feeds the clock input of the second, and the output of the second flip-flop feeds the clock input of the third, the output of which in turn feeds the clock input of the fourth flip-flop. The outputs of the four flip-flops are designated as Q_0 (LSB flip-flop), Q_1, Q_2 and Q_3 (MSB flip-flop). Figure 11.2(b) shows the waveforms appearing at Q_0, Q_1, Q_2 and Q_3 outputs as the clock signal goes through successive cycles of trigger pulses. The counter functions as follows.



(a)



(b)

Figure 11.2 Four-bit binary ripple counter.

Let us assume that all the flip-flops are initially cleared to the '0' state. On HIGH-to-LOW transition of the first clock pulse, Q_0 goes from '0' to '1' owing to the toggling action. As the flip-flops used are negative edge-triggered ones, the '0' to '1' transition of Q_0 does not trigger flip-flop FF1. FF1, along with FF2 and FF3, remains in its '0' state. So, on the occurrence of the first negative-going clock transition, Q0 = 1, Q1 = 0, Q2 = 0 and Q3 = 0.

On the HIGH-to-LOW transition of the second clock pulse, Q0 toggles again. That is, it goes from '1' to '0'. This '1' to '0' transition at the Q0 output triggers FF1, the output Q1 of which goes from '0' to '1'. The Q2 and Q3 outputs remain unaffected. Therefore, immediately after the occurrence of the second HIGH-to-LOW transition of the clock signal, Q0 = 0, Q1 = 1, Q2= 0 and Q3 = 0. On similar lines, we can explain the logic status of Q0, Q1, Q2 and Q3 outputs immediately after subsequent clock transitions. The logic status of outputs for the first 16 relevant (HIGH-to-LOW in the present case) clock signal transitions is summarized in Table 11.1.

Thus, we see that the counter goes through 16 distinct states from 0000 to 1111 and then, on the occurrence of the desired transition of the sixteenth clock pulse, it resets to the original state of 0000 from where it had started. In general, if we had N flip-flops, we could count up to 2^N pulses before the

counter resets to the initial state. We can also see from the Q0, Q1, Q2 and Q3 waveforms, as shown in Fig. 11.2(b), that the frequencies of the Q0, Q1, Q2 and Q3 waveforms are f/2,f/4, f/8 and f/16 respectively.

Table 11.1 Output logic states for different clock signal transitions for a four-bit binary ripple counter.

Clock signal transition number	Q_0	Q_1	Q_2	Q_3
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	1	1	0	0
After fourth clock transition	0	0	1	0
After fifth clock transition	1	0	1	0
After sixth clock transition	0	1	1	0
After seventh clock transition	1	1	1	0
After eighth clock transition	0	0	0	1
After ninth clock transition	1	0	0	1
After tenth clock transition	0	1	0	1
After eleventh clock transition	1	1	0	1
After twelfth clock transition	0	0	1	1
After thirteenth clock transition	1	0	1	1
After fourteenth clock transition	0	1	1	1
After fifteenth clock transition	1	1	1	1
After sixteenth clock transition	0	0	0	0

Here, f is the frequency of the clock input. This implies that a counter of this type can be used as a divide-by-2^N circuit, where N is the number of flip-flops in the counter chain. In fact, such a counter provides frequency-divided outputs of $f/2^N$, $f/2^{N-1}$, $f/2^{N-2}$, $f/2^{N-3}$, _ _ _ , f/2 at the outputs of the Nth, (N− 1)th, (N − 2)th, (N− 3)th, _ _ _ , first flip-flops. In the case of a four-bit counter of the type shown in Fig. 11.2(a), outputs are available at f/2 from the Q0 output, at f/4 from the Q1 output, at f/8 from the Q2 output and at f/16 from the Q3 output. It may be noted that frequency division is one of the major applications of counters.

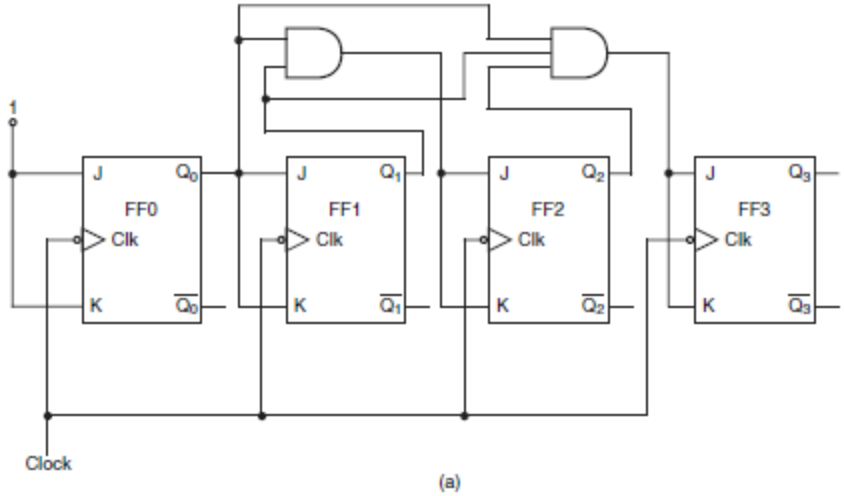
11.5 Synchronous (or Parallel) Counters

Ripple counters discussed thus far in this chapter are asynchronous in nature as the different flip-flops comprising the counter are not clocked simultaneously and in synchronism with the clock pulses. The total propagation delay in such a counter, as explained earlier, is equal to the sum of propagation delays due to different flip-flops. The propagation delay becomes prohibitively large in a ripple counter with a large count. On the other hand, in a synchronous counter, all flip-flops in the counter are clocked simultaneously in synchronism with the clock, and as a consequence all flip-flops change state at the same time. The propagation delay in this case is independent of the number of flip-flops used.

