## C H A P T E R 5

$\bigcirc 0_{4}$
FLIP-FLOPS AND - \&RELATED DEVICES

## - OUTLINE

5-1 NAND Gate Latch
5-2 NOR Gate Latch
5-3 Troubleshooting Case Study
5-4 Digital Pulses
5-5 Clock Signals and Clocked Flip-Flops

5-6 Clocked S-R Flip-Flop
5-7 Clocked J-K Flip-Flop
5-8 Clocked D Flip-Flop
5-9 $\quad D$ Latch (Transparent Latch)
5-10 Asynchronous Inputs
5-11 IEEE/ANSI Symbols
5-12 Flip-Flop Timing Considerations

5-13 Potential Timing Problem in FF Circuits
5-14 Flip-Flop Applications
5-15 Flip-Flop Synchronization

5-16 Detecting an Input Sequence

5-17 Data Storage and Transfer
5-18 Serial Data Transfer: Shift Registers

5-19 Frequency Division and Counting
5-20 Microcomputer Application
5-21 Schmitt-Trigger Devices
5-22 One-Shot (Monostable Multivibrator)
5-23 Clock Generator Circuits
5-24 Troubleshooting Flip-Flop Circuits
5-25 Sequential Circuits Using HDL

5-26 Edge-Triggered Devices
5-27 HDL Circuits with Multiple Components

## OBJECTIVES

Upon completion of this chapter, you will be able to:

- Construct and analyze the operation of a latch flip-flop made from NAND or NOR gates.
- Describe the difference between synchronous and asynchronous systems.
- Understand the operation of edge-triggered flip-flops.
- Analyze and apply the various flip-flop timing parameters specified by the manufacturers.
- Understand the major differences between parallel and serial data transfers.
- Draw the output timing waveforms of several types of flip-flops in response to a set of input signals.
- Recognize the various IEEE/ANSI flip-flop symbols.
- Use state transition diagrams to describe counter operation.
- Use flip-flops in synchronization circuits.
- Connect shift registers as data transfer circuits.
- Employ flip-flops as frequency-division and counting circuits.
- Understand the typical characteristics of Schmitt triggers.
- Apply two different types of one-shots in circuit design.
- Design a free-running oscillator using a 555 timer.
- Recognize and predict the effects of clock skew on synchronous circuits.
- Troubleshoot various types of flip-flop circuits.
- Write HDL code for latches.
- Use logic primitives, components, and libraries in HDL code.
- Build structural level circuits from components.


## INTRODUCTION

The logic circuits considered thus far have been combinational circuits whose output levels at any instant of time are dependent on the levels present at the inputs at that time. Any prior input-level conditions have no effect on the present outputs because combinational logic circuits have no memory. Most digital systems consist of both combinational circuits and memory elements.

Figure 5-1 shows a block diagram of a general digital system that combines combinational logic gates with memory devices. The combinational portion accepts logic signals from external inputs and from the outputs of the memory elements. The combinational circuit operates on these inputs

FIGURE 5-1 General digital system diagram.

to produce various outputs, some of which are used to determine the binary values to be stored in the memory elements. The outputs of some of the memory elements, in turn, go to the inputs of logic gates in the combinational circuits. This process indicates that the external outputs of a digital system are functions of both its external inputs and the information stored in its memory elements.

The most important memory element is the flip-flop, which is made up of an assembly of logic gates. Even though a logic gate, by itself, has no storage capability, several can be connected together in ways that permit information to be stored. Several different gate arrangements are used to produce these flip-flops (abbreviated FF).

Figure 5-2(a) is the general type of symbol used for a flip-flop. It shows two outputs, labeled $Q$ and $\bar{Q}$, that are the inverse of each other. $Q / \bar{Q}$ are the most common designations used for a FF's outputs. From time to time, we will use other designations such as $X / \bar{X}$ and $A / \bar{A}$ for convenience in identifying different FFs in a logic circuit.

The $Q$ output is called the normal FF output, and $\bar{Q}$ is the inverted FF output. Whenever we refer to the state of a FF, we are referring to the state of its normal $(Q)$ output; it is understood that its inverted output $(\bar{Q})$ is in the opposite state. For example, if we say that a FF is in the HIGH (1) state, we mean that $Q=1$; if we say that a FF is in the LOW ( 0 ) state, we mean that $Q=0$. Of course, the $\bar{Q}$ state will always be the inverse of $Q$.

The two possible operating states for a FF are summarized in Figure 5-2(b). Note that the HIGH or 1 state ( $Q=1 / \bar{Q}=0$ ) is also referred to as the SET state. Whenever the inputs to a FF cause it to go to the $Q=1$ state, we call this setting the FF; the FF has been set. In a similar way, the LOW or
Output states

$$
Q=1, \bar{Q}=0:
$$

called HIGH or 1 state; also called SET state
$\underline{Q=0, \bar{Q}=1:} \begin{aligned} & \text { called LOW or } 0 \text { state; } \\ & \text { also called CLEAR or }\end{aligned}$ RESET state

FIGURE 5-2 General flip-flop symbol and definition of its two possible output states.

0 state ( $Q=0 / \bar{Q}=1$ ) is also referred to as the CLEAR or RESET state. Whenever the inputs to a FF cause it to go to the $Q=0$ state, we call this clearing or resetting the FF; the FF has been cleared (reset). As we shall see, many FFs will have a SET input and/or a CLEAR (RESET) input that is used to drive the FF into a specific output state.

As the symbol in Figure 5-2(a) implies, a FF can have one or more inputs. These inputs are used to cause the FF to switch back and forth ("flip-flop") between its possible output states. We will find out that most FF inputs need only to be momentarily activated (pulsed) in order to cause a change in the FF output state, and the output will remain in that new state even after the input pulse is over. This is the FF's memory characteristic.

The flip-flop is known by other names, including latch and bistable multivibrator. The term latch is used for certain types of flip-flops that we will describe. The term bistable multivibrator is the more technical name for a flip-flop, but it is too much of a mouthful to be used regularly.

## 5-1 NAND GATE LATCH

The most basic FF circuit can be constructed from either two NAND gates or two NOR gates. The NAND gate version, called a NAND gate latch or simply a latch, is shown in Figure 5-3(a). The two NAND gates are cross-coupled so that the output of NAND- 1 is connected to one of the inputs of NAND-2, and vice versa. The gate outputs, labeled $Q$ and $\bar{Q}$, respectively, are the latch outputs. Under normal conditions, these outputs will always be the inverse of each other. There are two latch inputs: the SET input is the input that sets $Q$ to the 1 state; the RESET input is the input that resets $Q$ to the 0 state.

The SET and RESET inputs are both normally resting in the HIGH state, and one of them will be pulsed LOW whenever we want to change the latch outputs. We begin our analysis by showing that there are two equally likely output states when SET $=$ RESET $=1$. One possibility is shown in Figure 53(a), where we have $Q=0$ and $\bar{Q}=1$. With $Q=0$, the inputs to NAND-2 are 0 and 1 , which produce $\bar{Q}=1$. The 1 from $\bar{Q}$ causes NAND- 1 to have a 1 at both inputs to produce a 0 output at $Q$. In effect, what we have is the LOW at the NAND-1 output producing a HIGH at the NAND-2 output, which, in turn, keeps the NAND-1 output LOW.

The second possibility is shown in Figure 5-3(b), where $Q=1$ and $\bar{Q}=0$. The HIGH from NAND-1 produces a LOW at the NAND-2 output, which, in turn, keeps the NAND-1 output HIGH. Thus, there are two possible output states when SET $=$ RESET $=1$; as we shall soon see, the one that actually exists will depend on what has occurred previously at the inputs.

FIGURE 5-3 A NAND latch has two possible resting states when SET $=$ RESET $=1$.

(a)

(b)

## Setting the Latch (FF)

Now let's investigate what happens when the SET input is momentarily pulsed LOW while RESET is kept HIGH. Figure 5-4(a) shows what happens when $Q=0$ prior to the occurrence of the pulse. As SET is pulsed LOW at time $t_{0}, Q$ will go HIGH, and this HIGH will force $\bar{Q}$ to go LOW so that NAND-1 now has two LOW inputs. Thus, when SET returns to the 1 state at $t_{1}$, the NAND-1 output remains HIGH, which, in turn, keeps the NAND-2 output LOW.


FIGURE 5-4 Pulsing the SET input to the 0 state when (a) $Q=0$ prior to SET pulse; (b) $Q=1$ prior to SET pulse. Note that, in both cases, $Q$ ends up HIGH.

Figure 5-4(b) shows what happens when $Q=1$ and $\bar{Q}=0$ prior to the application of the SET pulse. Since $\bar{Q}=0$ is already keeping the NAND-1 output HIGH, the LOW pulse at SET will not change anything. Thus, when SET returns HIGH, the latch outputs are still in the $Q=1, \bar{Q}=0$ state.

We can summarize Figure 5-4 by stating that a LOW pulse on the SET input will always cause the latch to end up in the $Q=1$ state. This operation is called setting the latch or FF.

## Resetting the Latch (FF)

Now let's consider what occurs when the RESET input is pulsed LOW while SET is kept HIGH. Figure 5-5(a) shows what happens when $Q=0$ and $\bar{Q}=1$


FIGURE 5-5 Pulsing the RESET input to the LOW state when (a) $Q=0$ prior to RESET pulse; (b) $Q=1$ prior to RESET pulse. In each case, $Q$ ends up LOW.
prior to the application of the pulse. Since $Q=0$ is already keeping the NAND-2 output HIGH, the LOW pulse at RESET will not have any effect. When RESET returns HIGH, the latch outputs are still $Q=0$ and $\bar{Q}=1$.

Figure 5-5(b) shows the situation where $Q=1$ prior to the occurrence of the RESET pulse. As RESET is pulsed LOW at $t_{0}, \bar{Q}$ will go HIGH, and this HIGH forces $Q$ to go LOW so that NAND-2 now has two LOW inputs. Thus, when RESET returns HIGH at $t_{1}$, the NAND-2 output remains HIGH, which, in turn, keeps the NAND-1 output LOW.

Figure 5-5 can be summarized by stating that a LOW pulse on the RESET input will always cause the latch to end up in the $Q=0$ state. This operation is called clearing or resetting the latch.

## Simultaneous Setting and Resetting

The last case to consider is the case where the SET and RESET inputs are simultaneously pulsed LOW. This will produce HIGH levels at both NAND outputs so that $Q=\bar{Q}=1$. Clearly, this is an undesired condition because the two outputs are supposed to be inverses of each other. Furthermore, when the SET and RESET inputs return HIGH, the resulting output state will depend on which input returns HIGH first. Simultaneous transitions back to the 1 state will produce unpredictable results. For these reasons the SET $=$ RESET $=0$ condition is normally not used for the NAND latch.

## Summary of NAND Latch

The operation described above can be conveniently placed in a function table (Figure 5-6) and is summarized as follows:

1. $\operatorname{SET}=$ RESET $=1$. This condition is the normal resting state, and it has no effect on the output state. The $Q$ and $\bar{Q}$ outputs will remain in whatever state they were in prior to this input condition.
2. $\operatorname{SET}=0$, RESET $=1$. This will always cause the output to go to the $Q=1$ state, where it will remain even after SET returns HIGH. This is called setting the latch.
3. $\operatorname{SET}=1, \operatorname{RESET}=0$. This will always produce the $Q=0$ state, where the output will remain even after RESET returns HIGH. This is called clearing or resetting the latch.
4. $\operatorname{SET}=$ RESET $=0$. This condition tries to set and clear the latch at the same time, and it produces $Q=\bar{Q}=1$. If the inputs are returned to 1 simultaneously, the resulting state is unpredictable. This input condition should not be used.

FIGURE 5-6 (a) NAND latch; (b) function table.



| Set | Reset | Output |
| :---: | :---: | :---: |
| 1 | 1 | No change |
| 0 | 1 | $\mathrm{Q}=1$ |
| 1 | 0 | $\mathrm{Q}=0$ |
| 0 | 0 | Invalid* |

(b)
(a)

FIGURE 5-7 (a) NAND latch equivalent representation; (b) simplified block symbol.

## Alternate Representations

From the description of the NAND latch operation, it should be clear that the SET and RESET inputs are active-LOW. The SET input will set $Q=1$ when SET goes LOW; the RESET input will clear $Q=0$ when RESET goes LOW. For this reason, the NAND latch is often drawn using the alternate representation for each NAND gate, as shown in Figure 5-7(a). The bubbles on the inputs, as well as the labeling of the signals as $\overline{S E T}$ and $\overline{R E S E T}$, indicate the active-LOW status of these inputs. (You may want to review Sections 3-13 and 3-14 on this topic.)

Figure 5-7(b) shows a simplified block representation that we will sometimes use. The $S$ and $R$ labels represent the SET and RESET inputs, and the bubbles indicate the active-LOW nature of these inputs. Whenever we use this block symbol, it represents a NAND latch.


## Terminology

The action of resetting a FF or a latch is also called clearing, and both terms are used interchangeably in the digital field. In fact, a RESET input can also be called a CLEAR input, and a SET-RESET latch can be called a SETCLEAR latch.

## EXAMPLE 5-1

The waveforms of Figure 5-8 are applied to the inputs of the latch of Figure $5-7$. Assume that initially $Q=0$, and determine the $Q$ waveform.

FIGURE 5-8 Example 5-1.


## Solution

Initially, $\overline{\mathrm{SET}}=\overline{\mathrm{RESET}}=1$ so that $Q$ will remain in the 0 state. The LOW pulse that occurs on the $\overline{\operatorname{RESET}}$ input at time $T_{1}$ will have no effect because $Q$ is already in the cleared (0) state.

The only way that $Q$ can go to the 1 state is by a LOW pulse on the $\overline{\mathrm{SET}}$ input. This occurs at time $T_{2}$ when $\overline{\mathrm{SET}}$ first goes LOW. When $\overline{\mathrm{SET}}$ returns HIGH at $T_{3}, Q$ will remain in its new HIGH state.

At time $T_{4}$ when $\overline{\mathrm{SET}}$ goes LOW again, there will be no effect on $Q$ because $Q$ is already set to the 1 state.

The only way to bring $Q$ back to the 0 state is by a LOW pulse on the $\overline{\mathrm{RESET}}$ input. This occurs at time $T_{5}$. When $\overline{\mathrm{RESET}}$ returns to 1 at time $T_{6}$, $Q$ remains in the LOW state.

Example $5-1$ shows that the latch output "remembers" the last input that was activated and will not change states until the opposite input is activated.

## EXAMPLE 5-2

FIGURE 5-9
(a) Mechanical contact bounce will produce multiple transitions; (b) NAND latch used to debounce a mechanical switch.


## Solution

Assume that the switch is resting in position 1 so that the RESET input is LOW and $Q=0$. When the switch is moved to position 2, RESET will go HIGH, and a LOW will appear on the SET input as the switch first makes contact. This will set $Q=1$ within a matter of a few nanoseconds (the response time of the NAND gate). Now if the switch bounces off contact 2, $\overline{\mathrm{SET}}$ and RESET will both be HIGH, and $Q$ will not be affected; it will stay HIGH. Thus, nothing will happen at $Q$ as the switch bounces on and off contact 2 before finally coming to rest in position 2.

Likewise, when the switch is moved from position 2 back to position 1, it will place a LOW on the RESET input as it first makes contact. This clears $Q$ to the LOW state, where it will remain even if the switch bounces on and off contact 1 several times before coming to rest.

Thus, the output at $Q$ will consist of a single transition each time the switch is moved from one position to the other.

## REVIEW QUESTIONS

1. What is the normal resting state of the $\overline{\text { SET }}$ and $\overline{\text { RESET inputs? What is }}$ the active state of each input?
2. What will be the states of $Q$ and $\bar{Q}$ after a FF has been reset (cleared)?
3. True or false: The $\overline{\mathrm{SET}}$ input can never be used to make $Q=0$.
4. When power is first applied to any FF circuit, it is impossible to predict the initial states of $Q$ and $\bar{Q}$. What can be done to ensure that a NAND latch always starts off in the $Q=1$ state?

## 5-2 NOR GATE LATCH

Two cross-coupled NOR gates can be used as a NOR gate latch. The arrangement, shown in Figure 5-10(a), is similar to the NAND latch except that the $Q$ and $\bar{Q}$ outputs have reversed positions.


FIGURE 5-10 (a) NOR gate latch; (b) function table; (c) simplified block symbol.

The analysis of the operation of the NOR latch can be performed in exactly the same manner as for the NAND latch. The results are given in the function table in Figure 5-10(b) and are summarized as follows:

1. $\operatorname{SET}=$ RESET $=0$. This is the normal resting state for the NOR latch, and it has no effect on the output state. $Q$ and $\bar{Q}$ will remain in whatever state they were in prior to the occurrence of this input condition.
2. $\operatorname{SET}=1, \operatorname{RESET}=0$. This will always set $Q=1$, where it will remain even after SET returns to 0 .
3. $\operatorname{SET}=0, \operatorname{RESET}=1$. This will always clear $Q=0$, where it will remain even after RESET returns to 0 .
4. $\operatorname{SET}=1, \operatorname{RESET}=1$. This condition tries to set and reset the latch at the same time, and it produces $Q=\bar{Q}=0$. If the inputs are returned to 0 simultaneously, the resulting output state is unpredictable. This input condition should not be used.

The NOR gate latch operates exactly like the NAND latch except that the SET and RESET inputs are active-HIGH rather than active-LOW, and the normal resting state is SET $=$ RESET $=0 . Q$ will be set HIGH by a HIGH pulse on the SET input, and it will be cleared LOW by a HIGH pulse on the RESET input. The simplified block symbol for the NOR latch in Figure 5$10(\mathrm{c})$ is shown with no bubbles on the $S$ and $R$ inputs; this indicates that these inputs are active-HIGH.

## EXAMPLE 5-3

FIGURE 5-11 Example 5-3.


## Solution

Initially, SET $=$ RESET $=0$, which has no effect on $Q$, and $Q$ stays LOW. When SET goes HIGH at time $T_{1}, Q$ will be set to 1 and will remain there even after SET returns to 0 at $T_{2}$.

At $T_{3}$ the RESET input goes HIGH and clears $Q$ to the 0 state, where it remains even after RESET returns LOW at $T_{4}$.

The RESET pulse at $T_{5}$ has no effect on $Q$ because $Q$ is already LOW. The SET pulse at $T_{6}$ again sets $Q$ back to 1 , where it will stay.

Example 5-3 shows that the latch "remembers" the last input that was activated, and it will not change states until the opposite input is activated.

## EXAMPLE 5-4

Figure $5-12$ shows a simple circuit that can be used to detect the interruption of a light beam. The light is focused on a phototransistor that is connected in the common-emitter configuration to operate as a switch. Assume that the latch has previously been cleared to the 0 state by momentarily opening switch SW1, and describe what happens if the light beam is momentarily interrupted.

FIGURE 5-12
Example 5-4.


## Solution

With light on the phototransistor, we can assume that it is fully conducting so that the resistance between the collector and the emitter is very small. Thus, $v_{0}$ will be close to 0 V . This places a LOW on the SET input of the latch so that SET $=$ RESET $=0$.

When the light beam is interrupted, the phototransistor turns off, and its collector-emitter resistance becomes very high (i.e., essentially an open circuit). This causes $v_{0}$ to rise to approximately 5 V ; this activates the SET input, which sets $Q$ HIGH and turns on the alarm.
$Q$ will remain HIGH and the alarm will remain on even if $v_{0}$ returns to 0 V (i.e., the light beam was interrupted only momentarily) because SET and RESET will both be LOW, which will produce no change in $Q$.

In this application, the latch's memory characteristic is used to convert a momentary occurrence (beam interruption) into a constant output.

## Flip-Flop State on Power-Up

When power is applied to a circuit, it is not possible to predict the starting state of a flip-flop's output if its SET and RESET inputs are in their inactive state (e.g., $S=R=1$ for a NAND latch, $S=R=0$ for a NOR latch). There is just as much chance that the starting state will be $Q=0$ as $Q=1$. It will depend on factors such as internal propagation delays, parasitic capacitance, and external loading. If a latch or FF must start off in a particular state to ensure the proper operation of a circuit, then it must be placed in that state by momentarily activating the SET or RESET input at the start of the circuit's operation. This is often achieved by application of a pulse to the appropriate input.

1. What is the normal resting state of the NOR latch inputs? What is the ac- tive state?
2. When a latch is set, what are the states of $Q$ and $\bar{Q}$ ?
3. What is the only way to cause the $Q$ output of a NOR latch to change from 1 to 0 ?
4. If the NOR latch in Figure $5-12$ were replaced by a NAND latch, why wouldn't the circuit work properly?

## 5-3 TROUBLESHOOTING CASE STUDY

The following two examples present an illustration of the kinds of reasoning used in troubleshooting a circuit containing a latch.

## EXAMPLE 5-5

Analyze and describe the operation of the circuit in Figure 5-13.


| Switch <br> position | $\mathrm{X}_{\mathrm{A}}$ | $\mathrm{X}_{\mathrm{B}}$ |
| :---: | :---: | :---: |
| A | Pulses | LOW |
| B | LOW | Pulses |

FIGURE 5-13 Examples 5-5 and 5-6.

## Solution

The switch is used to set or clear the NAND latch to produce clean, bouncefree signals at $Q$ and $\bar{Q}$. These latch outputs control the passage of the $1-\mathrm{kHz}$ pulse signal through to the AND outputs $X_{A}$ and $X_{B}$.

When the switch moves to position $A$, the latch is set to $Q=1$. This enables the $1-\mathrm{kHz}$ pulses to pass through to $X_{A}$, while the LOW at $\bar{Q}$ keeps $X_{B}=0$. When the switch moves to position $B$, the latch is cleared to $Q=0$, which keeps $X_{A}=0$, while the HIGH at $\bar{Q}$ enables the pulses to pass through to $X_{B}$.

## EXAMPLE 5-6

A technician tests the circuit of Figure 5-13 and records the observations shown in Table 5-1. He notices that when the switch is in position $B$, the circuit functions correctly, but in position $A$ the latch does not set to the $Q=1$ state. What are the possible faults that could produce this malfunction?

## Solution

There are several possibilities:

1. An internal open connection at $\mathrm{Z} 1-1$, which would prevent $Q$ from responding to the SET input.

TABLE 5-1

| Switch <br> Position | SET <br> $(Z 1-1)$ | RESET <br> $(Z 1-5)$ | $Q$ <br> $(Z 1-3)$ | $\bar{Q}$ <br> $(Z 1-6)$ | $X_{A}$ <br> $(Z 2-3)$ | $X_{B}$ <br> $(Z 2-6)$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $A$ | LOW | HIGH | LOW | HIGH | LOW | Pulses |
| $B$ | HIGH | LOW | LOW | HIGH | LOW | Pulses |

2. An internal component failure in NAND gate Z 1 that prevents it from responding properly.
3. The $Q$ output is stuck LOW, which could be caused by:
(a) Z1-3 internally shorted to ground
(b) Z1-4 internally shorted to ground
(c) Z2-2 internally shorted to ground
(d) The $Q$ node externally shorted to ground

An ohmmeter check from $Q$ to ground will determine if any of these conditions are present. A visual check should reveal any external short.

What about $\bar{Q}$ internally or externally shorted to $V_{C C}$ ? A little thought will lead to the conclusion that this could not be the fault. If $\bar{Q}$ were shorted to $V_{C C}$, this would not prevent the $Q$ output from going HIGH when $\overline{\text { SET }}$ goes LOW. Because $Q$ does not go HIGH, this cannot be the fault. The reason that $\bar{Q}$ looks as if it is stuck HIGH is that $Q$ is stuck LOW, and that keeps $\bar{Q}$ HIGH through the bottom NAND gate.

## 5-4 DIGITAL PULSES

As you can see from our discussion of SR latches, there are situations in digital systems when a signal switches from a normal inactive state to the opposite (active) state, thus causing something to happen in the circuit. Then the signal returns to its inactive state while the effect of the recently activated signal remains in the system. These signals are called pulses, and it is very important to understand the terminology associated with pulses and pulse waveforms. A pulse that performs its intended function when it goes HIGH is called a positive pulse, and a pulse that performs its intended function when it goes LOW is called a negative pulse. In actual circuits it takes time for a pulse waveform to change from one level to the other. These transition times are called the rise time ( $t_{\mathrm{r}}$ ) and the fall time ( $t_{\mathrm{f}}$ ) and are defined as the time it takes the voltage to change between $10 \%$ and $90 \%$ of the HIGH level voltage as shown on the positive pulse in Figure 5-14(a). The transition at the beginning of the pulse is called the leading edge and the transition at the end of the pulse is the trailing edge. The duration (width) of the pulse ( $t_{\mathrm{w}}$ ) is defined as the time between the points when the leading and trailing edges are at $50 \%$ of the HIGH level voltage. Figure 5-14(b) shows an active-LOW or negative pulse.

FIGURE 5-14 (a) A positive pulse and (b) a negative pulse.

(b)

## EXAMPLE 5-7

FIGURE 5-15
Example 5-7.

When a microcontroller wants to access data in its external memory, it activates an active-LOW output pin called $\overline{\mathrm{RD}}$ (read). The data book says that the $\overline{\mathrm{RD}}$ pulse typically has a pulse width $t_{\mathrm{w}}$ of 50 ns , a rise time $t_{\mathrm{r}}$ of 15 ns , and a fall time $t_{\mathrm{f}}$ of 10 ns . Draw a scaled drawing of the $\overline{\mathrm{RD}}$ pulse.

## Solution

Figure 5-15 shows the drawing of the pulse. The $\overline{\mathrm{RD}}$ pulse is active-LOW, so the leading edge is a falling edge measured by $t_{\mathrm{f}}$ and the trailing edge is the rising edge measured by $t_{\mathrm{r}}$.


## 5-5 CLOCK SIGNALS AND CLOCKED FLIP-FLOPS

Digital systems can operate either asynchronously or synchronously. In asynchronous systems, the outputs of logic circuits can change state any time one or more of the inputs change. An asynchronous system is generally more difficult to design and troubleshoot than a synchronous system.

In synchronous systems, the exact times at which any output can change states are determined by a signal commonly called the clock. This clock signal is generally a rectangular pulse train or a square wave, as shown in Figure 5-16. The clock signal is distributed to all parts of the system, and

FIGURE 5-16 Clock signals.


The speed at which a synchronous digital system operates is dependent on how often the clock cycles occur. A clock cycle is measured from one PGT to the next PGT or from one NGT to the next NGT. The time it takes to complete one cycle (seconds/cycle) is called the period (T), as shown in Figure 5-16(b). The speed of a digital system is normally referred to by the number of clock cycles that happen in 1 s (cycles/second), which is known as the frequency ( $\mathbf{F}$ ) of the clock. The standard unit for frequency is hertz. One hertz $(1 \mathrm{~Hz})=1$ cycle/second.

## Clocked Flip-Flops

Several types of clocked FFs are used in a wide range of applications. Before we begin our study of the different clocked FFs, we will describe the principal ideas that are common to all of them.

1. Clocked FFs have a clock input that is typically labeled CLK, $C K$, or $C P$. We will normally use CLK, as shown in Figure 5-17. In most clocked FFs, the CLK input is edge-triggered, which means that it is activated by a signal transition; this is indicated by the presence of a small triangle on the CLK input. This contrasts with the latches, which are level-triggered.

Figure 5-17(a) is a FF with a small triangle on its CLK input to indicate that this input is activated only when a positive-going transition (PGT) occurs; no other part of the input pulse will have an effect on the CLK input. In Figure 5-17(b), the FF symbol has a bubble as well as a triangle on its CLK input. This signifies that the CLK input is activated only when a negative-going transition occurs; no other part of the input pulse will have an effect on the CLK input.

FIGURE 5-17 Clocked FFs have a clock input (CLK) that is active on either (a) the PGT or (b) the NGT. The control inputs determine the effect of the active clock transition.


2. Clocked FFs also have one or more control inputs that can have various names, depending on their operation. The control inputs will have no effect on $Q$ until the active clock transition occurs. In other words, their effect is synchronized with the signal applied to CLK. For this reason they are called synchronous control inputs.

For example, the control inputs of the FF in Figure 5-17(a) will have no effect on $Q$ until the PGT of the clock signal occurs. Likewise, the control inputs in Figure 5-17(b) will have no effect until the NGT of the clock signal occurs.
3. In summary, we can say that the control inputs get the FF outputs ready to change, while the active transition at the CLK input actually triggers the change. The control inputs control the WHAT (i.e., what state the output will go to); the CLK input determines the WHEN.

## Setup and Hold Times

Two timing requirements must be met if a clocked FF is to respond reliably to its control inputs when the active CLK transition occurs. These requirements are illustrated in Figure 5-18 for a FF that triggers on a PGT.

The setup time, $\boldsymbol{t}_{\mathbf{S}}$, is the time interval immediately preceding the active transition of the CLK signal during which the control input must be maintained at the proper level. IC manufacturers usually specify the minimum allowable setup time $t_{\mathrm{S}}(\mathrm{min})$. If this time requirement is not met, the FF may not respond reliably when the clock edge occurs.

The hold time, $t_{\mathbf{H}}$, is the time interval immediately following the active transition of the CLK signal during which the synchronous control input must be maintained at the proper level. IC manufacturers usually specify the

FIGURE 5-18 Control inputs must be held stable for (a) a time $t_{\text {S }}$ prior to active clock transition and for (b) a time $t_{\mathrm{H}}$ after the active block transition.

(a)

(b)
minimum acceptable value of hold time $t_{H}(\mathrm{~min})$. If this requirement is not met, the FF will not trigger reliably.

Thus, to ensure that a clocked FF will respond properly when the active clock transition occurs, the control inputs must be stable (unchanging) for at least a time interval equal to $t_{\mathrm{S}}(\mathrm{min})$ prior to the clock transition, and for at least a time interval equal to $t_{\mathrm{H}}(\mathrm{min})$ after the clock transition.

IC flip-flops will have minimum allowable $t_{\mathrm{S}}$ and $t_{\mathrm{H}}$ values in the nanosecond range. Setup times are usually in the range of 5 to 50 ns , whereas hold times are generally from 0 to 10 ns . Notice that these times are measured between the 50 percent points on the transitions.

These timing requirements are very important in synchronous systems because, as we shall see, there will be many situations where the synchronous control inputs to a FF are changing at approximately the same time as the CLK input.

1. What two types of inputs does a clocked FF have?
2. What is meant by the term edge-triggered?
3. True or false: The CLK input will affect the FF output only when the active transition of the control input occurs.
4. Define the setup time and hold time requirements of a clocked FF.

## 5-6 CLOCKED S-R FLIP-FLOP

Figure 5-19(a) shows the logic symbol for a clocked S-R flip-flop that is triggered by the positive-going edge of the clock signal. This means that the FF can change states only when a signal applied to its clock input makes a transition from 0 to 1 . The $S$ and $R$ inputs control the state of the FF in the same manner as described earlier for the NOR gate latch, but the FF does not respond to these inputs until the occurrence of the PGT of the clock signal.

The function table in Figure 5 -19(b) shows how the FF output will respond to the PGT at the CLK input for the various combinations of $S$ and $R$ inputs. This function table uses some new nomenclature. The up arrow ( $\uparrow$ ) indicates that a PGT is required at $C L K$; the label $Q_{0}$ indicates the level at $Q$ prior to the PGT. This nomenclature is often used by IC manufacturers in their IC data manuals.

The waveforms in Figure 5-19(c) illustrate the operation of the clocked S-R flip-flop. If we assume that the setup and hold time requirements are being met in all cases, we can analyze these waveforms as follows:

1. Initially all inputs are 0 and the $Q$ output is assumed to be 0 ; that is, $Q_{0}=0$.
2. When the PGT of the first clock pulse occurs (point $a$ ), the $S$ and $R$ inputs are both 0 , so the FF is not affected and remains in the $Q=0$ state (i.e., $Q=Q_{0}$ ).
3. At the occurrence of the PGT of the second clock pulse (point $c$ ), the $S$ input is now high, with $R$ still low. Thus, the FF sets to the 1 state at the rising edge of this clock pulse.
4. When the third clock pulse makes its positive transition (point $e$ ), it finds that $S=0$ and $R=1$, which causes the FF to clear to the 0 state.

FIGURE 5-19 (a) Clocked S-R flip-flop that responds only to the positive-going edge of a clock pulse;
(b) function table;
(c) typical waveforms.

(a)

| Inputs |  |  | Output |
| :---: | :---: | :---: | :---: |
| S | R | CLK | Q |
| 0 | 0 | $\uparrow$ | $Q_{0}$ (no change) |
| 1 | 0 | $\uparrow$ | 1 |
| 0 | 1 | $\uparrow$ | 0 |
| 1 | 1 | $\uparrow$ | Ambiguous |
| $Q_{0}$ is output level prior to $\uparrow$ of CLK. <br> $\downarrow$ of CLK produces no change in Q . |  |  |  |

(b)

(c)
5. The fourth pulse sets the FF once again to the $Q=1$ state (point $g$ ) because $S=1$ and $R=0$ when the positive edge occurs.
6. The fifth pulse also finds that $S=1$ and $R=0$ when it makes its positivegoing transition. However, $Q$ is already high, so it remains in that state.
7. The $S=R=1$ condition should not be used because it results in an ambiguous condition.

It should be noted from these waveforms that the FF is not affected by the negative-going transitions of the clock pulses. Also, note that the $S$ and $R$ levels have no effect on the FF, except upon the occurrence of a positive-going transition of the clock signal. The $S$ and $R$ inputs are synchronous control inputs; they control which state the FF will go to when the clock pulse occurs. The CLK input is the trigger input that causes the FF to change states according to what the $S$ and $R$ inputs are when the active clock transition occurs.

Figure 5-20 shows the symbol and the function table for a clocked S-R flip-flop that triggers on the negative-going transition at its CLK input. The small circle and triangle on the CLK input indicates that this FF will trigger only when the CLK input goes from 1 to 0 . This FF operates in the same

FIGURE 5-20 Clocked S-R flip-flop that triggers only on negative-going transitions.

FIGURE 5-21 Simplified version of the internal circuitry for an edge-triggered S-R flip-flop.


| Inputs |  | Output |  |
| :---: | :---: | :---: | :--- |
| S | R | CLK | Q |
| 0 | 0 | $\downarrow$ | $\mathrm{Q}_{0}$ (no change) |
| 1 | 0 | $\downarrow$ | 1 |
| 0 | 1 | $\downarrow$ | 0 |
| 1 | 1 | $\downarrow$ | Ambiguous |

manner as the positive-edge FF except that the output can change states only on the falling edge of the clock pulses (points $b, d, f, h$, and $j$ in Figure $5-19)$. Both positive-edge and negative-edge triggering FFs are used in digital systems.

## Internal Circuitry of the Edge-Triggered S-R Flip-Flop

A detailed analysis of the internal circuitry of a clocked FF is not necessary because all types are readily available as ICs. Although our main interest is in the FF's external operation, our understanding of this external operation can be aided by taking a look at a simplified version of the FF's internal circuitry. Figure 5-21 shows this for an edge-triggered S-R flip-flop.

The circuit contains three sections:

1. A basic NAND gate latch formed by NAND-3 and NAND-4
2. A pulse-steering circuit formed by NAND-1 and NAND-2
3. An edge-detector circuit


As shown in Figure 5-21, the edge detector produces a narrow positivegoing spike $\left(C L K^{\star}\right)$ that occurs coincident with the active transition of the CLK input pulse. The pulse-steering circuit "steers" the spike through to the SET or the RESET input of the latch in accordance with the levels present at $S$ and $R$. For example, with $S=1$ and $R=0$, the $C L K^{\star}$ signal is inverted and passed through NAND-1 to produce a LOW pulse at the SET input of the latch that sets $Q=1$. With $S=0, R=1$, the $C L K^{\star}$ signal is inverted and passed through NAND-2 to produce a low pulse at the RESET input of the latch that resets $Q=0$.

Figure 5-22(a) shows how the $C L K^{\star}$ signal is generated for edge-triggered FFs that trigger on a PGT. The INVERTER produces a delay of a few nanoseconds so that the transitions of $\overline{C L K}$ occur a little bit after those of $C L K$. The AND


FIGURE 5-22 Implementation of edge-detector circuits used in edge-triggered flip-flops: (a) PGT; (b) NGT. The duration of the $C L K^{\star}$ pulses is typically $2-5 \mathrm{~ns}$.
gate produces an output spike that is HIGH only for the few nanoseconds when $C L K$ and $\overline{C L K}$ are both HIGH. The result is a narrow pulse at $C L K^{\star}$, which occurs on the PGT of CLK. The arrangement of Figure 5-22(b) likewise produces $C L K^{\star}$ on the NGT of $C L K$ for FFs that are to trigger on a NGT.

Because the $C L K^{\star}$ signal is HIGH for only a few nanoseconds, $Q$ is affected by the levels at $S$ and $R$ only for a short time during and after the occurrence of the active edge of $C L K$. This is what gives the FF its edgetriggered property.

1. Suppose that the waveforms of Figure 5-19(c) are applied to the inputs of the FF of Figure 5-20. What will happen to $Q$ at point $b$ ? At point $f$ ? At point $h$ ?
2. Explain why the $S$ and $R$ inputs affect $Q$ only during the active transition of CLK.

## 5-7 CLOCKED J-K FLIP-FLOP

Figure 5-23(a) shows a clocked J-K flip-flop that is triggered by the positivegoing edge of the clock signal. The $J$ and $K$ inputs control the state of the FF in the same ways as the $S$ and $R$ inputs do for the clocked S-R flip-flop except for one major difference: the $J=K=1$ condition does not result in an ambiguous output. For this 1,1 condition, the FF will always go to its opposite state upon the positive transition of the clock signal. This is called the toggle mode of operation. In this mode, if both $J$ and $K$ are left HIGH, the FF will change states (toggle) for each PGT of the clock.

The function table in Figure 5-23(a) summarizes how the J-K flip-flop responds to the PGT for each combination of $J$ and $K$. Notice that the function table is the same as for the clocked S-R flip-flop (Figure 5-19) except for the $J=K=1$ condition. This condition results in $Q=\bar{Q}_{0}$, which means that the new value of $Q$ will be the inverse of the value it had prior to the PGT; this is the toggle operation.


| $J$ | $K$ | $C L K$ | Q |
| :--- | :--- | :--- | :--- | :--- |
| 0 | 0 | $\uparrow$ | $Q_{0}$ (no change) |
| 1 | 0 | $\uparrow$ | 1 |
| 0 | 1 | $\uparrow$ | 0 |
| 1 | 1 | $\uparrow$ | $\bar{Q}_{0}$ (toggles) |

(a)


The operation of this FF is illustrated by the waveforms in Figure 523(b). Once again, we assume that the setup and hold time requirements are being met.

1. Initially all inputs are 0 , and the $Q$ output is assumed to be 1 ; that is, $Q_{0}=1$.
2. When the positive-going edge of the first clock pulse occurs (point $a$ ), the $J=0, K=1$ condition exists. Thus, the FF will be reset to the $Q=0$ state.
3. The second clock pulse finds $J=K=1$ when it makes its positive transition (point $c$ ). This causes the FF to toggle to its opposite state, $Q=1$.
4. At point $e$ on the clock waveform, $J$ and $K$ are both 0 , so that the FF does not change states on this transition.
5. At point $g, J=1$ and $K=0$. This is the condition that sets $Q$ to the 1 state. However, it is already 1 , and so it will remain there.
6. At point $i, J=K=1$, and so the FF toggles to its opposite state. The same thing occurs at point $k$.

Note from these waveforms that the FF is not affected by the negativegoing edge of the clock pulses. Also, the $J$ and $K$ input levels have no effect except upon the occurrence of the PGT of the clock signal. The $J$ and $K$ inputs by themselves cannot cause the FF to change states.

Figure 5-24 shows the symbol for a clocked J-K flip-flop that triggers on the negative-going clock-signal transitions. The small circle on the CLK input

FIGURE 5-24 J-K flip-flop that triggers only on negative-going transitions.



| $J$ | $K$ | CLK | Q |
| :---: | :---: | :---: | :--- |
| 0 | 0 | $\downarrow$ | $\mathrm{Q}_{0}$ (no change) |
| 1 | 0 | $\downarrow$ | 1 |
| 0 | 1 | $\downarrow$ |  |
| 1 | 1 | $\downarrow$ |  |

indicates that this FF will trigger when the $C L K$ input goes from 1 to 0 . This FF operates in the same manner as the positive-edge FF of Figure 5-23 except that the output can change states only on negative-going clock-signal transitions (points $b, d, f, h$, and $j$ ). Both polarities of edge-triggered J-K flipflops are in common usage.

The J-K flip-flop is much more versatile than the S-R flip-flop because it has no ambiguous states. The $J=K=1$ condition, which produces the toggling operation, finds extensive use in all types of binary counters. In essence, the J-K flip-flop can do anything the S-R flip-flop can do plus operate in the toggle mode.

## Internal Circuitry of the Edge-Triggered J-K Flip-Flop

A simplified version of the internal circuitry of an edge-triggered J-K flipflop is shown in Figure 5-25. It contains the same three sections as the edgetriggered S-R flip-flop (Figure 5-21). In fact, the only difference between the two circuits is that the $Q$ and $\bar{Q}$ outputs are fed back to the pulse-steering NAND gates. This feedback connection is what gives the J-K flip-flop its toggle operation for the $J=K=1$ condition.

FIGURE 5-25 Internal circuit of the edge-triggered J-K flip-flop.


Let's examine this toggle condition more closely by assuming that $J=K=1$ and that $Q$ is sitting in the LOW state when a CLK pulse occurs. With $Q=0$ and $\bar{Q}=1$, NAND gate 1 will steer $C L K^{\star}$ (inverted) to the $\overline{\mathrm{SET}}$ input of the NAND latch to produce $Q=1$. If we assume that $Q$ is HIGH when a $C L K$ pulse occurs, NAND gate 2 will steer CLK ${ }^{\star}$ (inverted) to the RESET input of the latch to produce $Q=0$. Thus, $Q$ always ends up in the opposite state.

In order for the toggle operation to work as described above, the CLK* pulse must be very narrow. It must return to 0 before the $Q$ and $\bar{Q}$ outputs toggle to their new values; otherwise, the new values of $Q$ and $\bar{Q}$ will cause the $C L K^{\star}$ pulse to toggle the latch outputs again.

1. True or false: A J-K flip-flop can be used as an S-R flip-flop, but an S-R flip-flop cannot be used as a J-K flip-flop.
2. Does a J-K flip-flop have any ambiguous input conditions?
3. What $J-K$ input condition will always set $Q$ upon the occurrence of the active CLK transition?

## 5-8 CLOCKED D FLIP-FLOP

Figure 5-26(a) shows the symbol and the function table for a clocked D flipflop that triggers on a PGT. Unlike the S-R and J-K flip-flops, this flip-flop has only one synchronous control input, $D$, which stands for data. The operation of the D flip-flop is very simple: $Q$ will go to the same state that is present on the $D$ input when a PGT occurs at $C L K$. In other words, the level present at $D$ will be stored in the flip-flop at the instant the PGT occurs. The waveforms in Figure 5-26(b) illustrate this operation.

Assume that $Q$ is initially HIGH. When the first PGT occurs at point $a$, the $D$ input is LOW; thus, $Q$ will go to the 0 state. Even though the $D$ input level changes between points $a$ and $b$, it has no effect on $Q ; Q$ is storing the LOW that was on $D$ at point $a$. When the PGT at $b$ occurs, $Q$ goes HIGH because $D$ is HIGH at that time. $Q$ stores this HIGH until the PGT at point $c$ causes $Q$ to go LOW because $D$ is LOW at that time. In a similar manner, the $Q$ output takes on the levels present at $D$ when the PGTs occur at points $d, e$, $f$, and $g$. Note that $Q$ stays HIGH at point $e$ because $D$ is still HIGH.

Again, it is important to remember that $Q$ can change only when a PGT occurs. The $D$ input has no effect between PGTs.