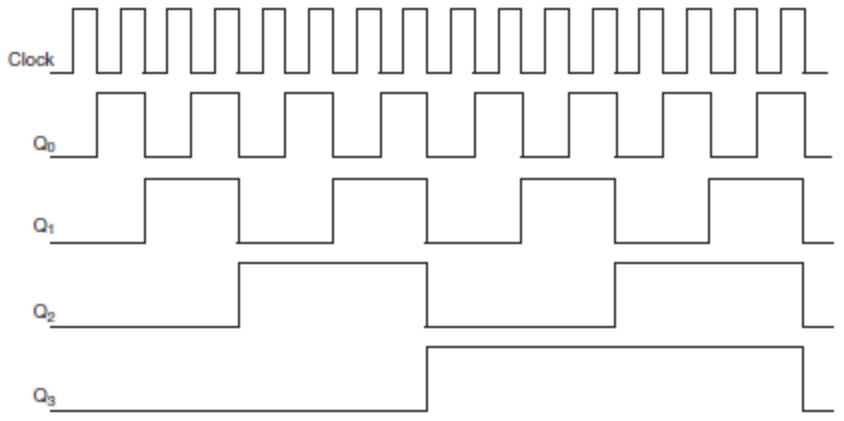
Since the different flip-flops in a synchronous counter are clocked at the same time, there needs to additional logic circuitry to ensure that the various flip-flops toggle at the right time. For instance, if we look at the count sequence of a four-bit binary counter shown in Table 11.4, we find that flip-flop FF0 toggles with every clock pulse, flip-flop FF1 toggles only when the output of FF0 is in the '1' state, flip-flop FF2 toggles only with those clock pulses when the outputs of FF0 and FF1 are both in the logic '1' state and flip-flop FF3 toggles only with those clock pulses when Q0,Q1 and Q2 are all in the logic '1' state. Such logic can be easily implemented with AND gates. Figure 11.9(a) shows the schematic arrangement of a four-bit synchronous counter. The timing waveforms are shown in Fig. 11.9(b).

Table 11.4 Count sequence of a four-bit binary counter.

Count	Q_3	Q_2	Q_1	Q_0	Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1



(a)



(b)

Figure 11.9 Four-bit synchronous counter.

11.12 Shift Register

A *shift register* is a digital device used for storage and transfer of data. The data to be stored could be data appearing at the output of an encoding matrix before they are fed to the main digital system for processing or they might be the data present at the output of a microprocessor before they are fed to the driver circuitry of the output devices. The shift register thus forms an important link between the main digital system and the input/output channels. The shift registers can also be configured to construct some special types of counter that can be used to perform a number of arithmetic operations such as subtraction, multiplication, division, complementation, etc. The basic building block in all shift registers is the flip-flop, mainly a D-type flip-flop. Although in many of the commercial shift register ICs their internal circuit diagram might indicate the use of *R-S* flip-flops, a careful examination will reveal that these *R-S* flip-flops have been wired as D flip-flops only.

The storage capacity of a shift register equals the total number of bits of digital data it can store, which in turn depends upon the number of flip-flops used to construct the shift register. Since each flip-flop can store one bit of data, the storage capacity of the shift register equals the number of flip-flops used. As an example, the internal architecture of an eight-bit shift register will have a cascade arrangement of eight flip-flops.

Based on the method used to load data onto and read data from shift registers, they are classified as serial-in serial-out (SISO) shift registers, serial-in parallel-out (SIPO) shift registers, parallel-in serial-out (PISO) shift registers and parallel-in parallel-out (PIPO) shift registers.

Figure 11.34 shows a circuit representation of the above-mentioned four types of shift register.

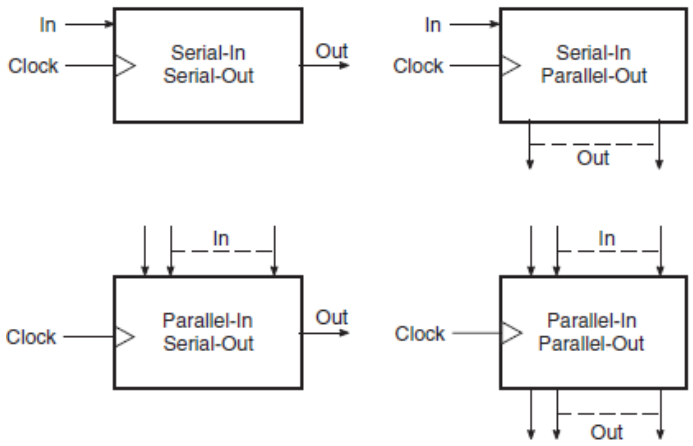


Figure 11.34 Circuit representation of shift registers.