A negative-edge-triggered $D$ flip-flop operates in the same way just described except that $Q$ will take on the value of $D$ when a NGT occurs at $C L K$. The symbol for the D flip-flop that triggers on NGTs will have a bubble on the CLK input.

FIGURE 5-26 (a) D flipflop that triggers only on positive-going transitions; (b) waveforms.


| D | CLK | Q |
| :---: | :---: | :---: |
| 0 | $\uparrow$ | 0 |
| 1 | $\uparrow$ | 1 |

(a)

(b)

## Implementation of the D Flip-Flop

An edge-triggered D flip-flop is easily implemented by adding a single INVERTER to the edge-triggered J-K flip-flop, as shown in Figure 5-27. If you try both values of $D$, you should see that $Q$ takes on the level present at $D$ when a PGT occurs. The same can be done to convert a S-R flip-flop to a D flip-flop.

FIGURE 5-27 Edgetriggered D flip-flop implementation from a J-K flip-flop.


FIGURE 5-28 Parallel transfer of binary data using D flip-flops.


[^0]
## REVIEW QUESTIONS

1. What will happen to the $Q$ waveform in Figure $5-26(\mathrm{~b})$ if the $D$ input is held permanently LOW?
2. True or false: The $Q$ output will equal the level at the $D$ input at all times.
3. Can J-K FFs be used for parallel data transfer?

## 5-9 D LATCH (TRANSPARENT LATCH)

The edge-triggered D flip-flop uses an edge-detector circuit to ensure that the output will respond to the $D$ input only when the active transition of the clock occurs. If this edge detector is not used, the resultant circuit operates somewhat differently. It is called a $\boldsymbol{D}$ latch and has the arrangement shown in Figure 5-29(a).


FIGURE 5-29 $D$ latch: (a) structure; (b) function table; (c) logic symbol.

The circuit contains the NAND latch and the steering NAND gates 1 and 2 without the edge-detector circuit. The common input to the steering gates is called an enable input (abbreviated $E N$ ) rather than a clock input because its effect on the $Q$ and $\bar{Q}$ outputs is not restricted to occurring only on its transitions. The operation of the $D$ latch is described as follows:

1. When $E N$ is HIGH, the $D$ input will produce a LOW at either the $\overline{\mathrm{SET}}$ or the $\overline{\operatorname{RESET}}$ inputs of the NAND latch to cause $Q$ to become the same level as $D$. If $D$ changes while $E N$ is HIGH, $Q$ will follow the changes exactly. In other words, while $E N=1$, the $Q$ output will look exactly like $D$; in this mode, the $D$ latch is said to be "transparent."
2. When $E N$ goes LOW, the $D$ input is inhibited from affecting the NAND latch because the outputs of both steering gates will be held HIGH. Thus, the $Q$ and $\bar{Q}$ outputs will stay at whatever level they had just before $E N$ went LOW. In other words, the outputs are "latched" to their current level and cannot change while $E N$ is LOW even if $D$ changes.

This operation is summarized in the function table in Figure 5-29(b). The logic symbol for the $D$ latch is given in Figure 5-29(c). Note that even though the $E N$ input operates much like the $C L K$ input of an edge-triggered FF , there is no small triangle on the $E N$ input. This is because the small triangle symbol is used strictly for inputs that can cause an output change only when a transition occurs. The D latch is not edge-triggered.

## EXAMPLE 5-8

FIGURE 5-30 Waveforms for Example 5-8 showing the two modes of operation of the transparent $D$ latch.

Determine the $Q$ waveform for a $D$ latch with the $E N$ and $D$ inputs of Figure $5-30$. Assume that $Q=0$ initially.


## Solution

Prior to time $T_{1}, E N$ is LOW, so that $Q$ is "latched" at its current 0 level and cannot change even though $D$ is changing. During the interval $T_{1}$ to $T_{2}, E N$ is HIGH so that $Q$ will follow the signal present at $D$. Thus, $Q$ goes HIGH at $T_{1}$ and stays there because $D$ is not changing. When $E N$ returns LOW at $T_{2}, Q$ will latch at the HIGH level that it has at $T_{2}$ and will remain there while $E N$ is LOW.

At $T_{3}$ when $E N$ goes HIGH again, $Q$ will follow the changes in the $D$ input until $T_{4}$ when $E N$ returns LOW. During the interval $T_{3}$ to $T_{4}$, the $D$ latch is "transparent" because the variations in $D$ go through to the output $Q$. At $T_{4}$ when $E N$ goes LOW, $Q$ will latch at the 0 level because that is its level at $T_{4}$. After $T_{4}$ the variations in $D$ will have no effect on $Q$ because it is latched (i.e., $E N=0$ ).

1. Describe how a $D$ latch operates differently from an edge-triggered $D$ flip-flop.
2. True or false: A $D$ latch is in its transparent mode when $E N=0$.
3. True or false: In a $D$ latch, the $D$ input can affect $Q$ only when $E N=1$.

## 5-10 ASYNCHRONOUS INPUTS

For the clocked flip-flops that we have been studying, the $S, R, J, K$, and $D$ inputs have been referred to as control inputs. These inputs are also called synchronous inputs because their effect on the FF output is synchronized with the $C L K$ input. As we have seen, the synchronous control inputs must be used in conjunction with a clock signal to trigger the FF.

FIGURE 5-31 Clocked J-K flip-flop with asynchronous inputs.


Most clocked FFs also have one or more asynchronous inputs that operate independently of the synchronous inputs and clock input. These asynchronous inputs can be used to set the FF to the 1 state or clear (reset) the FF to the 0 state at any time, regardless of the conditions at the other inputs. Stated in another way, the asynchronous inputs are override inputs, which can be used to override all the other inputs in order to place the FF in one state or the other.

Figure $5-31$ shows a J-K flip-flop with two asynchronous inputs designated as $\overline{\text { PRESET }}$ and $\overline{\text { CLEAR }}$. These are active-LOW inputs, as indicated by the bubbles on the FF symbol. The accompanying function table summarizes how they affect the FF output. Let's examine the various cases.


| J | K | Clk | $\overline{\text { PRE }}$ | $\overline{\mathrm{CLR}}$ | Q |
| :---: | :---: | :---: | :---: | :---: | :--- |
| 0 | 0 | $\downarrow$ | 1 | 1 | Q (no change) |
| 0 | 1 | $\downarrow$ | 1 | 1 | 0 (Synch reset) |
| 1 | 0 | $\downarrow$ | 1 | 1 | 1 ( Synch set) |
| 1 | 1 | $\downarrow$ | 1 | 1 | $\overline{\mathrm{Q}}$ (Synch toggle) |
| x | x | x | 1 | 1 | Q (no change) |
| x | x | x | 1 | 0 | 0 (asynch clear) |
| x | x | x | 0 | 1 | 1 (asynch preset) |
| x | x | x | 0 | 0 | (Invalid) |

- $\overline{\text { PRESET }}=\overline{\text { CLEAR }}=1$. The asynchronous inputs are inactive and the FF is free to respond to the $J, K$, and CLK inputs; in other words, the clocked operation can take place.
■ $\overline{\mathrm{PRESET}}=0 ; \overline{\mathrm{CLEAR}}=1$. The $\overline{\text { PRESET }}$ is activated and $Q$ is immediately set to 1 no matter what conditions are present at the $J, K$, and $C L K$ inputs. The CLK input cannot affect the FF while $\overline{\text { PRESET }}=0$.
$\square \overline{\text { PRESET }}=1 ; \overline{\mathrm{CLEAR}}=0$. The $\overline{\text { CLEAR }}$ is activated and $Q$ is immediately cleared to 0 independent of the conditions on the $J, K$, or $C L K$ inputs. The $C L K$ input has no effect while $\overline{\text { CLEAR }}=0$.
■ $\overline{\mathrm{PRESET}}=\overline{\mathrm{CLEAR}}=0$. This condition should not be used because it can result in an ambiguous response.

It is important to realize that these asynchronous inputs respond to dc levels. This means that if a constant 0 is held on the $\overline{\text { PRESET input, the FF }}$ will remain in the $Q=1$ state regardless of what is occurring at the other inputs. Similarly, a constant LOW on the CLEAR input holds the FF in the $Q=0$ state. Thus, the asynchronous inputs can be used to hold the FF in a particular state for any desired interval. Most often, however, the asynchronous inputs are used to set or clear the FF to the desired state by application of a momentary pulse.

Many clocked FFs that are available as ICs will have both of these asynchronous inputs; some will have only the $\overline{\text { CLEAR }}$ input. Some FFs will have asynchronous inputs that are active-HIGH rather than active-LOW. For these FFs the FF symbol would not have a bubble on the asynchronous inputs.

## Designations for Asynchronous Inputs

IC manufacturers do not all agree on the nomenclature to use for these asynchronous inputs. The most common designations are PRE (short for PRESET) and CLR (short for CLEAR). These labels clearly distinguish them from the
synchronous SET and RESET inputs. Other labels such as $S_{D}$ (direct SET) and $R_{D}$ (direct RESET) are also used. From now on, we will use the labels PRE and $C L R$ to represent the asynchronous inputs because these seem to be the most commonly used labels. When these asynchronous inputs are active-LOW, as they generally are, we will use the overbar to indicate their active-LOW status, that is $\overline{P R E}$ and $\overline{C L R}$.

Although most IC flip-flops have at least one or more asynchronous inputs, there are some circuit applications where they are not used. In such cases they are held permanently at their inactive level. Often, in our use of FFs throughout the remainder of the text, we will not show a FF's unused asynchronous inputs; it will be assumed that they are permanently connected to their inactive logic level.

## EXAMPLE 5-9

Figure 5-32(a) shows the symbol for a J-K FF that responds to a NGT on its clock input and has active-LOW asynchronous inputs. Before proceeding with the example, take note of the way the inputs are labeled. First, note that the clock signal applied to the FF is labeled $\overline{C L K}$ (the overbar indicates that this signal is active on the NGT), whereas on the other side of the bubble (inside the block), it is labeled CLK. Likewise, the external active-LOW asynchronous


(a)

| Point | Operation |
| :---: | :--- |
| a | Synchronous toggle on NGT of $\overline{\text { CLK }}$ |
| b | Asynchronous set on $\overline{\text { PRE }}=0$ |
| c | Synchronous toggle |
| d | Synchronous toggle |
| e | Asynchronous clear on $\overline{C L R}=0$ |
| f | $\overline{\text { CLR overrides the NGT of } \overline{C L K}}$ |
| g | Synchronous toggle |

(b)

FIGURE 5-32 Waveforms for Example 5-9 showing how a clocked flip-flop responds to asynchronous inputs.
inputs are labeled $\overline{P R E}$ and $\overline{C L R}$, whereas inside the block on the other side of the bubble, they are labeled $P R E$ and $C L R$. The important thing to remember is that the presence of the bubble on an input means that the input responds to a logic LOW signal.

The $J$ and $K$ inputs are shown tied HIGH for this example. Determine the $Q$ output in response to the input waveforms shown in Figure 5-32(a). Assume that $Q$ is initially HIGH.

## Solution

Initially, $\overline{P R E}$ and $\overline{C L R}$ are in their inactive HIGH state, so that they will have no effect on $Q$. Thus, when the first NGT of the $\overline{C L K}$ signal occurs at point $a$, $Q$ will toggle to its opposite state; remember, $J=K=1$ produces the toggle operation.

At point $b$, the $\overline{P R E}$ input is pulsed to its active-LOW state. This will immediately set $Q=1$. Note that $\overline{P R E}$ produces $Q=1$ without waiting for a NGT at $\overline{C L K}$. The asynchronous inputs operate independently of $\overline{C L K}$.

At point $c$, the NGT of $\overline{C L K}$ will again cause $Q$ to toggle to its opposite state. Note that $\overline{P R E}$ has returned to its inactive state prior to point $c$. Likewise, the NGT of $\overline{C L K}$ at point $d$ will toggle $Q$ back HIGH.

At point $e$, the $\overline{C L R}$ input is pulsed to its active-LOW state and will immediately clear $Q=0$. Again, it does this independently of $\overline{C L K}$.

The NGT of $\overline{C L K}$ at point $f$ will not toggle $Q$ because the $\overline{C L R}$ input is still active. The LOW at $\overline{C L R}$ overrides the $\overline{C L K}$ input and holds $Q=0$.

When the NGT of $\overline{C L K}$ occurs at point $g$, it will toggle $Q$ to the HIGH state because neither asynchronous input is active at that point.

These steps are summarized in Figure 5-32(b).

1. How does the operation of an asynchronous input differ from that of a synchronous input?
2. Can a D flip-flop respond to its $D$ and $C L K$ inputs while $\overline{P R E}=1$ ?
3. List the conditions necessary for a positive-edge-triggered J-K flip-flop with active-LOW asynchronous inputs to toggle to its opposite state.

## 5-11 IEEE/ANSI SYMBOLS

Figure 5-33(a) shows the IEEE/ANSI symbol for a negative-edge-triggered J-K flip-flop with asynchronous inputs. Note the right triangle on the CLK input to indicate that it is activated by a NGT. Recall that in the IEEE/ANSI symbols, a right triangle has the same meanings as the small bubble in the traditional symbols. Also note that the clock input is labeled "C" inside the rectangle. IEEE/ANSI always uses a "C" to denote any input that controls when other inputs will affect the output. The $\overline{P R E}$ and $\overline{C L R}$ inputs are activeLOW, as indicated by the right triangles on these inputs. IEEE/ANSI also uses the labels " $S$ " and " $R$ " inside the rectangle to denote the asynchronous SET and RESET operations, which are the same as PRESET and CLEAR, respectively.

Figure 5-33(b) shows the IEEE/ANSI logic symbol for an IC that is part of the 74LS series of TTL devices. The 74LS112 is a dual negative-edge-triggered J-K flip-flop with preset and clear capabilities. It contains two J-K flip-flops,

FIGURE 5-33 IEEE/ANSI symbols for (a) a single edge-triggered J-K flip-flop and (b) an actual IC (74LS112 dual negative-edge-triggered J-K flipflop).

like the one symbolized in Figure 5-33(a). Note how the inputs and outputs are numbered. Also note that the input labels inside the rectangles are shown only for the top FF. It is understood that the inputs to the bottom FF are in the same arrangement as the top one. This same IC symbol applies to the CMOS 74HC112.

Figure 5-34(a) is the IEEE/ANSI symbol for a positive-edge-triggered D flip-flop with asynchronous inputs. There is no right triangle on the clock input because this FF is clocked by PGTs.

FIGURE 5-34 IEEE/ANSI symbols for (a) a single edge-triggered D flip-flop and (b) an actual IC (74HC175 quad flip-flop with common clock and clear).


Figure 5-34(b) is the IEEE/ANSI symbol for a 74 HC 175 IC , which contains four D flip-flops that share a common CLK input and a common $\overline{C L R}$ input. The FFs do not have a $\overline{P R E}$ input. This symbol contains a separate rectangle to represent each FF, and a special common-control block, which is the notched rectangle on top. The common-control block is used whenever an

IC has one or more inputs that are common to more than one of the circuits on the chip. For the 74 HC 175 , the $C L K$ and $\overline{C L R}$ inputs are common to all four of the D flip-flops on the IC. This means that a PGT on CLK will cause each $Q$ output to take on the level present at its $D$ input; it also means that a LOW on $\overline{C L R}$ will clear all $Q$ outputs to the LOW state.

## REVIEW QUESTIONS

1. Explain the meaning of the two different triangles that can be part of the IEEE/ANSI symbology at a clock input.
2. Describe the meaning of the common-control block.

## 5-12 FLIP-FLOP TIMING CONSIDERATIONS

Manufacturers of IC flip-flops will specify several important timing parameters and characteristics that must be considered before a FF is used in any circuit application. We will describe the most important of these and then give some actual examples of specific IC flip-flops from the TTL and CMOS logic families.

## Setup and Hold Times

The setup and hold times have already been discussed, and you may recall from Section 5-5 that they represent requirements that must be met for reliable FF triggering. The manufacturer's IC data sheet will always specify the minimum values of $t_{\mathrm{S}}$ and $t_{\mathrm{H}}$.

## Propagation Delays

Whenever a signal is to change the state of a FF's output, there is a delay from the time the signal is applied to the time when the output makes its change. Figure 5-35 illustrates the propagation delays that occur in response to a positive transition on the CLK input. Note that these delays are measured between the 50 percent points on the input and output waveforms. The same types of delays occur in response to signals on a FF's asynchronous inputs (PRESET and CLEAR). The manufacturers' data sheets usually specify propagation delays in response to all inputs, and they usually specify the maximum values for $t_{\text {PLH }}$ and $t_{\text {PHL }}$.

FIGURE 5-35 FF propagation delays.

(a)

(b)

Modern IC flip-flops have propagation delays that range from a few nanoseconds to around 100 ns . The values of $t_{\text {PLH }}$ and $t_{\text {PHL }}$ are generally not the same, and they increase in direct proportion to the number of loads being driven by the $Q$ output. FF propagation delays play an important part in certain situations that we will encounter later.

## Maximum Clocking Frequency, $f_{\text {MAX }}$

This is the highest frequency that may be applied to the CLK input of a FF and still have it trigger reliably. The $f_{\mathrm{MAX}}$ limit will vary from FF to FF , even with FFs having the same device number. For example, the manufacturer of the 7470 J -K flip-flop IC tests many of these FFs and may find that the $f_{\mathrm{MAX}}$ values fall in the range of 20 to 35 MHz . He will then specify the minimum $f_{\text {MAX }}$ as 20 MHz . This may seem confusing, but a little thought should make it clear that what the manufacturer is saying is that he cannot guarantee that the 7470 FF that you put in your circuit will work above 20 MHz ; most of them will, but some of them will not. If you operate them below 20 MHz , however, he guarantees that they will all work.

## Clock Pulse HIGH and LOW Times

The manufacturer will also specify the minimum time duration that the CLK signal must remain LOW before it goes HIGH, sometimes called $t_{\mathrm{W}}(L)$, and the minimum time that CLK must be kept HIGH before it returns LOW, sometimes called $t_{\mathrm{W}}(H)$. These times are defined in Figure 5-36(a). Failure to meet these minimum time requirements can result in unreliable triggering. Note that these time values are measured between the halfway points on the signal transitions.


FIGURE 5-36 (a) Clock LOW and HIGH times; (b) asynchronous pulse width.

## Asynchronous Active Pulse Width

The manufacturer will also specify the minimum time duration that a PRESET or CLEAR input must be kept in its active state in order to set or clear the FF reliably. Figure 5 -36(b) shows $t_{\mathrm{W}}(L)$ for active-LOW asynchronous inputs.

## Clock Transition Times

For reliable triggering, the clock waveform transition times (rise and fall times) should be kept very short. If the clock signal takes too long to make the transitions from one level to the other, the FF may trigger erratically or not at all. Manufacturers usually do not list a maximum transition time requirement for each FF integrated circuit. Instead, it is usually given as a general requirement for all ICs within a given logic family. For example, the transition times should generally be $\leq 50 \mathrm{~ns}$ for TTL devices and $\leq 200 \mathrm{~ns}$ for CMOS. These requirements will vary among the different manufacturers and among the various subfamilies within the broad TTL and CMOS logic families.

## Actual ICs

As practical examples of these timing parameters, let's take a look at several actual integrated-circuit FFs. In particular, we will look at the following ICs:

| 7474 | Dual edge-triggered D flip-flop (standard TTL) |
| :--- | :--- |
| 74 LS 112 | Dual edge-triggered J-K flip-flop (low-power Schottky TTL) |
| 74 C 74 | Dual edge-triggered D flip-flop (metal-gate CMOS) |
| 74 HC 112 | Dual edge-triggered J-K flip-flop (high-speed CMOS) |

Table 5-2 lists the various timing values for each of these FFs as they appear in the manufacturers' data books. All of the listed values are minimum values, except for the propagation delays, which are maximum values. Examination of Table 5-2 reveals two interesting points.

TABLE 5-2 Flip-flop timing values (in nanoseconds).

|  |  | TTL |  |  | CMOS |  |
| :--- | :--- | ---: | :---: | :---: | :---: | :---: |
|  |  | $\mathbf{7 4 7 4}$ | 74LS112 |  | 74C74 | 74HC112 |
| $t_{\mathrm{S}}$ |  | 50 | 20 |  | 60 | 25 |
| $t_{\mathrm{H}}$ |  | 0 |  | 0 | 0 |  |
| $t_{\mathrm{PHL}}$ | from CLK to $Q$ | 40 | 24 |  | 200 | 31 |
| $t_{\mathrm{PLH}}$ | from $C L K$ to $Q$ | 25 | 16 |  | 200 | 31 |
| $t_{\mathrm{PHL}}$ | from $\overline{C L R}$ to $Q$ | 40 | 24 |  | 225 | 41 |
| $t_{\mathrm{PLH}}$ | from $\overline{P R E}$ to $Q$ | 25 | 16 |  | 225 | 41 |
| $t_{\mathrm{W}}(L)$ | $C L K L O W$ time | 37 | 15 |  | 100 | 25 |
| $t_{\mathrm{W}}(H)$ | $C L K \mathrm{HIGH}$ time | 30 | 20 |  | 100 | 25 |
| $t_{\mathrm{W}}(L)$ | at $\overline{P R E}$ or $\overline{C L R}$ | 30 | 15 |  | 60 | 25 |
| $t_{\mathrm{MAX}}$ | in MHz | 15 | 30 |  | 5 | 20 |

1. All of the FFs have very low $t_{\mathrm{H}}$ requirements; this is typical of most modern edge-triggered FFs.
2. The 74 HC series of CMOS devices has timing values that are comparable to those of the TTL devices. The 74C series is much slower than the 74HC series.

## EXAMPLE 5-10

From Table 5-2 determine the following.
(a) Assume that $Q=0$. How long can it take for $Q$ to go HIGH when a PGT occurs at the CLK input of a 7474?
(b) Assume that $Q=1$. How long can it take for $Q$ to go LOW in response to the $\overline{C L R}$ input of a 74 HC 112 ?
(c) What is the narrowest pulse that should be applied to the $\overline{C L R}$ input of the 74 LS 112 FF to clear $Q$ reliably?
(d) Which FF in Table 5-2 requires that the control inputs remain stable after the occurrence of the active clock transition?
(e) For which FFs must the control inputs be held stable for a minimum time prior to the active clock transition?