11.12.1 Serial-In Serial-Out Shift Register

Figure 11.35 shows the basic four-bit serial-in serial-out shift register implemented using D flip-flops.

The circuit functions as follows. A reset applied to the CLEAR input of all the flip-flops resets their Q outputs to 0s. Refer to the timing waveforms of Fig. 11.36. The waveforms shown include the clock pulse train, the waveform representing the data to be loaded onto the shift register and the Q outputs of different flip-flops.

The flip-flops shown respond to the LOW-to-HIGH transition of the clock pulses as indicated by their logic symbols. During the first clock transition, the Q_A output goes from logic '0' to logic '1'.

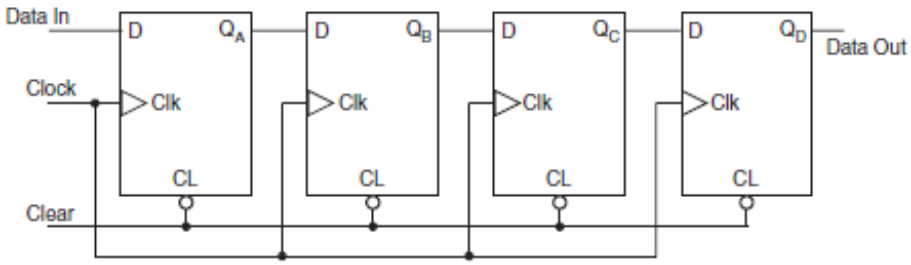


Figure 11.35 Serial-in, serial-out shift register.

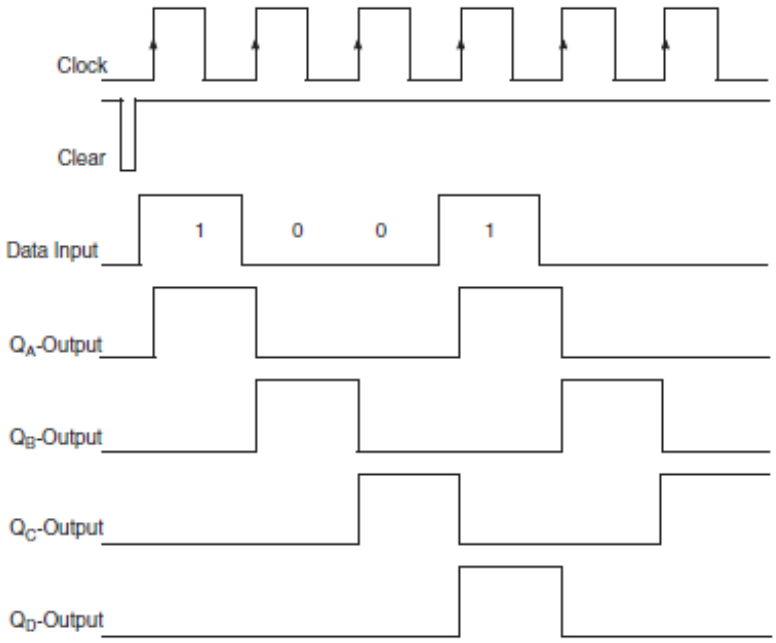


Figure 11.36 Timing waveforms for the shift register of Fig. 11.35.

The outputs of the other three flip-flops remain in the logic '0' state as their D inputs were in the logic '0' state at the time of clock transition. During the second clock transition, the Q_A output goes from logic '1' to logic '0' and the Q_B output goes from logic '0' to logic '1', again in accordance with the logic status of the D inputs at the time of relevant clock transition.

Thus, we have seen that a logic '1' that was present at the data input prior to the occurrence of the first clock transition has reached the Q_B output at the end of two clock transitions. This bit will reach the Q_D output at the end of four clock transitions. In general, in a four-bit shift register of the type shown in Fig. 11.35, a data bit present at the data input terminal at the time of the nth clock transition reaches the Q_D output at the end of the (n+4)th clock transition. During the fifth and subsequent clock transitions, data bits continue to shift to the right, and at the end of the eighth clock transition the shift register is again reset to all 0s. Thus, in a four-bit serial-in serial-out shift register, it takes four clock cycles to load the data bits and another four cycles to read the data bits out of the register.

The contents of the register for the first eight clock cycles are summarized in Table 11.14. We can see that the register is loaded with the four-bit data in four clock cycles, and also that the stored four-bit data are read out in the subsequent four clock cycles.

Table 11.14 Contents of four-bit serial-in serial-out shift register for the first eight clock cycles.

Clock	Q_A	Q_B	Q_C	Q_D
Initial contents	0	0	0	0
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	0	0	1	0
After fourth clock transition	**1**	**0**	**0**	**1**
After fifth clock transition	0	1	0	0
After sixth clock transition	0	0	1	0
After seventh clock transition	0	0	0	1
After eighth clock transition	**0**	**0**	**0**	**0**