## REVIEW QUESTIONS

FIGURE 5-37 $Q_{2}$ will respond properly to the level present at $Q_{1}$ prior to the present at $Q_{1}$ prior to the
NGT of $C L K$, provided that $Q_{2}$ 's hold time requirement, $t_{\mathrm{H}}$, is less than $Q_{1}$ 's propagation delay.

## Solution

(a) The PGT will cause $Q$ to go from LOW to HIGH. The delay from CLK to $Q$ is listed as $t_{\mathrm{PLH}}=25 \mathrm{~ns}$ for the 7474 .
(b) For the 74 HC 112 , the time required for $Q$ to go from HIGH to LOW in response to the $\overline{C L R}$ input is listed as $t_{\mathrm{PHL}}=41 \mathrm{~ns}$.
(c) For the 74 LS 112 , the narrowest pulse at the $\overline{C L R}$ input is listed as $t_{\mathrm{W}}(L)=15 \mathrm{~ns}$.
(d) The 7474 is the only FF in Table 5-2 that has a nonzero hold time requirement.
(e) All of the FFs have a nonzero setup time requirement.

1. Which FF timing parameters indicate the time it takes the $Q$ output to respond to an input?
2. True or false: A FF that has an $f_{\mathrm{MAX}}$ rating of 25 MHz can be reliably triggered by any CLK pulse waveform with a frequency below 25 MHz .

## 5-13 POTENTIAL TIMING PROBLEM IN FF CIRCUITS

In many digital circuits, the output of one FF is connected either directly or through logic gates to the input of another FF, and both FFs are triggered by the same clock signal. This presents a potential timing problem. A typical situation is illustrated in Figure 5-37, where the output of $Q_{1}$ is connected to the $J$ input of $Q_{2}$ and both FFs are clocked by the same signal at their $C L K$ inputs.


The potential timing problem is this: because $Q_{1}$ will change on the NGT of the clock pulse, the $J_{2}$ input of $Q_{2}$ will be changing as it receives the same NGT. This could lead to an unpredictable response at $Q_{2}$.

Let's assume that initially $Q_{1}=1$ and $Q_{2}=0$. Thus, the $Q_{1} \mathrm{FF}$ has $J_{1}=K_{1}=1$, and $Q_{2}$ has $J_{2}=Q_{1}=1, K_{2}=0$ prior to the NGT of the clock pulse. When the NGT occurs, $Q_{1}$ will toggle to the LOW state, but it will not actually go LOW until after its propagation delay, $t_{\text {PHL }}$. The same NGT will reliably clock $Q_{2}$ to the HIGH state provided that $t_{\text {PHL }}$ is greater than $Q_{2}$ 's hold time requirement, $t_{\mathrm{H}}$. If this condition is not met, the response of $Q_{2}$ will be unpredictable.

Fortunately, all modern edge-triggered FFs have hold time requirements that are 5 ns or less; most have $t_{\mathrm{H}}=0$, which means that they have no hold time requirement. For these FFs, situations like that in Figure 5-37 will not be a problem.

Unless stated otherwise, in all of the FF circuits that we encounter throughout the text, we will assume that the FF's hold time requirement is short enough to respond reliably according to the following rule:

The FF output will go to a state determined by the logic levels present at its synchronous control inputs just prior to the active clock transition.

If we apply this rule to Figure $5-37$, it says that $Q_{2}$ will go to a state determined by the $J_{2}=1, K_{2}=0$ condition that is present just prior to the NGT of the clock pulse. The fact that $J_{2}$ is changing in response to the same NGT has no effect.

## EXAMPLE 5-11

Determine the $Q$ output for a negative-edge-triggered J-K flip-flop for the input waveforms shown in Figure 5-38. Assume that $t_{\mathrm{H}}=0$ and that $Q=0$ initially.

FIGURE 5-38 Example 5-11.


## Solution

The FF will respond only at times $T_{2}, T_{4}, T_{6}$, and $T_{8}$. At $T_{2}, Q$ will respond to the $J=K=0$ condition present just prior to $T_{2}$. At $T_{4}, Q$ will respond to the $J=1$, $K=0$ condition present just prior to $T_{4}$. At $T_{6}, Q$ will respond to the $J=0$, $K=1$ condition present just prior to $T_{6}$. At $T_{8}, Q$ responds to $J=K=1$.

## 5-14 FLIP-FLOP APPLICATIONS

Edge-triggered (clocked) flip-flops are versatile devices that can be used in a wide variety of applications including counting, storing of binary data, transferring binary data from one location to another, and many more. Almost all of these applications utilize the FF's clocked operation. Many of them fall into the category of sequential circuits. A sequential circuit is one in which the outputs follow a predetermined sequence of states, with a new state occurring each time a clock pulse occurs. We will introduce some of the basic applications in the following sections, and we will expand on them in subsequent chapters.

## 5-15 FLIP-FLOP SYNCHRONIZATION

Most digital systems are principally synchronous in their operation because most of the signals will change states in synchronism with the clock transitions. In many cases, however, there will be an external signal that is not synchronized to the clock; in other words, it is asynchronous. Asynchronous signals often occur as a result of a human operator's actuating an input switch at some random time relative to the clock signal. This randomness can produce unpredictable and undesirable results. The following example illustrates how a FF can be used to synchronize the effect of an asynchronous input.

## EXAMPLE 5-12

Figure 5-39(a) shows a situation where input signal $A$ is generated from a debounced switch that is actuated by an operator (a debounced switch was first introduced in Example 5-2). $A$ goes HIGH when the operator actuates the switch and goes LOW when the operator releases the switch. This $A$ input is used to control the passage of the clock signal through the AND gate so that clock pulses appear at output $X$ only as long as $A$ is HIGH.


FIGURE 5-39 Asynchronous signal $A$ can produce partial pulses at $X$.
The problem with this circuit is that $A$ is asynchronous; it can change states at any time relative to the clock signal because the exact times when the operator actuates or releases the switch are essentially random. This can produce partial clock pulses at output $X$ if either transition of $A$ occurs while the clock signal is HIGH, as shown in the waveforms of Figure 5-39(b).

This type of output is often not acceptable, so a method for preventing the appearance of partial pulses at $X$ must be developed. One solution is shown in Figure 5-40(a). Describe how this circuit solves the problem, and draw the $X$ waveform for the same situation as in Figure 5-39(b).

FIGURE 5-40 An edgetriggered D flip-flop is used to synchronize the enabling of the AND gate to the NGTs of the clock.


(a)

(b)

## Solution

The $A$ signal is connected to the $D$ input of FF $Q$, which is clocked by the NGT of the clock signal. Thus, when $A$ goes HIGH, $Q$ will not go HIGH until the next NGT of the clock at time $T_{1}$. This HIGH at $Q$ will enable the AND gate to pass subsequent complete clock pulses to $X$, as shown in Figure 5-40(b).

When $A$ returns LOW, $Q$ will not go LOW until the next NGT of the clock at $T_{2}$. Thus, the AND gate will not inhibit clock pulses until the clock pulse that ends at $T_{2}$ has been passed through to $X$. Therefore, output $X$ contains only complete pulses.

There is a potential problem with this circuit. Since $A$ could go HIGH at any moment, it may by random chance violate the setup time requirement of the flip-flop. In other words, the transition of $A$ may occur so close to the clock edge that it causes an unstable response (glitch) from the $Q$ output. Preventing this would require a more complex synchronizing circuit.

## 5-16 DETECTING AN INPUT SEQUENCE

In many situations, an output is to be activated only when the inputs are activated in a certain sequence. This cannot be accomplished using pure combinational logic but requires the storage characteristic of FFs.

For example, an AND gate can be used to determine when two inputs $A$ and $B$ are both HIGH, but its output will respond the same regardless of which input goes HIGH first. But suppose that we want to generate a HIGH output only if $A$ goes HIGH and then $B$ goes HIGH some time later. One way to accomplish this is shown in Figure 5-41(a).

The waveforms in Figure 5-41(b) and (c) show that $Q$ will go HIGH only if $A$ goes HIGH before $B$ goes HIGH. This is because $A$ must be HIGH in order for $Q$ to go HIGH on the PGT of $B$.

FIGURE 5-41 Clocked D flip-flop used to respond to a particular sequence of inputs.


In order for this circuit to work properly, $A$ must go HIGH prior to $B$ by at least an amount of time equal to the setup time requirement of the FF.

## 5-17 DATA STORAGE AND TRANSFER

By far the most common use of flip-flops is for the storage of data or information. The data may represent numerical values (e.g., binary numbers, BCD-coded decimal numbers) or any of a wide variety of types of data that have been encoded in binary. These data are generally stored in groups of FFs called registers.

The operation most often performed on data that are stored in a FF or a register is the data transfer operation. This involves the transfer of data from one FF or register to another. Figure 5-42 illustrates how data transfer can be accomplished between two FFs using clocked S-R, J-K, and D flip-flops. In each case, the logic value that is currently stored in FF $A$ is transferred to FF $B$ upon the NGT of the TRANSFER pulse. Thus, after this NGT, the $B$ output will be the same as the $A$ output.

The transfer operations in Figure 5-42 are examples of synchronous transfer because the synchronous control and CLK inputs are used to perform the transfer. A transfer operation can also be obtained using the asynchronous inputs of a FF. Figure 5-43 shows how an asynchronous transfer can be accomplished using the PRESET and CLEAR inputs of any type of FF.


FIGURE 5-42 Synchronous data transfer operation performed by various types of clocked FFs.

FIGURE 5-43
Asynchronous data transfer operation.

FIGURE 5-44 Parallel transfer of contents of reg. ister $X$ into register Y.


Here, the asynchronous inputs respond to LOW levels. When the TRANSFER ENABLE line is held LOW, the two NAND outputs are kept HIGH, with no effect on the FF outputs. When the TRANSFER ENABLE line is made HIGH, one of the NAND outputs will go LOW, depending on the state of the $A$ and $\bar{A}$ outputs. This LOW will either set or clear FF $B$ to the same state as FF $A$. This asynchronous transfer is done independently of the synchronous and CLK inputs of the FF. Asynchronous transfer is also called jam transfer because the data can be "jammed" into FF B even if its synchronous inputs are active.

## Parallel Data Transfer

Figure 5-44 illustrates data transfer from one register to another using D-type FFs. Register $X$ consists of FFs $X_{2}, X_{1}$, and $X_{0}$; register $Y$ consists of FFs $Y_{2}$, $Y_{1}$, and $Y_{0}$. Upon application of the PGT of the TRANSFER pulse, the level stored in $X_{2}$ is transferred to $Y_{2}, X_{1}$ to $Y_{1}$, and $X_{0}$ to $Y_{0}$. The transfer of the

contents of the $X$ register into the $Y$ register is a synchronous transfer. It is also referred to as a parallel transfer because the contents of $X_{2}, X_{1}$, and $X_{0}$ are transferred simultaneously into $\mathrm{Y}_{2}, \mathrm{Y}_{1}$, and $\mathrm{Y}_{0}$, respectively. If a serial data transfer were performed, the contents of the $X$ register would be transferred to the Y register one bit at a time. This will be examined in the next section.

It is important to understand that parallel transfer does not change the contents of the register that is the source of data. For example, in Figure 5-44, if $X_{2} X_{1} X_{0}=101$ and $Y_{2} Y_{1} Y_{0}=011$ prior to the occurrence of the TRANSFER pulse, then both registers will be holding 101 after the TRANSFER pulse.

1. True or false: Asynchronous data transfer uses the CLK input.
2. Which type of FF is best suited for synchronous transfer because it requires the fewest interconnections from one FF to the other?
3. If J-K flip-flops were used in the registers of Figure 5-44, how many total interconnections would be required from register $X$ to register $Y$ ?
4. True or false: Synchronous data transfer requires less circuitry than asynchronous transfer.

## 5-18 SERIAL DATA TRANSFER: SHIFT REGISTERS

Before we describe the serial data transfer operation, we must first examine the basic shift-register arrangement. A shift register is a group of FFs arranged so that the binary numbers stored in the FFs are shifted from one FF to the next for every clock pulse. You have undoubtedly seen shift registers in action in devices such as an electronic calculator, where the digits shown on the display shift over each time you key in a new digit. This is the same action taking place in a shift register.

Figure 5-45(a) shows one way to arrange J-K flip-flops to operate as a four-bit shift register. Note that the FFs are connected so that the output of $X_{3}$ transfers into $X_{2}, X_{2}$ into $X_{1}$, and $X_{1}$ into $X_{0}$. What this means is that upon the occurrence of the NGT of a shift pulse, each FF takes on the value stored previously in the FF on its left. Flip-flop $X_{3}$ takes on a value determined by the conditions present on its $J$ and $K$ inputs when the NGT occurs. For now, we will assume that $X_{3}$ 's $J$ and $K$ inputs are fed by the DATA IN waveform shown in Figure 5-45(b). We will also assume that all FFs are in the 0 state before shift pulses are applied.

The waveforms in Figure 5-45(b) show how the input data are shifted from left to right from FF to FF as shift pulses are applied. When the first NGT occurs at $T_{1}$, each of the FFs $X_{2}, X_{1}$, and $X_{0}$ will have the $J=0, K=1$ condition present at its inputs because of the state of the FF on its left. Flipflop $X_{3}$ will have $J=1, K=0$ because of DATA IN. Thus, at $T_{1}$, only $X_{3}$ will go HIGH, while all the other FFs remain LOW. When the second NGT occurs at $T_{2}$, flip-flop $X_{3}$ will have $J=0, K=1$ because of DATA IN. Flip-flop $X_{2}$ will have $J=1, K=0$ because of the current HIGH at $X_{3}$. Flip-flops $X_{1}$ and $X_{0}$ will still have $J=0, K=1$. Thus, at $T_{2}$, only FF $X_{2}$ will go HIGH, FF $X_{3}$ will go LOW, and FFs $X_{1}$ and $X_{0}$ will remain LOW.

Similar reasoning can be used to determine how the waveforms change at $T_{3}$ and $T_{4}$. Note that on each NGT of the shift pulses, each FF output takes on the level that was present at the output of the FF on its left just prior to the NGT. Of course, $X_{3}$ takes on the level that was present at DATA IN just prior to the NGT.


FIGURE 5-45 Four-bit shift register.

## Hold Time Requirement

In this shift-register arrangement, it is necessary that the FFs have a very small hold time requirement because there are times when the $J, K$ inputs are changing at about the same time as the $C L K$ transition. For example, the $X_{3}$ output switches from 1 to 0 in response to the NGT at $T_{2}$, causing the $J, K$ inputs of $X_{2}$ to change while its CLK input is changing. Actually, because of the propagation delay of $X_{3}$, the $J, K$ inputs of $X_{2}$ won't change for a short time after the NGT. For this reason, a shift register should be implemented using edge-triggered FFs that have a $t_{\mathrm{H}}$ value less than one $C L K$-to-output propagation delay. This latter requirement is easily satisfied by most modern edge-triggered FFs.

## Serial Transfer Between Registers

Figure 5-46(a) shows two three-bit shift registers connected so that the contents of the $X$ register will be serially transferred (shifted) into register $Y$. We are using D flip-flops for each shift register because this requires fewer connections than J-K flip-flops. Notice how $X_{0}$, the last FF of register $X$, is connected to the $D$ input of $\mathrm{Y}_{2}$, the first FF of register Y . Thus, as the shift pulses are applied, the information transfer takes place as follows:

(a)

(b)

FIGURE 5-46 Serial transfer of information from $X$ register into $Y$ register.
$X_{2} \rightarrow X_{1} \rightarrow X_{0} \rightarrow Y_{2} \rightarrow Y_{1} \rightarrow Y_{0}$. The $X_{2}$ FF will go to a state determined by its $D$ input. For now, $D$ will be held LOW, so that $X_{2}$ will go LOW on the first pulse and will remain there.

To illustrate, let us assume that before any shift pulses are applied, the contents of the $X$ register are 101 (i.e., $X_{2}=1, X_{1}=0, X_{0}=1$ ) and the Yregister is at 000 . Refer to the table in Figure $5-46(\mathrm{~b})$, which shows how the states of each FF change as shift pulses are applied. The following points should be noted:

1. On the NGT of each pulse, each FF takes on the value that was stored in the FF on its left prior to the occurrence of the pulse.
2. After three pulses, the 1 that was initially in $X_{2}$ is in $Y_{2}$, the 0 initially in $X_{1}$ is in $Y_{1}$, and the 1 initially in $X_{0}$ is in $Y_{0}$. In other words, the 101 stored in the $X$ register has now been shifted into the $Y$ register. The $X$ register is at 000; it has lost its original data.
3. The complete transfer of the three bits of data requires three shift pulses.

## EXAMPLE 5-13

Assume the same initial contents of the $X$ and $Y$ registers in Figure 5-46. What will be the contents of each FF after the occurrence of the sixth shift pulse?

## Solution

If we continue the process shown in Figure $5-46(b)$ for three more shift pulses, we will find that all of the FFs will be in the 0 state after the sixth pulse. Another way to arrive at this result is to reason as follows: the constant 0 level at the $D$ input of $X_{2}$ shifts in a new 0 with each pulse so that, after six pulses, the registers are filled up with 0 s.

## Shift-Left Operation

The FFs in Figure 5-46 can just as easily be connected so that information shifts from right to left. There is no general advantage of shifting in one direction over another; the direction chosen by a logic designer will often be dictated by the nature of the application, as we shall see.

## Parallel Versus Serial Transfer

In parallel transfer, all of the information is transferred simultaneously upon the occurrence of a single transfer command pulse (Figure 5-44), no matter how many bits are being transferred. In serial transfer, as exemplified by Figure 5-46, the complete transfer of $N$ bits of information requires $N$ clock pulses (three bits requires three pulses, four bits requires four pulses, etc.). Parallel transfer, then, is obviously much faster than serial transfer using shift registers.

In parallel transfer, the output of each FF in register $X$ is connected to a corresponding FF input in register Y. In serial transfer, only the last FF in register $X$ is connected to register Y. In general, then, parallel transfer requires more interconnections between the sending register $(X)$ and the receiving register ( Y ) than does serial transfer. This difference becomes more critical when a greater number of bits of information are being transferred. This is an important consideration when the sending and receiving registers are remote from each other because it determines how many lines (wires) are needed for the transmission of the information.

The choice of either parallel or serial transmission depends on the particular system application and specifications. Often, a combination of the two types is used to take advantage of the speed of parallel transfer and the economy and simplicity of serial transfer. More will be said later about information transfer.

1. True or false: The fastest method for transferring data from one register to another is parallel transfer.
2. What is the major advantage of serial transfer over parallel transfer?
3. Refer to Figure 5-46. Assume that the initial contents of the registers are $X_{2}=0, X_{1}=1, X_{0}=0, Y_{2}=1, Y_{1}=1, Y_{0}=0$. Also assume that the $D$ input of $X_{2}$ is held HIGH. Determine the value of each FF output after the occurrence of the fourth shift pulse.
4. In which form of data transfer does the source of the data not lose its data?

## 5-19 FREQUENCY DIVISION AND COUNTING

Refer to Figure 5-47(a). Each FF has its $J$ and $K$ inputs at the 1 level, so that it will change states (toggle) whenever the signal on its CLK input goes from HIGH to LOW. The clock pulses are applied only to the CLK input of FF $Q_{0}$. Output $Q_{0}$ is connected to the CLK input of FF $Q_{1}$, and output $Q_{1}$ is connected to the CLK input of FF $Q_{2}$. The waveforms in Figure 5-47(b) show how the FFs change states as the pulses are applied. The following important points should be noted:

1. Flip-flop $Q_{0}$ toggles on the negative-going transition of each input clock pulse. Thus, the $Q_{0}$ output waveform has a frequency that is exactly onehalf of the clock pulse frequency.

FIGURE 5-47 J-K flipflops wired as a three-bit binary counter (MOD-8).


(b)
2. Flip-flop $Q_{1}$ toggles each time the $Q_{0}$ output goes from HIGH to LOW. The $Q_{1}$ waveform has a frequency equal to exactly one-half the frequency of the $Q_{0}$ output and therefore one-fourth of the clock frequency.
3. Flip-flop $Q_{2}$ toggles each time the $Q_{1}$ output goes from HIGH to LOW. Thus, the $Q_{2}$ waveform has one-half the frequency of $Q_{1}$ and therefore one-eighth of the clock frequency.
4. Each FF output is a square wave ( $50 \%$ duty cycle).

As described above, each FF divides the frequency of its input by 2 . Thus, if we were to add a fourth FF to the chain, it would have a frequency equal to one-sixteenth of the clock frequency, and so on. Using the appropriate number of FFs, this circuit could divide a frequency by any power of 2. Specifically, using $N$ flip-flops would produce an output frequency from the last FF, which is equal to $1 / 2^{N}$ of the input frequency.

This application of flip-flops is referred to as frequency division. Many applications require a frequency division. For example, your wristwatch is no doubt a "quartz" watch. The term quartz watch means that a quartz crystal is used to generate a very stable oscillator frequency. The natural resonant frequency of the quartz crystal in your watch is likely 1 MHz or more. In order to advance the "seconds" display once every second, the oscillator frequency is divided by a value that will produce a very stable and accurate 1 Hz output frequency.

## Counting Operation

In addition to functioning as a frequency divider, the circuit of Figure 5-47 also operates as a binary counter. This can be demonstrated by examining

Each circle represents one possible state, as indicated by the binary number inside the circle. For example, the circle containing the number 100 represents the 100 state (i.e., $Q_{2}=1, Q_{1}=Q_{0}=0$ ).

The arrows connecting one circle to another show how one state changes to another as a clock pulse is applied. By looking at a particular state circle, we can see which state precedes it and which state follows it. For example, looking at the 000 state, we see that this state is reached whenever the counter is in the 111 state and a clock pulse is applied. Likewise, we see that the 000 state is always followed by the 001 state.

We will use state transition diagrams to help describe, analyze, and design counters and other sequential circuits.

## MOD Number

The counter of Figure $5-47$ has $2^{3}=8$ different states ( 000 through 111). It would be referred to as a MOD-8 counter, where the MOD number indicates the number of states in the counting sequence. If a fourth FF were added, the sequence of states would count in binary from 0000 to 1111 , a total of 16 states. This would be called a MOD-16 counter. In general, if $N$ flip-flops are connected in the arrangement of Figure 5-47, the counter will have $2^{N}$ different states, and so it is a MOD- $2^{N}$ counter. It would be capable of counting up to $2^{N}-1$ before returning to its 0 state.

The MOD number of a counter also indicates the frequency division obtained from the last FF. For instance, a four-bit counter has four FFs, each representing one binary digit (bit), and so it is a MOD-2 ${ }^{4}=$ MOD-16 counter. It can therefore count up to $15\left(=2^{4}-1\right)$. It can also be used to divide the input pulse frequency by a factor of 16 (the MOD number).

We have looked only at the basic FF binary counter. We examine counters in much more detail in Chapter 7.

## EXAMPLE 5-14

Assume that the MOD-8 counter in Figure 5-47 is in the 101 state. What will be the state (count) after 13 pulses have been applied?

## Solution

Locate the 101 state on the state transition diagram. Proceed around the state diagram through eight state changes, and you should be back in the 101 state. Now continue through five more state changes (for a total of 13), and you should end up in the 010 state.

Notice that because this is a MOD-8 counter with eight states, it takes eight state transitions to make one complete excursion around the diagram back to the starting state.

## EXAMPLE 5-15

Consider a counter circuit that contains six FFs wired in the arrangement of Figure 5-47 (i.e., $Q_{5}, Q_{4}, Q_{3}, Q_{2}, Q_{1}, Q_{0}$ ).
(a) Determine the counter's MOD number.
(b) Determine the frequency at the output of the last $\mathrm{FF}\left(Q_{5}\right)$ when the input clock frequency is 1 MHz .
(c) What is the range of counting states for this counter?
(d) Assume a starting state (count) of 000000 . What will be the counter's state after 129 pulses?

## REVIEW QUESTIONS

1. A $20-\mathrm{kHz}$ clock signal is applied to a J-K flip-flop with $J=K=1$. What is the frequency of the FF output waveform?
2. How many FFs are required for a counter that will count 0 to $255_{10}$ ?
3. What is the MOD number of the counter in question 2 ?
4. What is the frequency of the output of the eighth FF when the input clock frequency is 512 kHz ?
5. If this counter starts at 00000000 , what will be its state after 520 pulses?

$$
\text { 5. If this counter starts at } 00000000, \text { what will be its state after } 520 \text { pulses? }
$$

## 5-20 MICROCOMPUTER APPLICATION

Your study of digital systems is still in a relatively early stage, and you have not learned very much about microprocessors and microcomputers. However, you can get a basic idea of how FFs are employed in a typical microprocessor-controlled application without being concerned with all of the details you will need to know later.

Figure 5-50 shows a microprocessor unit (MPU) with its outputs used to transfer binary data to register $X$, which consists of four D flip-flops $X_{3}, X_{2}$, $X_{1}, X_{0}$. One set of MPU outputs is the address code made up of the eight

FIGURE 5-50 Example of a microprocessor transferring binary data to an external register. $\boldsymbol{X}_{1}, X_{0}$ One set of MPU outputs is the address code made up of the eight

## Solution

(a) MOD number $=2^{6}=64$.
(b) The frequency at the last FF will equal the input clock frequency divided by the MOD number. That is,

$$
f\left(\text { at } Q_{5}\right)=\frac{1 \mathrm{MHz}}{64}=15.625 \mathrm{kHz}
$$

(c) The counter will count from $000000_{2}$ to $111111_{2}\left(0\right.$ to $\left.63_{10}\right)$ for a total of 64 states. Note that the number of states is the same as the MOD number.
(d) Because this is a MOD-64 counter, every 64 clock pulses will bring the counter back to its starting state. Therefore, after 128 pulses, the count is back to 000000 . The 129 th pulse brings the counter to the 000001 counter.

outputs $A_{15}, A_{14}, A_{13}, A_{12}, A_{11}, A_{10}, A_{9}, A_{8}$. Most MPUs have at least 16 available address outputs, but they are not always all used. A second set of MPU outputs consists of the four data lines $D_{3}, D_{2}, D_{1}, D_{0}$. Most MPUs have at least eight available data lines. The other MPU output is a timing control signal $\overline{W R}$, which goes LOW when the MPU is ready to write.

Recall that the MPU is the central processing unit of a microcomputer, and its main function is to execute a program of instructions stored in the computer's memory. One of the instructions it might execute could be one that tells the MPU to transfer a binary number from a storage resister within the MPU to the external register $X$. This is called a write cycle. In executing this instruction, the MPU would perform the following steps:

1. Place the binary number onto its data output lines $D_{3}$ through $D_{0}$.
2. Place the proper address code on its output lines $A_{15}$ through $A_{8}$ to select register $X$ as the recipient of the data.
3. Once the data and address outputs are stabilized, the MPU generates the write pulse $W R$ to clock the register and complete the parallel transfer of data into $X$.

There are many situations where an MPU, under the control of a program, will send data to an external register in order to control external events. For example, the individual FFs in the register can control the ON/OFF status of electromechanical devices such as solenoids, relays, motors, and so on (through appropriate interface circuits, of course). The data sent from the MPU to the register will determine which devices are ON and which are OFF. Another common example is when the register is used to hold a binary number for input to a digital-to-analog converter (DAC). The MPU sends the binary number to the register, and the DAC converts it to an analog voltage that may be used to control something such as the position of an electron beam on a CRT screen or the speed of a motor.

## EXAMPLE 5-16

(a) What address code must the MPU generate in order for the data to be transferred into $X$ ?
(b) Assume that $X_{3}-X_{0}=0110, A_{15}-A_{8}=11111111$, and $D_{3}-D_{0}=1011$. What will be in $X$ after a $\overline{W R}$ pulse occurs?

## Solution

(a) In order for the data to be transferred into $X$, the clock pulse must pass through AND gate 2 into the CLK inputs of the FFs. This will happen only if the top input of AND gate 2 is HIGH. This means that all of the inputs to AND gate 1 must be HIGH; that is, $A_{15}$ through $A_{9}$ must be 1 , and $A_{8}$ must be 0 . Thus, the presence of address code 11111110 is needed to allow data to be transferred into $X$.
(b) With $A_{8}=1$, the LOW from AND gate 1 will inhibit $\overline{W R}$ from getting through AND gate 2, and the FFs will not be clocked. Therefore, the contents of register $X$ will not change from 0110 .

## 5-21 SCHMITT-TRIGGER DEVICES

A Schmitt-trigger circuit is not classified as a flip-flop, but it does exhibit a type of memory characteristic that makes it useful in certain special situations. One of those situations is shown in Figure 5-51(a). Here a standard INVERTER is being driven by a logic input that has relatively slow transition times. When these transition times exceed the maximum allowed values (this depends on the particular logic family), the outputs of logic gates and INVERTERs may produce oscillations as the input signal passes through the indeterminate range. The same input conditions can also produce erratic triggering of FFs.

A device that has a Schmitt-trigger type of input is designed to accept noisy slow-changing signals and produce an output that has oscillation-free transitions. The output will generally have very rapid transition times (typically 10 ns ) that are independent of the input signal characteristics. Figure 5-51(b) shows a Schmitt-trigger INVERTER and its response to a slow-changing input.

If you examine the waveforms in Figure 5-51(b), you should note that the output does not change from HIGH to LOW until the input exceeds the positive-going threshold voltage, $V_{\mathrm{T}+}$. Once the output goes LOW, it will remain there even when the input drops back below $V_{\mathrm{T}+}$ (this is its memory characteristic) until it drops all the way down below the negative-going threshold voltage, $V_{\mathrm{T}-\text {. }}$ The values of the two threshold voltages will vary from logic family to logic family, but $V_{\mathrm{T}-}$ will always be less than $V_{\mathrm{T}+}$.

The Schmitt-trigger INVERTER, and all other devices with Schmitttrigger inputs, uses the distinctive symbol shown in Figure 5-51(b) to indicate that they can reliably respond to slow-changing input signals. Logic designers use ICs with Schmitt-trigger inputs to convert slow-changing signals to clean, fast-changing signals that can drive standard IC inputs.

Several ICs are available with Schmitt-trigger inputs. The 7414, 74LS14, and 74HC14 are hex INVERTER ICs with Schmitt-trigger inputs. The 7413, 74LS13, and 74HC13 are dual four-input NANDs with Schmitt-trigger inputs.

1. What could occur when a slow-changing signal is applied to a standard logic IC?
2. How does a Schmitt-trigger logic device operate differently from a standard logic device?

## 5-22 ONE-SHOT (MONOSTABLE MULTIVIBRATOR)

A digital circuit that is somewhat related to the FF is the one-shot (OS). Like the FF, the OS has two outputs, $Q$ and $\bar{Q}$, which are the inverse of each other. Unlike the FF, the OS has only one stable output state (normally $Q=0$, $\bar{Q}=1$ ), where it remains until it is triggered by an input signal. Once triggered, the $O S$ outputs switch to the opposite state ( $Q=1, \bar{Q}=0$ ). It remains in this quasi-stable state for a fixed period of time, $t_{\mathrm{p}}$, which is usually determined by an $R C$ time constant that results from the values of external components connected to the OS. After a time $t_{\mathrm{p}}$, the OS outputs return to their resting state until triggered again.

Figure 5-52(a) shows the logic symbol for a OS. The value of $t_{\mathrm{p}}$ is often indicated somewhere on the OS symbol. In practice, $t_{\mathrm{p}}$ can vary from several nanoseconds to several tens of seconds. The exact value of $t_{\mathrm{p}}$ is variable and is determined by the values of external components $R_{\mathrm{T}}$ and $C_{\mathrm{T}}$.

Input A
HIGH
Single $\ldots$
Threshold
LOW

(a)

(b)

FIGURE 5-51 (a) Standard inverter response to slow noisy input, and (b) Schmitttrigger response to slow noisy input.

FIGURE 5-52 OS symbol and typical waveforms for nonretriggerable operation.

(b)

Two types of one-shots are available in IC form: the nonretriggerable OS and the retriggerable $0 S$.

## Nonretriggerable One-Shot

The waveforms in Figure 5-52(b) illustrate the operation of a nonretriggerable OS that triggers on positive-going transitions at its trigger ( $T$ ) input. The important points to note are:

1. The PGTs at points $a, b, c$, and $e$ will trigger the OS to its quasi-stable state for a time $t_{\mathrm{p}}$, after which it automatically returns to the stable state.
2. The PGTs at points $d$ and $f$ have no effect on the OS because it has already been triggered to the quasi-stable state. The OS must return to the stable state before it can be triggered.
3. The OS output-pulse duration is always the same, regardless of the duration of the input pulses. As stated above, $t_{\mathrm{p}}$ depends only on $R_{\mathrm{T}}$ and $C_{\mathrm{T}}$ and the internal OS circuitry. A typical OS may have a $t_{\mathrm{p}}$ given by $t_{\mathrm{p}}=0.693 R_{\mathrm{T}} C_{\mathrm{T}}$.

## Retriggerable One-Shot

The retriggerable OS operates much like the nonretriggerable OS except for one major difference: it can be retriggered while it is in the quasi-stable state, and it will begin a new $t_{\mathrm{p}}$ interval. Figure 5-53(a) compares the response of both types of OS using a $t_{\mathrm{p}}$ of 2 ms . Let's examine these waveforms.

FIGURE 5-53
(a) Comparison of nonretriggerable and retriggerable OS responses for $t_{\mathrm{p}}=2 \mathrm{~ms}$. (b) Retriggerable OS begins a new $t_{\mathrm{p}}$ interval each time it receives a trigger pulse.

(a)

(b)

Both types of OS respond to the first trigger pulse at $t=1 \mathrm{~ms}$ by going HIGH for 2 ms and then returning LOW. The second trigger pulse at $t=5 \mathrm{~ms}$ triggers both one-shots to the HIGH state. The third trigger pulse at $t=6 \mathrm{~ms}$ has no effect on the nonretriggerable OS because it is already in its quasistable state. However, this trigger pulse will retrigger the retriggerable OS to begin a new $t_{\mathrm{p}}=2 \mathrm{~ms}$ interval. Thus, it will stay HIGH for 2 ms after this third trigger pulse.

In effect, then, a retriggerable OS begins a new $t_{\mathrm{p}}$ interval each time a trigger pulse is applied, regardless of the current state of its $Q$ output. In fact, trigger pulses can be applied at a rate fast enough that the OS will always be retriggered before the end of the $t_{p}$ interval and $Q$ will remain HIGH. This is shown in Figure 5-53(b), where eight pulses are applied every 1 ms . $Q$ does not return LOW until 2 ms after the last trigger pulse.

## Actual Devices

Several one-shot ICs are available in both the retriggerable and the nonretriggerable versions. The 74121 is a single nonretriggerable one-shot IC; the 74221, 74LS221, and 74HC221 are dual nonretriggerable one-shot ICs; the 74122 and 74LS122 are single retriggerable one-shot ICs; the 74123, 74LS123, and 74HC123 are dual retriggerable one-shot ICs.

Figure 5-54(a) shows the traditional symbol for the 74121 nonretriggerable one-shot IC. Note that it contains internal logic gates to allow inputs $A_{1}$, $A_{2}$, and $B$ to trigger the OS in a variety of ways. The $B$ input is a Schmitttrigger type of input that is allowed to have slow transition times and still reliably trigger the OS. The pins labeled $R_{\mathrm{INT}}, R_{\mathrm{EXT}} / C_{\mathrm{EXT}}$, and $C_{\mathrm{EXT}}$ are used to connect an external resistor and capacitor to achieve the desired output pulse duration. Figure 5 -54(b) is the IEEE/ANSI symbol for the 74121 nonretriggerable OS. Note how this symbol represents the logic gates. Also notice the presence of a small pulse with 1 in front of it. This indicates that the device is a nonretriggerable OS. The IEEE/ANSI symbol for a retriggerable OS would not have the 1 in front of the pulse.

## Monostable Multivibrator

Another name for the one-shot is monostable multivibrator because it has only one stable state. One-shots find limited application in most sequential


FIGURE 5-54 Logic symbols for the 74121 nonretriggerable one-shot: (a) traditional; (b) IEEE/ANSI.
clock-controlled systems, and experienced designers generally avoid using them because they are prone to false triggering by spurious noise. When they are used, it is usually in simple timing applications that utilize the predetermined $t_{\mathrm{p}}$ interval. Several of the end-of-chapter problems will illustrate how a OS is used.

1. In the absence of a trigger pulse, what will be the state of a OS output?
2. True or false: When a nonretriggerable OS is pulsed while it is in its quasistable state, the output is not affected.
3. What determines the $t_{\mathrm{p}}$ value for a OS?
4. Describe how a retriggerable OS operates differently from a nonretriggerable OS.

## 5-23 CLOCK GENERATOR CIRCUITS

Flip-flops have two stable states; therefore, we can say that they are bistable multivibrators. One-shots have one stable state, and so we call them monostable multivibrators. A third type of multivibrator has no stable states; it is called an astable or free-running multivibrator. This type of logic circuit switches back and forth (oscillates) between two unstable output states. It is useful for generating clock signals for synchronous digital circuits.

Several types of astable multivibrators are in common use. We will present three of them without any attempt to analyze their operation. They are presented here so that you can construct a clock generator circuit if needed for a project or for testing digital circuits in the lab.

## Schmitt-Trigger Oscillator

Figure $5-55$ shows how a Schmitt-trigger INVERTER can be connected as an oscillator. The signal at $V_{\text {OUT }}$ is an approximate square wave with a frequency that depends on the $R$ and $C$ values. The relationship between the frequency

FIGURE 5-55 Schmitttrigger oscillator using a 7414 INVERTER. A 7413 Schmitt-trigger NAND may also be used.


and $R C$ values is shown in Figure 5-55 for three different Schmitt-trigger INVERTERs. Note the maximum limits on the resistance value for each device. The circuit will fail to oscillate if $R$ is not kept below these limits.

## 555 Timer Used as an Astable Multivibrator

The 555 timer IC is a TTL-compatible device that can operate in several different modes. Figure $5-56$ shows how external components can be connected to a 555 so that it operates as a free-running oscillator. Its output is a repetitive

rectangular waveform that switches between two logic levels, with the time intervals at each logic level determined by the $R$ and $C$ values.

The heart of the 555 timer is made up of two voltage comparators and an SR latch as shown in Figure 5-56. The voltage comparators are devices that produce a HIGH out whenever the voltage on the + input is greater than the voltage on the - input. The external capacitor $(C)$ charges up until its voltage exceeds $2 / 3 \times V_{C C}$ as determined by the upper voltage comparator monitoring $V_{T+}$. When this comparator output goes HIGH, it resets the $S R$ latch, causing the output pin (3) to go LOW. At the same time, $\bar{Q}$ goes HIGH, closing the discharge switch and causing the capacitor to begin to discharge through $R_{B}$. It will continue to discharge until the capacitor voltage drops below $1 / 3 \times V_{C C}$ as determined by the lower-voltage comparator monitoring $V_{T-}$. When this comparator output goes HIGH, it sets the $S R$ latch, causing the output pin to go HIGH, opening the discharge switch, and allowing the capacitor to start charging again as the cycle repeats.

The formulas for these time intervals, $t_{\mathrm{L}}$ and $t_{\mathrm{H}}$, and the overall period of the oscillations, $T$, are given in the figure. The frequency of the oscillations is, of course, the reciprocal of $T$. As the formulas in the diagram indicate, the $t_{\mathrm{L}}$ and $t_{\mathrm{H}}$ intervals cannot be equal unless $R_{A}$ is made zero. This cannot be done without producing excess current through the device. This means that it is impossible to produce a perfect 50 percent duty-cycle square wave output with this circuit. It is possible, however, to get very close to a 50 percent duty cycle by making $R_{B} \gg R_{A}$ (while keeping $R_{A}$ greater than $1 \mathrm{k} \Omega$ ), so that $t_{\mathrm{L}} \approx t_{\mathrm{H}}$.

Calculate the frequency and the duty cycle of the 555 astable multivibrator output for $C=0.001 \mu \mathrm{~F}, R_{A}=2.2 \mathrm{k} \Omega$, and $R_{B}=100 \mathrm{k} \Omega$.

Solution

$$
\begin{aligned}
t_{\mathrm{L}} & =0.693(100 \mathrm{k} \Omega)(0.001 \mu \mathrm{~F})=69.3 \mu \mathrm{~s} \\
t_{\mathrm{H}} & =0.693(102.2 \mathrm{k} \Omega)(0.001 \mu \mathrm{~F})=70.7 \mu \mathrm{~s} \\
T & =69.3+70.7=140 \mu \mathrm{~s} \\
f & =1 / 140 \mu \mathrm{~s}=7.29 \mathrm{kHz} \\
\text { duty cycle } & =70.7 / 140=50.5 \%
\end{aligned}
$$

Note that the duty cycle is close to 50 percent (square wave) because $R_{B}$ is much greater than $R_{A}$. It can be made even closer to 50 percent by making $R_{B}$ even larger compared with $R_{A}$. For instance, you should verify that if we change $R_{A}$ to $1 \mathrm{k} \Omega$ (its minimum allowed value), the results are $f=7.18 \mathrm{kHz}$ and duty cycle $=50.3$ percent.

A simple modification can be made to this circuit to allow a duty cycle of less than 50 percent. The strategy is to allow the capacitor to fill up (charge) with charged particles that flow only through $R_{A}$ and empty (discharge) as charged particles flow only through $R_{B}$. This can be accomplished by simply connecting one diode ( $D_{2}$ ) in series with $R_{B}$ and another diode $\left(D_{1}\right)$ in parallel with $R_{B}$ and $D_{2}$ as shown in the inset of Figure 5-56. The inset circuit replaces $R_{B}$ in the drawing. Diodes are devices that allow charged particles to flow through them in only one direction, as indicated by the arrow head. Diode $D_{1}$ allows all the charging current which has come through $R_{A}$ to bypass $R_{B}$, and $D_{2}$ ensures that none of the charging current can flow through $R_{B}$. All of the discharge current flows through $D_{2}$ and $R_{B}$
when the discharge switch is closed. The equations for the time high and time low for this circuit are

$$
\begin{aligned}
& t_{\mathrm{L}}=0.75 R_{B} C \\
& t_{\mathrm{H}}=0.75 R_{A} C
\end{aligned}
$$

Note: The constant 0.75 is correct only for $V_{C C}=5 \mathrm{~V}$.

Using the diodes along with $R_{B}$ as shown in Figure 5-56, calculate the values of $R_{A}$ and $R_{B}$ necessary to get a $1 \mathrm{kHz}, 25$ percent duty cycle waveform out of a 555 . Assume $C$ is a $0.1 \mu \mathrm{~F}$ capacitor.

## Solution

$$
\begin{aligned}
T & =\frac{1}{F}=\frac{1}{1000}=0.001 \mathrm{~s}=1 \mathrm{~ms} \\
t_{\mathrm{H}} & =0.25 \times T=0.25 \times 1 \mathrm{~ms}=250 \mu \mathrm{~s} \\
R_{A} & =\frac{250 \mu \mathrm{~s}}{0.75 \times C}=\frac{250 \mu \mathrm{~s}}{0.75 \times 0.1 \mu \mathrm{~F}}=3.3 \mathrm{k} \Omega \\
R_{B} & =\frac{750 \mu \mathrm{~s}}{0.75 \times C}=\frac{750 \mu \mathrm{~s}}{0.75 \times 0.1 \mu \mathrm{~F}}=10 \mathrm{k} \Omega
\end{aligned}
$$

## Crystal-Controlled Clock Generators

The output frequencies of the signals from the clock-generating circuits described above depend on the values of resistors and capacitors, and thus they are not extremely accurate or stable. Even if variable resistors are used so that the desired frequency can be adjusted by "tweaking" the resistance values, changes in the $R$ and $C$ values will occur with changes in ambient temperature and with aging, thereby causing the adjusted frequency to drift. If frequency accuracy and stability are critical, another method of generating clock signals can be used: a crystal-controlled clock generator. It employs a highly stable and accurate component called a quartz crystal. A piece of quartz crystal can be cut to a specific size and shape to vibrate (resonate) at a precise frequency that is extremely stable with temperature and aging; frequencies from 10 kHz to 80 MHz are readily achievable. When a crystal is placed in certain circuit configurations, it can produce oscillations at an accurate and stable frequency equal to the crystal's resonant frequency. Crystal oscillators are available as IC packages.

Crystal-controlled clock generator circuits are used in all microprocessorbased systems and microcomputers, and in any application in which a clock signal is used to generate accurate timing intervals. We will see this in some of the applications we encounter in the following chapters.

## REVIEW QUESTIONS

1. Determine the approximate frequency of a Schmitt-trigger oscillator that uses a 74 HC 14 with $R=10 \mathrm{k} \Omega$ and $C=0.005 \mu \mathrm{~F}$.
2. Determine the approximate frequency and duty cycle of the 555 oscillator for $R_{A}=R_{B}=2.2 \mathrm{k} \Omega$ and $C=2000 \mathrm{pF}$.
3. What is the advantage of crystal-controlled clock generator circuits over $R C$-controlled circuits?

## 5-24 TROUBLESHOOTING FLIP-FLOP CIRCUITS

Flip-flop ICs are susceptible to the same kinds of internal and external faults that occur in combinational logic circuits. All of the troubleshooting ideas that were discussed in Chapter 4 can readily be applied to circuits that contain FFs as well as logic gates.

Because of their memory characteristic and their clocked operation, FF circuits are subject to several types of faults and associated symptoms that do not occur in combinational circuits. In particular, FF circuits are susceptible to timing problems that are generally not a concern in combinational circuits. The most common types of FF circuit faults are described.

## Open Inputs

Unconnected or floating inputs of any logic circuit are particularly susceptible to picking up spurious voltage fluctuations called noise. If the noise is large enough in amplitude and long enough in duration, the logic circuit's output may change states in response to the noise. In a logic gate, the output will return to its original state when the noise signal subsides. In a FF, however, the output will remain in its new state because of its memory characteristic. Thus, the effect of noise pickup at any open input is usually more critical for a FF or latch than it is for a logic gate.

The most susceptible FF inputs are those that can trigger the FF to a different state-such as the CLK, PRESET, and CLEAR. Whenever you see a FF output that is changing states erratically, you should consider the possibility of an open connection at one of these inputs.

## EXAMPLE 5-19

Figure 5-57 shows a three-bit shift register made up of TTL flip-flops. Initially, all of the FFs are in the LOW state before clock pulses are applied.

FIGURE 5-57 Example 5-19.


| Clock pulse <br> number | "Expected" |  | "Actual" |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :--- | :--- |
|  | $\mathrm{X}_{2}$ | $\mathrm{X}_{1}$ | $\mathrm{X}_{0}$ | $\mathrm{X}_{2}$ | $\mathrm{X}_{1}$ | $\mathrm{X}_{0}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 1 | 1 | 0 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 0 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 0 |

As clock pulses are applied, each PGT will cause the information to shift from each FF to the one on its right. The diagram shows the "expected" sequence of FF states after each clock pulse. Since $J_{2}=1$ and $K_{2}=0$, flip-flop $X_{2}$ will go HIGH on clock pulse 1 and will stay there for all subsequent pulses. This HIGH will shift into $X_{1}$, and then $X_{0}$ on clock pulses 2 and 3, respectively. Thus, after the third pulse, all FFs will be HIGH and should remain there as pulses are continually applied.

Now let's suppose that the "actual" response of the FF states is as shown in the diagram. Here the FFs change as expected for the first three clock pulses. From then on, flip-flop $X_{0}$, instead of staying HIGH, alternates between HIGH and LOW. What possible circuit fault can produce this operation?

## Solution

On the second pulse, $X_{1}$ goes HIGH. This should make $J_{0}=1, K_{0}=0$ so that all subsequent clock pulses should set $X_{0}=1$. Instead, we see $X_{0}$ changing states (toggling) on all pulses after the second one. This toggle operation would occur if $J_{0}$ and $K_{0}$ were both HIGH. The most probable fault is a break in the connection between $\bar{X}_{1}$ and $K_{0}$. Recall that a TTL device responds to an open input as if it were a logic HIGH, so that an open at $K_{0}$ is the same as a HIGH.

## Shorted Outputs

The following example illustrates how a fault in a FF circuit can cause a misleading symptom that may result in a longer time to isolate the fault.

## EXAMPLE 5-20

Consider the circuit in Figure 5-58 and examine the logic probe indications shown in the accompanying table. There is a LOW at the $D$ input of the FF when pulses are applied to its $C L K$ input, but the $Q$ output fails to go to the LOW state. The technician testing this circuit considers each of the following possible circuit faults:

1. Z2-5 is internally shorted to $V_{C C}$.
2. Z1-4 is internally shorted to $V_{C C}$.

FIGURE 5-58 Example 5-20.

3. Z2-5 or Z1-4 is externally shorted to $V_{C C}$.
4. Z2-4 is internally or externally shorted to GROUND. This would keep $\overline{P R E}$ activated and would override the CLK input.
5. There is an internal failure in Z 2 that prevents $Q$ from responding properly to its inputs.

The technician, after making the necessary ohmmeter checks, rules out the first four possibilities. He also checks Z2's $V_{C C}$ and GROUND pins and finds that they are at the proper voltages. He is reluctant to unsolder Z2 from the circuit until he is certain that it is faulty, and so he decides to look at the clock signal. He uses an oscilloscope to check its amplitude, frequency, pulse width, and transition times. He finds that they are all within the specifications for the 74LS74. Finally, he concludes that Z2 is faulty.

He removes the 74LS74 chip and replaces it with another one. To his dismay, the circuit with the new chip behaves in exactly the same way. After scratching his head, he decides to change the NAND gate chip, although he doesn't know why. As expected, he sees no change in the circuit operation.

Becoming more puzzled, he recalls that his electronics lab instructor emphasized the value of performing a thorough visual check on the circuit board, and so he begins to examine it carefully. While he is doing that, he detects a solder bridge between pins 6 and 7 of Z2. He removes it and tests the circuit, and it functions correctly. Explain how this fault produced the operation observed.

## Solution

The solder bridge was shorting the $\bar{Q}$ output to GROUND. This means that $\bar{Q}$ is permanently stuck LOW. Recall that in all latches and FFs, the $\bar{Q}$ and $Q$ outputs are internally cross-coupled so that the level on one will affect the other. For example, take another look at the internal circuitry for a J-K flipflop in Figure 5-25. Note that a constant LOW at $\bar{Q}$ would keep a LOW at one input of NAND gate 3 so that $Q$ would have to stay HIGH, regardless of the conditions at $J, K$, and $C L K$.

The technician learned a valuable lesson about troubleshooting FF circuits. He learned that both outputs should be checked for faults, even those that are not connected to other devices.

## Clock Skew

One of the most common timing problems in sequential circuits is clock skew. One type of clock skew occurs when a clock signal, because of propagation delays, arrives at the CLK inputs of different FFs at different times. In many situations, the skew can cause a FF to go to a wrong state. This is best illustrated with an example.

Refer to Figure 5-59(a), where the signal CLOCK1 is connected directly to FF $Q_{1}$, and indirectly to $Q_{2}$ through a NAND gate and INVERTER. Both FFs are supposed to be clocked by the occurrence of a NGT of CLOCK1 provided that $X$ is HIGH. If we assume that initially $Q_{1}=Q_{2}=0$ and $X=1$, the NGT of CLOCK1 should set $Q_{1}=1$ and have no effect on $Q_{2}$. The waveforms in Figure 5-59(b) show how clock skew can produce incorrect triggering of $Q_{2}$.

Because of the combined propagation delays of the NAND gate and INVERTER, the transitions of the CLOCK2 signal are delayed with respect to CLOCK1 by an amount of time $t_{1}$. The NGT of CLOCK2 arrives at $Q_{2}$ 's $C L K$ input $t_{1}$ later than the NGT of CLOCK1 appears at $Q_{1}$ 's $C L K$ input. This $t_{1}$ is the

FIGURE 5-59 Clock skew occurs when two flip-flops that are supposed to be clocked simultaneously are clocked at slightly different times due to a delay in the arrival of the clock signal at the second flip-flop.
(a) Extra gating circuits that can cause clock skew; (b) timing showing the later arrival of CLOCK 2.


clock skew. The NGT of CLOCK1 will cause $Q_{1}$ to go HIGH after a time $t_{2}$ that is equal to $Q_{1}$ 's $t_{\text {PLH }}$ propagation delay. If $t_{2}$ is less than the skew $t_{1}, Q_{1}$ will be HIGH when the NGT of CLOCK2 occurs, and this may incorrectly set $Q_{2}=1$ if its setup time requirement, $t_{\mathrm{s}}$, is met.

For example, assume that the clock skew is 40 ns and the $t_{\mathrm{PLH}}$ of $Q_{1}$ is 25 ns . Thus, $Q_{1}$ will go HIGH 15 ns before the NGT of CLOCK2. If $Q_{2}$ 's setup time requirement is smaller than $15 \mathrm{~ns}, Q_{2}$ will respond to the HIGH at its $D$ input when the NGT of CLOCK2 occurs, and $Q_{2}$ will go HIGH. This, of course, is not the expected response of $Q_{2}$. It is supposed to remain LOW.

The effects of clock skew are not always easy to detect because the response of the affected FF may be intermittent (sometimes it works correctly, sometimes it doesn't). This is because the situation is dependent on circuit propagation delays and FF timing parameters, which vary with temperature, length of connections, power supply voltage, and loading. Sometimes just connecting an oscilloscope probe to a FF or gate output will add enough load capacitance to increase the device's propagation delay so that the circuit functions correctly; then when the probe is removed, the incorrect operation reappears. This is the kind of situation that explains why some technicians are prematurely gray.

Problems caused by clock skew can be eliminated by equalizing the delays in the various paths of the clock signal so that the active transition arrives at each FF at approximately the same time. This situation is examined in Problem 5-52.

REVIEW OUESTION

FIGURE 5-60 Three input/output modes.

1. What is clock skew? How can it cause a problem?

## 5-25 SEQUENTIAL CIRCUITS USING HDL*

In Chapters 3 and 4, we used HDL to program simple combinational logic circuits. In this chapter, we have studied logic circuits that latch and clocked flip-flop circuits that sequence through various states in response to a clock edge. These latching and sequential circuits can also be implemented using PLDs and described using HDL.

Section 5-1 of this chapter described a NAND gate latch. You will recall that the unique characteristic of this circuit is the fact that its outputs are cross coupled back to its gates' inputs. This causes the circuit to respond differently depending on which state its output happens to be in. Describing circuits that have outputs that feed back to the input with Boolean equations or HDL involves using the output variables in the conditional portion of the description. With Boolean equations it means including output terms in the right-hand side of the equation. Using IF/THEN constructs it means including output variables in the IF clause. Most PLDs have the ability to feed back the output signal to the input circuitry in order to accommodate latching action.

When writing equations that use feedback, some languages, such as VHDL, require a special designation for the output port. In these cases the port bit is not only an output; it is an output with feedback. The difference is shown in Figure 5-60.


[^1]Rather than describing the operation of a latch using Boolean equations, let's try to think of a behavioral description of how the latch should operate. The situations we need to address are when SBAR is activated, when RBAR is activated, and when neither is activated. Recall that the invalid state occurs when both inputs are activated simultaneously. If we can describe a circuit that always recognizes one of the inputs as the winner when both are active, we can avoid the undesirable results of having an invalid input condition. To describe such a circuit, let's ask ourselves under what conditions the latch should be set ( $Q=1$ ). Certainly, the latch should be set if the SET input is active, but what about after SET goes back to its inactive level? How does the latch know to stay in the SET state? The description needs to use the condition of the output now to determine the future condition of the output. The following statement describes the conditions that should make the output HIGH on an SR latch:

## IF SET is active, THEN $Q$ should be HIGH.

What conditions should make the output LOW?

## IF RESET is active, THEN $Q$ should be LOW.

What if neither input is activated? Then the output should remain the same and we can express this as $Q=Q$. This expression provides the feedback of the output state to be combined with input conditions for the purpose of deciding what happens next to the output.

What if both inputs are activated (i.e., the invalid input combination)? The structure of the IF/ELSE decision shown graphically in Figure 5-61 makes sure that the latch never tries to respond to both inputs. If the SET is active, regardless of what is on RESET, the output will be forced HIGH. The invalid input will always default to a set condition this way. The ELSIF clause is considered only when SET is not active. The use of the feedback term ( $Q=Q$ ) affects the operation (holding action) only when neither input is active.

When you design sequential circuits that feed the output value back to the inputs, it is possible to create an unstable system. A change in the output

FIGURE 5-61 The logic of a behavioral description of an SR latch.


## EXAMPLE 5-21

state might be fed back to the inputs, which changes the output state again, which feeds back to the inputs, which changes the output back again. This oscillation is obviously undesirable and so it is very important to make sure that no combination of inputs and outputs can make this happen. Careful analysis, simulation, and testing should be used to ensure that your circuit is stable under all conditions. For this circuit it is necessary to enable multilevel synthesis for the MAX 7000 series of components before compiling in order to avoid oscillation (at least in the simulator) when changing from the "invalid" input to the "no change" input.

Describe an active-LOW input SR latch with inputs named SBAR, RBAR, and one output named $Q$. It should follow the function table of a NAND latch (see Figure 5-6) and the invalid input combination should produce $Q=1$.
(a) Use AHDL.
(b) Use VHDL.

## Solution

(a) Figure $5-62$ shows a possible AHDL solution. Important items to note are:

1. $Q$ is defined as an OUTPUT, even though it is fed back in the circuit. AHDL allows outputs to be fed back into the circuit.
2. The clause after IF will determine which output state occurs when both inputs are active (invalid state). In this code the SET command rules.
3. To evaluate equality, the double equal sign is used. In other words, SBAR $==0$ evaluates TRUE when SBAR is active (LOW).
```
SUBDESIGN fig5_62
(
        sbar, rbar :INPUT;
    q :OUTPUT;
)
BEGIN
    IF sbar == 0 THEN q = VCC; -- set or illegal command
    ELSIF rbar == 0 THEN q = GND; -- reset
    ELSE q = q; -- hold
    END IF;
END;
```

FIGURE 5-62 A NAND latch using AHDL.
(b) Figure 5-63 shows a possible VHDL solution. Important items to note are:

1. $Q$ is defined as a BUFFER rather than an OUTPUT. This allows it to be fed back in the circuit.
2. A PROCESS describes what happens when the values in the sensitivity list (SBAR, RBAR) change state.
3. The clause after IF will determine which output state occurs when both inputs are active (invalid state). In this code the SET command rules.
```
-- must compile with Multi-Level Synthesis for }7000\mathrm{ enabled
ENTITY fig5_63 IS
PORT ( sbar, rbar :IN BIT;
    q :BUFFER BIT);
END fig5_63;
ARCHITECTURE behavior OF fig5_63 IS
BEGIN
    PROCESS (sbar, rbar)
        BEGIN
            IF sbar = '0' THEN q <= '1'; -- set or illegal command
            ELSIF rbar = '0' THEN q <= '0'; -- reset
            ELSE q <= q; -- hold
            END IF;
        END PROCESS;
END behavior;
```

FIGURE 5-63 A NAND latch using VHDL.

## The $D$ Latch

The transparent $D$ latch can also be easily implemented with HDLs. Altera's software has a library primitive called LATCH that is available. The AHDL module below illustrates using this LATCH primitive. All that is needed is to connect the primitive's enable (.ena) and data (.d) ports to the appropriate module signals. The VHDL module also shown below is a behavioral description of the $D$ latch function. You can also use the LATCH primitive as a component in VHDL.

## AHDL D latch

```
```

SUBDESIGN dlatch_ahdl

```
```

SUBDESIGN dlatch_ahdl
(enable, din :INPUT;
(enable, din :INPUT;
q :OUTPUT ; )
q :OUTPUT ; )
VARIABLE
VARIABLE
q :LATCH;
q :LATCH;
BEGIN
BEGIN
q.ena = enable;
q.ena = enable;
q.d = din;
q.d = din;
END;

```
```

END;

```
```

VHDL D latch

```
ENTITY dlatch_vhdl IS
PORT (enable, din :IN BIT;
        q :OUT BIT);
END dlatch_vhdl;
ARCHITECTURE v OF dlatch_vhdl IS
BEGIN
    PROCESS (enable, din)
    BEGIN
            IF enable = '1' THEN
                q <= din;
            END IF;
        END PROCESS;
END v;
```

1. What is the distinguishing hardware characteristic of latching logic circuits?
2. What is the major characteristic of sequential circuits?

## 5-26 EDGE-TRIGGERED DEVICES

Earlier in this chapter, we introduced edge-triggered devices whose outputs

FIGURE 5-64 JK flip-flop logic primitive.
 respond to the inputs when the clock input sees an "edge." An edge simply means a transition from HIGH to LOW, or vice versa, and is often referred to as an event. If we are writing statements in the code that are concurrent, how can outputs change only when a clock input detects an edge event? The answer to this question differs substantially, depending on the HDL you use. In this section, we want to concentrate on creating clocked logic circuits in their simplest form using HDL. We will use JK flip-flops to correlate with many of the examples found earlier in this chapter.

The JK flip-flop is a standard building block of clocked (sequential) logic circuits known as a logic primitive. In its most common form, it has five inputs and one output, as shown in Figure 5-64. The input/output names can be standardized to allow us to refer to the connections of this primitive or fundamental circuit. The actual operation of the primitive circuit is defined in a library of components that is available to the HDL compiler as it generates a circuit from our description. AHDL uses logic primitives to describe flipflop operation. VHDL offers something similar, but it also allows the designer to describe the clocked logic circuit's operation explicitly in the code.

## AHDL FLIP-FLOPS

A flip-flop can be used in AHDL by declaring a register (even one flip-flop is called a register). Several different types of register primitives are available for use in AHDL, including JKFF, DFF, SRFF, and latch. Each different type of register primitive has its own official names (according to Altera software) for the ports of these primitives. These can be found by using the HELP menu in the ALTERA software and looking under Primitives. Table 5-3 lists some of these names. Registers that use these primitives are declared in the VARIABLE section of the code. The register is given an instance name, just as we have named intermediate variables or buried nodes in previous examples. Instead of declaring it as a node, however, it is declared by the type of the register primitive. For example, a JK flip-flop can be declared as:

VARIABLE
ff1 : JKFF;

TABLE 5-3 Altera primitive port identifiers.

| Standard Part Function | Primitive Port Name |
| :--- | :---: |
| Clock input | clk |
| Asynchronous preset (active-LOW) | prn |
| Asynchronous clear (active-LOW) | clrn |
| J, K, S, R, D inputs | $\mathrm{j}, \mathrm{k}, \mathrm{s}, \mathrm{r}, \mathrm{d}$ |
| Level triggered ENABLE input | ena |
| Q output | q |

The instance name is ff1 (which you can make up) and the register primitive type is JKFF (which Altera requires you to use). Once you have declared a register, it is connected to the other logic in the design using its standard

```
% JK flip-flop circuit %
SUBDESIGN fig5_65
(
    jin, kin, clkin, preset, clear :INPUT;
    qout :OUTPUT;
)
VARIABLE
ff1 :JKFF; -- define this flip-flop as a JKFF type
BEGIN
            ff1.prn = preset; -- these are optional and default to vcc
            ff1.clrn = clear;
            ffl.j = jin; -- connect primitive to the input signal
            ff1.k = kin;
            ff1.clk = clkin;
            qout = ff1.q; -- connect the output pin to the primitive
END;
```

FIGURE 5-65 Single JK flip-flop using AHDL.
port names. The ports (or pins) on the flip-flop are referred to using the instance name, with a dot extension that designates the particular input or output. An example for a JK flip-flop in AHDL is shown in Figure 5-65. Notice that we have made up our own input/output names for this SUBDESIGN in order to distinguish them from the primitive port names. The single flip-flop is declared on line 8, as previously described. The J input or port for this device is then labeled $f f 1 . j$, the K input is $f f 1 . k$, the clock input is $f f 1 . c l k$, and so on. Each of the given port assignment statements will make the needed wiring connections for this design block. The prn and clrn ports are both active-LOW, asynchronous controls such as those commonly found on a standard flip-flop. In fact, these asynchronous controls on an FF primitive can be used to implement an SR latch more efficiently than the code in Figure 5-62. The prn and clrn controls are optional in AHDL and will default to a disabled condition (at a logic 1) if they are omitted from the logic section. In other words, if lines 10 and 11 were deleted, the prn and clrn ports of ff1 would automatically be tied to $V_{C C}$.

## VHDL LIBRARY COMPONENTS

The Altera software comes with some extensive libraries of components and primitives that can be used by a designer. The graphic description of a JKFF component in the Altera library is shown in Figure 5-66(a). After placing the component on the worksheet, each of its ports is connected to inputs and outputs of the module. This same concept can be implemented in VHDL using a library component. The inputs and outputs of these library components can be found by looking under the HELP/Primitives menu. Figure 5-66(b) shows the VHDL COMPONENT declaration for a JK flip-flop primitive. The key things to notice are the name of the component (JKFF) and the names of the ports. They are the same names as those used in the graphic symbol of Figure 5-66(a). Also, notice that the type of each input and output variable is STD_LOGIC. This is one of the IEEE standard data types defined in the library and used by many components in the library.

(b)

FIGURE 5-66 (a) Graphic representation using a component. (b) VHDL component declaration.

Figure 5-67 uses a JKFF component from the library in VHDL to create a circuit equivalent to the graphic design of Figure 5-66(a). The first two lines tell the compiler to use the IEEE library to find the definitions of the std_logic data types. The next two lines tell the compiler that it should look in the Altera library for any standard library components that are used later on in the code. The module inputs and outputs are declared as they were in previous examples, except that the type is now STD_LOGIC rather than BIT. This is because the module port types must match the component port types. Within the architecture section, a name (ff1) is given to this instance of the

```
LIBRARY ieee;
USE ieee.std_logic_1164.all; --defines std_logic types
LIBRARY altera;
USE altera.maxplus2.all; -- provides standard components
ENTITY fig5_67 IS
PORT( clkin, jin, kin, preset, clear :IN std_logic;
    qout :OUT std_logic);
END fig5_67;
ARCHITECTURE a OF fig5_67 IS
BEGIN
    ff1: JKFF PORT MAP ( clk => clkin,
            j => jin,
            k => kin,
            prn => preset,
                        clrn => clear,
                        q => qout);
end a;
```

FIGURE 5-67 A JK flip-flop using VHDL.
component JKFF. The keywords PORT MAP are followed by a list of all the connections that must be made to the component ports. Notice that the component ports (e.g., clk) are listed on the left of the symbol $=>$ and the objects they are connected to (e.g., clkin) are listed on the right.

## VHDL FLIP-FLOPS

Now that we have seen how to use standard components that are available in the library, let's look next at how to create our own component that can be used over and over again. For the sake of comparison we will describe the VHDL code for a JK flip-flop that is identical to the library component JKFF.

VHDL was created as a very flexible language and it allows us to define the operation of clocked devices explicitly in the code, without relying on logic primitives. The key to edge-triggered sequential circuits in VHDL is the PROCESS. As you recall, this keyword is followed by a sensitivity list in parentheses. Whenever a variable in the sensitivity list changes state, the code in the process block determines how the circuit should respond. This is very much like a flip-flop that does nothing until the clock input changes state, at which time it evaluates its inputs and updates its outputs. If the flipflop needs to respond to inputs other than the clock (e.g., preset and clear), they can be added to the sensitivity list. The code in Figure 5-68 demonstrates a JK flip-flop written in VHDL.

On line 9 of the figure, a signal is declared with a name of qstate. Signals can be thought of as wires that connect two points in the circuit description, but they also have characteristics of implied "memory." This means that once a value is assigned to the signal, it will stay at that value until a different value is assigned in the code. In VHDL, a VARIABLE is often used to

```
-- JK Flip-Flop Circuit
ENTITY jk IS
PORT (
    clk, j, k, prn, clrn :IN BIT;
    q :OUT BIT);
END jk ;
ARCHITECTURE a OF jk IS
SIGNAL qstate :BIT;
BEGIN
    PROCESS(clk, prn, clrn) -- respond to any of these signals
    BEGIN
        IF prn = '0' THEN qstate <= '1'; -- asynch preset
        ELSIF clrn = '0' THEN qstate <= '0';-- asynch clear
        ELSIF clk = '1' AND clk'EVENT THEN -- on PGT clock edge
            IF j = 'I' AND k = '1' THEN qstate <= NOT qstate;
            ELSIF j = '1' AND k = '0' THEN qstate <= '1';
            ELSIF j = '0' AND k = '1' THEN qstate <= '0';
            END IF;
        END IF;
    END PROCESS;
    q <= qstate; -- update output pin
END a;
```

FIGURE 5-68 Single JK flip-flop using VHDL.
implement this feature of "memory", but variables must be declared and used within the same description block. In this example, if qstate were declared as a VARIABLE, it would need to be declared within the PROCESS (after line 11) and must be assigned to $q$ before the end of the PROCESS (line 21). Our example uses a SIGNAL that can be declared and used throughout the architecture description.

Notice that the PROCESS sensitivity list contains the asynchronous preset and clear signals. The flip-flop must respond to these inputs as soon as they are asserted (LOW), and these inputs should override the $J, K$, and clock inputs. To accomplish this, we can use the sequential nature of the IF/ELSE constructs. First, the PROCESS will describe what happens only when one of the three signals-clk, prn, or clrn-changes state. The highest priority input in this example is prn because it is evaluated first in line 13. If it is asserted, qstate will be set HIGH and the other inputs will not even be evaluated because they are in the else branch of the decision. If prn is HIGH, clrn will be evaluated in line 14 to see if it is LOW. If it is, the flip-flop will be cleared and nothing else will be evaluated in the PROCESS. Line 15 will be evaluated only if both prn and clrn are HIGH. The term clk' EVENT in line 15 evaluates as TRUE only if there has been a transition on clk. Because $c l k=$ ' 1 ' must be TRUE also, this condition responds only to a rising edge transition on the clock. The next three conditions of lines 16,17 , and 18 are evaluated only following a rising edge on $c l k$ and serve to update the flip-flop's state. In other words, they are nested within the ELSIF statement of line 15. Only the JK input commands for toggle, set, and reset are evaluated by the IF/ELSIF on lines $16-18$. Of course, with a JK there is a fourth command, hold. The "missing" ELSE condition will be interpreted by VHDL as an implied memory device that will then hold the PRESENT state if none of the given JK conditions is TRUE. Note that each IF/ELSIF structure has its own END IF statement. Line 19 ends the decision structure that decides to set, clear, or toggle. Line 20 ends the IF/ELSIF structure that decides among the preset, clear, and clock edge responses. As soon as the PROCESS ends, the flip-flop's state is transferred to the output port $q$.

Regardless of whether you develop your description in AHDL or VHDL, the circuit's proper operation can be verified using a simulator. The most important and challenging part of verification using a simulator is creating a set of hypothetical input conditions that will prove that the circuit does everything it is intended to do. There are many ways to do this, and it is up to the designer to decide which way is best. The simulation used to verify the operation of the JKFF primitive is shown in Figure 5-69. The preset input is initially activated and then, at t1, the clear input is activated. These tests ensure that preset and clear are operating asynchronously. The jin input is HIGH at t 2 and kin is HIGH at t 3 . In between these points, the inputs on jin and kin


FIGURE 5-69 Simulation of the JK flip-flop.
are both LOW. This portion of the simulation tests the synchronous modes of set, hold, and reset. Starting at t4, the toggle command is tested with $j i n=k i n=1$. Notice at t5, preset is asserted (LOW) to test whether preset overrides the toggle command. After t6, the output starts toggling again, and at t 7 , the clear input is shown overriding the synchronous inputs. Testing of all modes of operation and the interaction of various controls is very important when you are simulating.

## REVIEW QUESTIONS

1. What is a logic primitive?
2. What does the designer need to know in order to use a logic primitive?
3. In the Altera system, where can you find information on primitives and library functions?
4. What is the key VHDL element that allows the explicit description of clocked logic circuits?
5. Which library defines the std_logic data types?
6. Which library defines the logic primitives and common components?

## 5-27 HDL CIRCUITS WITH MULTIPLE COMPONENTS

We began this chapter by studying latches. Latches were used to make flipflops and flip-flops were used to make many circuits, including binary counters. A graphic description (logic diagram) of a simple binary up counter is shown in Figure 5-70. This circuit is functionally the same as Figure 5-47, which was drawn with the LSB on the right to make it easier to visualize the numeric value of the binary count. The circuit has been redrawn here to show the signal flow in the more conventional format, with inputs on the left and outputs on the right. Notice that these logic symbols are negative edgetriggered. These flip-flops also do not have asynchronous inputs prn or clrn. Our goal is to describe this counter circuit using HDL by interconnecting three instances of the same JK flip-flop component.

FIGURE 5-70 A three-bit binary counter.


## AHDL RIPPLE-UP COUNTER

A text-based description of this circuit requires three of the same type of flip-flop, just like the graphic description. Refer to Figure 5-71. On line 8 of the figure, bit array notation is used to declare a register of three JK flip-flops. The name of this register is $q$, just like the name of the output port. AHDL can interpret this to mean that the output of each flip-flop

```
% MOD 8 ripple up counter. %
SUBDESIGN fig5_71
(
        clock :INPUT;
        q[2..0] :OUTPUT;
)
VARIABLE
        q[2..0]:JKFF; -- defines three JK FFs
BEGIN
                                    -- note: prn, clrn default to vcc!
        q[2..0].j = VCC; -- toggle mode J=K=1 for all FFs
        q[2..0].k = VCC;
        q[0].clk = !clock;
        q[1].clk = !q[0].q;
        q[2].clk = !q[1].q; -- connect clocks in ripple form
END;
```

FIGURE 5-71 MOD-8 ripple counter in AHDL.
should be connected to the output port. Each bit of the array $q$ has all the attributes of a JKFF primitive. AHDL is very flexible in its use of indexed sets like this. As an example of the use of this set notation, notice how all the J and K inputs for all the flip-flops are tied to VCC in lines 11 and 12. If the flip-flops had been named A, B, and C rather than using a bit array, then individual assignments would be necessary for each J and K input, making the code much longer. Next, the key interconnections are made between the flip-flops to make this a ripple-up counter. The clock signal is inverted and assigned to FF0 clock input (line 13), the Q output of FF0 is inverted and assigned to FF1 clock input (line 14), and so on, forming a ripple counter.

## VHDL RIPPLE-UP COUNTER

We described in Figure 5-68 the VHDL code for a positive-edge-triggered JKFF with preset and clear controls. The counter in Figure 5-70 is negative-edge-triggered and does not require asynchronous preset or clear. Our goal now is write the VHDL code for one of these flip-flops, represent three instances of the same flip-flop, and interconnect the ports to create the counter.

We will start by looking at the VHDL description in Figure 5-72, starting at line 18. This module of VHDL code is describing the operation of a single JK flip-flop component. The name of the component is neg_jk (line 18) and it has inputs $c l k, j$, and $k$ (line 19) and output $q$ (line 20). A signal named qstate is used to hold the state of the flip-flop and connect it to the $q$ output. On line 25, the PROCESS has only clk in its sensitivity list, so it only responds to changes in the clk (PGTs and NGTs). The statement that makes this flip-flop negative-edge-triggered is on line 27. IF (clk'EVENT AND $c l k=$ ' 0 ') is true, then a clk edge has just occurred and clk is now LOW, meaning it must have been an NGT of clk. The IF/ELSE decisions that follow implement the four states of a JK flip-flop.

```
ENTITY fig5_72 IS
PORT ( clock :IN BIT;
    qout :BUFFER BIT_VECTOR (2 DOWNTO 0));
END fig5_72;
ARCHITECTURE counter OF fig5_72 IS
    SIGNAL high :BIT;
    COMPONENT neg_jk
    PORT ( clk, j, k :IN BIT;
            q :OUT BIT);
        END COMPONENT;
BEGIN
    high <= 'l'; -- connect to Vcc
ff0: neg_jk PORT MAP (j => high, k => high, clk => clock, q => qout(0));
ffl: neg_jk PORT MAP (j => high, k => high, clk => qout(0),q => qout(1));
ff2: neg_jk PORT MAP (j => high, k => high, clk => qout(1),q => qout(2));
END counter;
ENTITY neg_jk IS
PORT ( clk, j, k :IN BIT;
            q :OUT BIT);
END neg_jk;
ARCHITECTURE simple of neg_jk IS
        SIGNAL qstate :BIT;
BEGIN
    PROCESS (clk)
    BEGIN
            IF (clk'EVENT AND clk = '0') THEN
                IF j = 'I' AND k = '1' THEN qstate <= NOT qstate; - - toggle
                ELSIF j ='1' AND k = '0' THEN qstate <= '1'; -- set
                ELSIF j = '0' AND k = '1' THEN qstate <= '0'; -- reset
                END IF;
            END IF;
        END PROCESS;
        q <= qstate -- connect flip-flop state to output
END simple;;
```

FIGURE 5-72 MOD-8 ripple counter in VHDL.

Now that we know how one flip-flop named neg_jk works, let's see how we can use it three times in a circuit and hook all the ports together. Line 1 defines the ENTITY that will make up the three-bit counter. Lines $2-3$ contain the definitions of the inputs and outputs. Notice that the outputs are in the form of a three-bit array (bit vector). On line 6 the SIGNAL high can be thought of as a wire used to connect points in the circuit to $V_{C C}$. Line 7 is very important because this is where we declare that we plan to use a component in our design whose name is neg_jk. In this example, the actual code is written at the bottom of the page, but it could be in a separate file or even in a library. This declaration tells the compiler all the important facts about the component and its port names.

The final part of the description is the concurrent section of lines 12-15. First, the signal high is connected to $V_{C C}$ on line 12. The next three lines are instantiations of the flip-flop components. The three instances are named ff0, ff1, and ff2. Each instance is followed by a PORT MAP which lists each port of the component and describes what it is connected to in the module.

[^2]Connecting components together using HDL is not difficult, but it is very tedious. As you can see, the file for even a very simple circuit can be quite long. This method of describing circuits is referred to as the structural level of abstraction. It requires the designer to account for each pin of each component and define signals for each wire that is to interconnect the components. People who are accustomed to using logic diagrams to describe circuits generally find it easy to understand the structural level, but not as easy to read at a glance as the equivalent logic circuit diagram. In fact, it is safe to say that if the structural level of description was all that was available, most people would prefer using graphic descriptions (schematics) rather than HDL. The real advantage of HDL is found in the use of higher levels of abstraction and the ability to tailor components to fit the needs of the project exactly. We will explore the use of these methods, as well as graphical tools to connect modules, in the following chapters.

1. Can the same component be used more than once in the same circuit?
2. In AHDL, where are multiple instances of a component declared?
3. How do you distinguish between multiple instances of a component?
4. In AHDL, what operator is used to "connect" signals?
5. In VHDL, what serves as "wires" that connect components?
6. In VHDL, what keyword identifies the section of code where connections are specified for instances of components?

## SUMMARY

1. A flip-flop is a logic circuit with a memory characteristic such that its $Q$ and $\bar{Q}$ outputs will go to a new state in response to an input pulse and will remain in that new state after the input pulse is terminated.
2. A NAND latch and a NOR latch are simple FFs that respond to logic levels on their SET and RESET inputs.
3. Clearing (resetting) a FF means that its output ends up in the $Q=0 / \bar{Q}=1$ state. Setting a FF means that it ends up in the $Q=1 / \bar{Q}=0$ state.
4. Clocked FFs have a clock input ( $C L K, C P, C K$ ) that is edge-triggered, meaning that it triggers the FF on a positive-going transition (PGT) or a negative-going transition (NGT).
5. Edge-triggered (clocked) FFs can be triggered to a new state by the active edge of the clock input according to the state of the FF's synchronous control inputs ( $S, R$ or $J, K$ or $D$ ).
6. Most clocked FFs also have asynchronous inputs that can set or clear the FF independently of the clock input.
7. The $D$ latch is a modified NAND latch that operates like a D flip-flop except that it is not edge-triggered.
8. Some of the principal uses of FFs include data storage and transfer, data shifting, counting, and frequency division. They are used in sequential circuits that follow a predetermined sequence of states.
9. A one-shot (OS) is a logic circuit that can be triggered from its normal resting state $(Q=0)$ to its triggered state $(Q=1)$, where it remains for a time interval proportional to an $R C$ time constant.
10. Circuits that have a Schmitt-trigger type of input will respond reliably to slow-changing signals and will produce outputs with clean, sharp edges.
11. A variety of circuits can be used to generate clock signals at a desired frequency, including Schmitt-trigger oscillators, a 555 timer, and a crystalcontrolled oscillator.
12. A complete summary of the various types of FFs can be found inside the back cover.
13. Programmable logic devices can be programmed to operate as latching circuits and sequential circuits.
14. Fundamental building blocks called logic primitives are available in the Altera library to help implement larger systems.
15. Clocked flip-flops are available as logic primitives.
16. VHDL code can be written to describe clocked logic explicitly without using logic primitives.
17. VHDL allows HDL files to be used as components in larger systems. Prefabricated components are available in the Altera library.
18. HDL can be used to describe interconnected components in a manner much like a graphic schematic capture tool.

## IMPORTANT TERMS

| flip-flop | trigger | state table |
| :---: | :---: | :---: |
| SET (states/inputs) | pulse-steering circuit | state transition |
| CLEAR | edge-detector circuit | diagram |
| (states/inputs) | clocked J-K flip-flop | MOD number |
| RESET <br> (states/inputs) | toggle mode clocked D flip-flop | Schmitt-trigger circuit |
| NAND gate latch | parallel data transfer | one-shot (OS) |
| contact bounce | $D$ latch | quasi-stable state |
| NOR gate latch | asynchronous inputs | nonretriggerable OS |
| pulses | override inputs | retriggerable OS |
| clock | common-control | astable or free- |
| positive-going | block | running |
| transition (PGT) | propagation delay | multivibrator |
| negative-going | sequential circuits | 555 timer |
| transition (NGT) | registers | crystal-controlled |
| clocked flip-flop | data transfer | clock generator |
| period frequency | synchronous transfer | clock skew |
| edge-triggered | asynchronous (jam) | EVENT |
| control inputs | transfer | logic primitive |
| synchronous control | jam transfer | nested |
| inputs | serial data transfer | COMPONENT |
| setup time, $t_{\text {S }}$ | shift register | PORT MAP |
| hold time, $t_{\mathrm{H}}$ | frequency division | structural level of |
| clocked S-R flip-flop | binary counter | abstraction |

## PROBLEMS

## SECTIONS 5-1 TO 5-3

B $\quad 5-1$. Assuming that $Q=0$ initially, apply the $x$ and $y$ waveforms of Figure 5-73 to the SET and RESET inputs of a NAND latch, and determine the $Q$ and $\bar{Q}$ waveforms.

FIGURE 5-73 Problems
5-1 to 5-3.


B 5-2. Invert the $x$ and $y$ waveforms of Figure 5-73, apply them to the SET and RESET inputs of a NOR latch, and determine the $Q$ and $\bar{Q}$ waveforms. Assume that $Q=0$ initially.
5-3. ${ }^{\star}$ The waveforms of Figure 5-73 are connected to the circuit of Figure 5-74. Assume that $Q=0$ initially, and determine the $Q$ waveform.

FIGURE 5-74 Problem
5-3.


D 5-4. Modify the circuit of Figure 5-9 to use a NOR gate latch.
D 5-5. Modify the circuit of Figure 5-12 to use a NAND gate latch.
T 5-6. ${ }^{\star}$ Refer to the circuit of Figure 5-13. A technician tests the circuit operation by observing the outputs with a storage oscilloscope while the switch is moved from $A$ to $B$. When the switch is moved from $A$ to $B$, the scope display of $X_{B}$ appears as shown in Figure 5-75. What circuit fault could produce this result? (Hint: What is the function of the NAND latch?)

FIGURE 5-75 Problem 5-6.


## SECTIONS 5-4 THROUGH 5-6

B 5-7. A certain clocked FF has minimum $t_{\mathrm{S}}=20 \mathrm{~ns}$ and $t_{\mathrm{H}}=5 \mathrm{~ns}$. How long must the control inputs be stable prior to the active clock transition?

[^3]B 5-8. Apply the $S, R$, and $C L K$ waveforms of Figure 5-19 to the FF of Figure 5-20, and determine the $Q$ waveform.
B 5-9.*Apply the waveforms of Figure 5-76 to the FF of Figure 5-19 and determine the waveform at $Q$. Repeat for the FF of Figure 5-20. Assume $Q=0$ initially.

FIGURE 5-76 Problem 5-9.


5-10. Draw the following digital pulse waveforms. Label $t_{\mathrm{r}}, t_{\mathrm{f}}$, and $t_{\mathrm{w}}$, leading edge, and trailing edge.
(a) A negative TTL pulse with $t_{\mathrm{r}}=20 \mathrm{~ns}, t_{\mathrm{f}}=5 \mathrm{~ns}$, and $t_{\mathrm{W}}=50 \mathrm{~ns}$.
(b) A positive TTL pulse with $t_{\mathrm{r}}=5 \mathrm{~ns}, t_{\mathrm{f}}=1 \mathrm{~ns}, t_{\mathrm{W}}=25 \mathrm{~ns}$.
(c) A positive pulse with $t_{\mathrm{w}}=1 \mathrm{~ms}$ whose leading edge occurs every 5 ms . Give the frequency of this waveform.

## SECTION 5-7

B 5-11. Apply the $J, K$, and CLK waveforms of Figure $5-23$ to the FF of Figure 5-24. Assume that $Q=1$ initially, and determine the $Q$ waveform.
D 5-12. (a) ${ }^{\star}$ Show how a J-K flip-flop can operate as a toggle FF (changes states on each clock pulse). Then apply a $10-\mathrm{kHz}$ clock signal to its $C L K$ input and determine the waveform at $Q$.
(b) Connect $Q$ from this FF to the CLK input of a second J-K FF that also has $J=K=1$. Determine the frequency of the signal at this FF's output.
B 5-13. The waveforms shown in Figure 5-77 are to be applied to two different FFs:
(a) positive-edge-triggered J-K
(b) negative-edge-triggered J -K

Draw the $Q$ waveform response for each of these FFs, assuming that $Q=0$ initially. Assume that each FF has $t_{\mathrm{H}}=0$.

FIGURE 5-77 Problem 5-13.


## SECTION 5-8

N $\quad 5-14$. A D FF is sometimes used to delay a binary waveform so that the binary information appears at the output a certain amount of time after it appears at the $D$ input.
(a) ${ }^{\star}$ Determine the $Q$ waveform in Figure 5-78, and compare it with the input waveform. Note that it is delayed from the input by one clock period.
(b) How can a delay of two clock periods be obtained?


FIGURE 5-78 Problem 5-14.

B 5-15. (a) Apply the $S$ and CLK waveforms of Figure 5-76 to the $D$ and $C L K$ inputs of a D FF that triggers on PGTs. Then determine the waveform at $Q$.
(b) Repeat using the $C$ waveform of Figure 5-76 for the $D$ input.

B 5-16. ${ }^{\star}$ An edge-triggered D flip-flop can be made to operate in the toggle mode by connecting it as shown in Figure 5-79. Assume that $Q=0$ initially, and determine the $Q$ waveform.

FIGURE 5-79 D flip-flop connected to toggle (Problem 5-16).


## SECTION 5-9

5-17. (a) Apply the $S$ and CLK waveforms of Figure 5-76 to the $D$ and $E N$ inputs of a D latch, respectively, and determine the waveform at $Q$.
(b) Repeat using the $C$ waveform applied to $D$.

5-18. Compare the operation of the $D$ latch with a negative-edge-triggered D flip-flop by applying the waveforms of Figure 5-80 to each and determining the $Q$ waveforms.

FIGURE 5-80 Problem 5-18.


5-19. In Problem 5-16, we saw how an edge-triggered $D$ flip-flop can be operated in the toggle mode. Explain why this same idea will not work for a $D$ latch.

## SECTION 5-10

5-20. Determine the $Q$ waveform for the FF in Figure 5-81. Assume that $Q=0$ initially, and remember that the asynchronous inputs override all other inputs.

FIGURE 5-81 Problem 5-20.


B 5-21. Apply the $\overline{C L K}, \overline{P R E}$, and $\overline{C L R}$ waveforms of Figure $5-32$ to a positive-edge-triggered D flip-flop with active-LOW asynchronous inputs. Assume that $D$ is kept HIGH and $Q$ is initially LOW. Determine the $Q$ waveform.
B 5-22. Apply the waveforms of Figure 5-81 to a D flip-flop that triggers on NGTs and has active-LOW asynchronous inputs. Assume that $D$ is kept LOW and that $Q$ is initially HIGH. Draw the resulting $Q$ waveform.

## SECTION 5-12

B 5-23. Use Table 5-2 in Section 5-12 to determine the following.
(a) ${ }^{\star}$ How long can it take for the $Q$ output of a 74C74 to switch from 0 to 1 in response to an active $C L K$ transition?
(b) ${ }^{\star}$ Which FF in Table $5-2$ requires its control inputs to remain stable for the longest time after the active CLK transition? Before the transition?
(c) What is the narrowest pulse that can be applied to the $\overline{P R E}$ of a 7474 FF?
B 5-24. Use Table 5-2 to determine the following:
(a) How long does it take to asynchronously clear a 74LS112?
(b) How long does it take to asynchronously set a 74 HC 112 ?
(c) What is the shortest acceptable interval between active clock transitions for a 7474?
(d) The D input of a 74 HC 112 goes HIGH 15 ns before the active clock edge. Will the data be stored reliably in the flip-flop?
(e) How long does it take (after the clock edge) to synchronously store a 1 in a cleared 7474 D flip-flop?

## SECTIONS 5-15 AND 5-16

D 5-25.^Modify the circuit of Figure 5-40 to use a J-K flip-flop.

D 5-26. In the circuit of Figure $5-82$, inputs $A, B$, and $C$ are all initially LOW. Output $Y$ is supposed to go HIGH only when $A, B$, and $C$ go HIGH in a certain sequence.
(a) Determine the sequence that will make Y go HIGH.
(b) Explain why the START pulse is needed.
(c) Modify this circuit to use D FFs.

FIGURE 5-82 Problem 5-26.


SECTIONS 5-17 AND 5-18
D 5-27.^(a) Draw a circuit diagram for the synchronous parallel transfer of data from one three-bit register to another using J-K flip-flops.
(b) Repeat for asynchronous parallel transfer.

N, D 5-28. A recirculating shift register is a shift register that keeps the binary information circulating through the register as clock pulses are applied. The shift register of Figure 5-45 can be made into a circulating register by connecting $X_{0}$ to the DATA IN line. No external inputs are used. Assume that this circulating register starts out with 1011 stored in it (i.e., $X_{3}=1, X_{2}=0, X_{1}=1$, and $X_{0}=1$ ). List the sequence of states that the register FFs go through as eight shift pulses are applied.
D 5-29.*Refer to Figure 5-46, where a three-bit number stored in register $X$ is serially shifted into register Y. How can the circuit be modified so that, at the end of the transfer operation, the original number stored in $X$ is present in both registers? (Hint: See Problem 5-28.)

## SECTION 5-19

B 5-30. Refer to the counter circuit of Figure 5-47 and answer the following:
(a) ${ }^{\star}$ If the counter starts at 000 , what will be the count after 13 clock pulses? After 99 pulses? After 256 pulses?
(b) If the counter starts at 100 , what will be the count after 13 pulses? After 99 pulses? After 256 pulses?
(c) Connect a fourth J-K FF $\left(X_{3}\right)$ to this counter and draw the state transition diagram for this 4 -bit counter. If the input clock frequency is 80 MHz , what will the waveform at $X_{3}$ look like?
B 5-31. Refer to the binary counter of Figure 5-47. Change it by connecting $\bar{X}_{0}$ to the CLK of flip-flop $X_{1}$, and $\bar{X}_{1}$ to the CLK of flip-flop $X_{2}$. Start with all FFs in the 1 state, and draw the various FF output waveforms ( $X_{0}$, $X_{1}, X_{2}$ ) for 16 input pulses. Then list the sequence of FF states as was done in Figure 5-48. This counter is called a down counter. Why?
B 5-32. Draw the state transition diagram for this down counter, and compare it with the diagram of Figure 5-49. How are they different?

B 5-33.*(a) How many FFs are required to build a binary counter that counts from 0 to 1023?
(b) Determine the frequency at the output of the last FF of this counter for an input clock frequency of 2 MHz .
(c) What is the counter's MOD number?
(d) If the counter is initially at zero, what count will it hold after 2060 pulses?
B 5-34. A binary counter is being pulsed by a $256-\mathrm{kHz}$ clock signal. The output frequency from the last FF is 2 kHz .
(a) Determine the MOD number.
(b) Determine the counting range.

B $\quad 5-35$. A photodetector circuit is being used to generate a pulse each time a customer walks into a certain establishment. The pulses are fed to an eight-bit counter. The counter is used to count these pulses as a means for determining how many customers have entered the store. After closing the store, the proprietor checks the counter and finds that it shows a count of $00001001_{2}=9_{10}$. He knows that this is incorrect because there were many more than nine people in his store. Assuming that the counter circuit is working properly, what could be the reason for the discrepancy?

## SECTION 5-20

D $5-36$. Modify the circuit of Figure $5-50$ so that only the presence of address code 10110110 will allow data to be transferred to register X.
T 5-37. Suppose that the circuit of Figure 5-50 is malfunctioning so that data are being transferred to $X$ for either of the address codes 11111110 or 11111111. What are some circuit faults that could be causing this?

N, D 5-38. Many microcontrollers share the same pins to output the lower address and transfer data. In order to hold the address constant while the data are transferred. The address information is stored in a latch which is enabled by the control signal ALE (address latch enable) as shown in Figure 5-83. Connect this latch to the microcontroller such that it takes what is on the lower address and data lines while ALE is HIGH and holds it on the lower address only lines when ALE is LOW.

FIGURE 5-83 Problem 5-38.


D 5-39. Modify the circuit of Figure 5-50 so that the MPU has eight data output lines connected to transfer eight bits of data to an eight-bit register made up of two 74HC175 ICs [Figure 5-34(b)]. Show all circuit connections.

## SECTION 5-22

B $\quad 5-40$. Refer to the waveforms in Figure 5-53(a). Change the OS pulse duration to 0.5 ms and determine the $Q$ output for both types of OS. Then repeat using a $O S$ pulse duration of 1.5 ms .

## N

5-41. ${ }^{\star}$ Figure 5-84 shows three nonretriggerable one-shots connected in a timing chain that produces three sequential output pulses. Note the " 1 " in front of the pulse on each OS symbol to indicate nonretriggerable operation. Draw a timing diagram showing the relationship between the input pulse and the three OS outputs. Assume an input pulse duration of 10 ms .

FIGURE 5-84 Problem 5-41.


5-42. A retriggerable OS can be used as a pulse-frequency detector that detects when the frequency of a pulse input is below a predetermined value. A simple example of this application is shown in Figure 5-85. The operation begins by momentarily closing switch SW1.
(a) Describe how the circuit responds to input frequencies above 1 kHz .
(b) Describe how the circuit responds to input frequencies below 1 kHz .
(c) How would you modify the circuit to detect when the input frequency drops below 50 kHz ?

FIGURE 5-85 Problem 5-42.


5-43. Refer to the logic symbol for a 74121 nonretriggerable one-shot in Figure 5-54(a).
(a)^What input conditions are necessary for the OS to be triggered by a signal at the $B$ input?
(b) What input conditions are necessary for the OS to be triggered by a signal at the $A_{1}$ input?
C, D 5-44. The output pulse width from a 74121 OS is given by the approximate formula

$$
t_{\mathrm{p}} \approx 0.7 R_{\mathrm{T}} C_{\mathrm{T}}
$$

where $R_{\mathrm{T}}$ is the resistance connected between the $R_{\mathrm{EXT}} / C_{\mathrm{EXT}}$ pin and $V_{C C}$, and $C_{\mathrm{T}}$ is the capacitance connected between the $C_{\text {EXT }}$ pin and the $R_{\mathrm{EXT}} / C_{\mathrm{EXT}}$ pin. The value for $R_{\mathrm{T}}$ can be varied between 2 and $40 \mathrm{k} \Omega$, and $C_{\mathrm{T}}$ can be as large as $1000 \mu \mathrm{~F}$.
(a) Show how a 74121 can be connected to produce a negative-going pulse with a 5 -ms duration whenever either of two logic signals ( $E$ or $F$ ) makes a NGT. Both $E$ and $F$ are normally in the HIGH state.
(b) Modify the circuit so that a control input signal, $G$, can disable the OS output pulse, regardless of what occurs at $E$ or $F$.

## SECTION 5-23

B, D 5-45. . Show how to use a 74LS14 Schmitt-trigger INVERTER to produce an approximate square wave with a frequency of 10 kHz .
B, D 5-46. Design a 555 free-running oscillator to produce an approximate square wave at 40 kHz . $C$ should be kept at 500 pF or greater.
D 5-47. A 555 oscillator can be combined with a J-K flip-flop to produce a perfect ( 50 percent duty cycle) square wave. Modify the circuit of Problem 5-46 to include a J-K flip-flop. The final output is still to be a $40-\mathrm{kHz}$ square wave.
5-48. Design a 555 timer circuit that will produce a 10 percent duty-cycle $5-\mathrm{kHz}$ waveform. Choose a capacitor greater than 500 pF and resistors less than $100 \mathrm{k} \Omega$. Draw the circuit diagram with pin numbers labeled.
C, $\mathbf{N}$ 5-49. The circuit in Figure 5-86 can be used to generate two nonoverlapping clock signals at the same frequency. These clock signals were used in early microprocessor systems that required four different clock transitions to synchronize their operations.
(a) Draw the CP1 and CP2 timing waveforms if CLOCK is a $1-\mathrm{MHz}$ square wave. Assume that $t_{\text {PLH }}$ and $t_{\text {PHL }}$ are 20 ns for the FF and 10 ns for the AND gates.

FIGURE 5-86 Problem 5-49.

(b) This circuit would have a problem if the FF were changed to one that responds to a PGT at CLK. Draw the CP1 and CP2 waveforms for that situation. Pay particular attention to conditions that can produce glitches.

## SECTION 5-24

5-50. Refer to the counter circuit in Figure 5-47. Assume that all asynchronous inputs are connected to $V_{C C}$. When tested, the circuit waveforms appear as shown in Figure 5-87. Consider the following list of possible faults. For each one, indicate "yes" or "no" as to whether it could cause the observed results. Explain each response.
(a) ${ }^{\star} C L R$ input of $X_{2}$ is open.
(b) ${ }^{\star} X_{1}$ output's transition times are too long, possibly due to loading.
(c) $X_{2}$ output is shorted to ground.
(d) $X_{2}$ 's hold time requirement is not being met.


FIGURE 5-87 Problem 5-50.

C, T 5-51. Consider the situation of Figure 5-59 for each of the following sets of timing values. For each, indicate whether or not flip-flop $Q_{2}$ will respond correctly.
$(\mathrm{a})^{\star}$ Each FF: $t_{\mathrm{PLH}}=12 \mathrm{~ns} ; t_{\mathrm{PHL}}=8 \mathrm{~ns} ; t_{\mathrm{S}}=5 \mathrm{~ns} ; t_{\mathrm{H}}=0 \mathrm{~ns}$
NAND gate: $t_{\text {PLH }}=8 \mathrm{~ns} ; t_{\mathrm{PHL}}=6 \mathrm{~ns}$
INVERTER: $t_{\text {PLH }}=7 \mathrm{~ns} ; t_{\text {PHL }}=5 \mathrm{~ns}$
(b) Each FF: $t_{\text {PLH }}=10 \mathrm{~ns} ; t_{\mathrm{PHL}}=8 \mathrm{~ns} ; t_{\mathrm{S}}=5 \mathrm{~ns} ; t_{\mathrm{H}}=0 \mathrm{~ns}$

NAND gate: $t_{\text {PLH }}=12 \mathrm{~ns} ; t_{\text {PHL }}=10 \mathrm{~ns}$
INVERTER: $t_{\text {PLH }}=8 \mathrm{~ns} ; t_{\mathrm{PHL}}=6 \mathrm{~ns}$
D 5-52. Show and explain how the clock skew problem in Figure 5-59 can be eliminated by the appropriate insertion of two INVERTERs.
T 5-53. Refer to the circuit of Figure 5-88. Assume that the ICs are of the TTL logic family. The $Q$ waveform was obtained when the circuit was tested with the input signals shown and with the switch in the "up" position; it is not correct. Consider the following list of faults, and for each indicate "yes" or "no" as to whether it could be the actual fault. Explain each response.
(a)* Point $X$ is always LOW due to a faulty switch.
(b) ${ }^{\star}$ Z1 pin 1 is internally shorted to $V_{C C}$.

FIGURE 5-88 Problem 5-53.

(a)

(b)
(c) The connection from $\mathrm{Z} 1-3$ to $\mathrm{Z} 2-3$ is broken.
(d) There is a solder bridge between pins 6 and 7 of Z 1 .

C 5-54. The circuit of Figure 5-89 functions as a sequential combination lock. To operate the lock, proceed as follows:

1. Momentarily activate the RESET switch.
2. Set the switches SWA, SWB, and SWC to the first part of the combination. Then momentarily toggle the ENTER switch back and forth.
3. Set the switches to the second part of the combination, and toggle ENTER again. This should produce a HIGH at $Q_{2}$ to open the lock.
If the incorrect combination is entered in either step, the operator must start the sequence over. Analyze the circuit and determine the correct sequence of combinations that will open the lock.
C, T $5-55$. ${ }^{\star}$ When the combination lock of Figure $5-89$ is tested, it is found that entering the correct combination does not open the lock. A logic probe check shows that entering the correct first combination sets $Q_{1}$ HIGH, but entering the correct second combination produces only a momentary pulse at $Q_{2}$. Consider each of the following faults and indicate which one(s) could produce the observed operation. Explain each choice.
(a) Switch bounce at SWA, SWB, or SWC.
(b) $C L R$ input of $Q_{2}$ is open.
(c) Connection from NAND gate 4 output to NAND gate 3 input is open.


FIGURE 5-89 Problems 5-54 and 5-55.

## DRILL QUESTIONS

B $\quad$ 5-56. For each statement indicate what type of FF is being described.
(a)^Has a SET and a CLEAR input but does not have a CLK input
(b) ${ }^{\star}$ Will toggle on each CLK pulse when its control inputs are both HIGH
(c) ${ }^{\star}$ Has an ENABLE input instead of a $C L K$ input
(d) ${ }^{\star}$ Is used to transfer data easily from one FF register to another
(e) Has only one control input
(f) Has two outputs that are complements of each other
(g) Can change states only on the active transition of $C L K$
(h) Is used in binary counters

B 5-57. Define the following terms.
(a) Asynchronous inputs
(b) Edge-triggered
(c) Shift register
(d) Frequency division
(e) Asynchronous (jam) transfer
(f) State transition diagram
(g) Parallel data transfer
(h) Serial data transfer
(i) Retriggerable one-shot
(j) Schmitt-trigger inputs

## SECTION 5-25

B 5-58. Simulate the HDL design for a NAND latch given in Figure 5-62 (AHDL) or Figure 5-63 (VHDL). What does this SR latch do if an "invalid" input command is applied? Since we know that any SR latch can have an unusual output result when an invalid input command is applied, you should simulate that input condition as well as the latch's normal set, reset, and hold commands. Some latch designs can have a tendency for the output to oscillate when an invalid command is followed by a hold command, so be sure to check for that in your simulation.
B, $\mathbf{H} \quad$ 5-59.* Write an HDL design file for an active-HIGH input SR latch.
B, H 5-60. Modify the latch description given in Figure 5-62 (AHDL) or Figure 5-63 (VHDL) to make the SR reset if an invalid input is applied. Simulate the design.
B, H 5-61. ${ }^{\star}$ Add inverted outputs to the HDL NAND latch designs given in Figure 5-62 or Figure 5-63. Verify correct operation by simulation.
B 5-62. Simulate the AHDL or VHDL design for a D latch given in Section 5-25.
D, H 5-63. Create a four-bit transparent latch with one enable input based on the AHDL or VHDL design for a single D latch device given in Section 5-25. Simulate the four-bit latch.
$\mathbf{D}, \mathbf{H}, \mathbf{N}$ 5-64. A toggle ( T ) flip-flop has a single control input ( T ). When $\mathrm{T}=0$, the flip-flop is in the no change state, similar to a JKFF with $\mathrm{J}=\mathrm{K}=0$. When $\mathrm{T}=1$, the flip-flop is in the toggle mode, similar to a JKFF with $\mathrm{J}=\mathrm{K}=1$. Write the design file in
(a) AHDL
(b) VHDL

H 5-65. (a) Write an AHDL design file for the shift register shown in Figure 5-45.
(b) Write a VHDL design file for the shift register shown in Figure 5-45.
H $\quad$ 5-66. (a) ${ }^{\star}$ Write an AHDL design file for the shift register shown in Figure 5-46.
(b) ${ }^{\star}$ Write a VHDL design file for the shift register shown in Figure 5-46.

5-67. (a) Write an AHDL design file for the FF circuit shown in Figure 5-59.
(b) Write a VHDL design file for the FF circuit shown in Figure 5-59.

5-68. Simulate the operation of either Problem 5-74 or 5-75. (The simulations should be identical and match the results in Figure 5-58.)
H 5-69. (a) Write an AHDL design file to implement the entire circuit of Figure 5-89.
(b) Write a VHDL design file to implement the entire circuit of Figure 5-89.

## ANSWERS TO SECTION REVIEW OUESTIONS

## SECTION 5-1

1. HIGH; LOW
2. $Q=0, \bar{Q}=1$
3. True
4. Apply a momentary LOW to $\overline{\mathrm{SET}}$ input.

## SECTION 5-2

1. LOW, HIGH
2. $Q=1$ and $\bar{Q}=0 \quad$ 3. Make CLEAR $=1$
3. $\overline{\text { SET }}$ and

RESET would both be normally in their active-LOW state.

## SECTION 5-4

1. Synchronous control inputs and clock input 2. The FF output can change only when the appropriate clock transition occurs. 3. False 4. Setup time is the required interval immediately prior to the active edge of the CLK signal during which the control inputs must be held stable. Hold time is the required interval immediately following the active edge of $C L K$ during which the control inputs must be held stable.

## SECTION 5-5

## 1. HIGH; LOW; HIGH 2. Because $C L K^{\star}$ is HIGH only for a few nanoseconds

## SECTION 5-6

1. True
2. No
3. $J=1, K=0$

## SECTION 5-7

1. $Q$ will go LOW at point $a$ and remain LOW. 2. False. The $D$ input can change without affecting $Q$ because $Q$ can change only on the active CLK edge. 3. Yes, by converting to D FFs (Figure 5-25).

## SECTION 5-8

1. In a $D$ latch, the $Q$ output can change while $E N$ is HIGH. In a D flip-flop, the output can change only on the active edge of $C L K$. 2. False 3. True

## SECTION 5-9

1. Asynchronous inputs work independently of the $C L K$ input. 2. Yes, because $\overline{P R E}$ is active-LOW

$$
\text { 3. } J=K=1, \overline{P R E}=\overline{C L R}=1 \text {, and a PGT at } C L K
$$

## SECTION 5-10

1. The triangle inside the rectangle indicates edge-triggered operation; the right triangle outside the rectangle indicates triggering on a NGT. 2. It is used to indicate the function of those inputs that are common to more than one circuit on the chip.

## SECTION 5-11

1. $t_{\mathrm{PLH}}$ and $t_{\mathrm{PHL}} \quad$ 2. False; the waveform must also satisfy $t_{\mathrm{W}}(L)$ and $t_{\mathrm{W}}(H)$ requirements.

## SECTION 5-17

1. False
2. D flip-flop
3. Six
4. True

## SECTION 5-18

1. True
2. Fewer interconnections between registers
3. $X_{2} X_{1} X_{0}=111$;
$Y_{2} Y_{1} Y_{0}=101 \quad$ 4. Parallel

## SECTION 5-19

1. 10 kHz
2. Eight
3. 256
4. 2 kHz
5. $00001000_{2}=8_{10}$

SECTION 5-21

1. The output may contain oscillations. 2 . It will produce clean, fast output signals even for slow-changing input signals.

## SECTION 5-22

1. $Q=0, \bar{Q}=1 \quad$ 2. True $\quad$ 3. External $R$ and $C$ values $\quad$ 4. For a retriggerable OS, each new trigger pulse begins a new $t_{\mathrm{p}}$ interval, regardless of the state of the $Q$ output.

## SECTION 5-23

1. 24 kHz
2. $109.3 \mathrm{kHz} ; 66.7$ percent
3. Frequency stability

## SECTION 5-24

1. Clock skew is the arrival of a clock signal at the CLK inputs of different FFs at different times. It can cause a FF to go to an incorrect state.

## SECTION 5-25

1. Feedback: The outputs are combined with the inputs to determine the next state of the outputs. 2. It progresses through a predetermined sequence of states in response to an input clock signal.

## SECTION 5-26

1. A standard building block from a library of components that performs some fundamental logic function. 2. The names of each input and output and the primitive name that is recognized by the development system. 3. Under the HELP menu. 4. The PROCESS allows sequential IF constructs and the EVENT attribute detects transitions. 5. ieee.std_logic_1164. 6. altera.maxplus2

## SECTION 5-27

1. Yes 2. In the VARIABLE section.
2. Each is assigned a variable name.
3. $=$ 5. SIGNALs 6. PORT MAP

[^0]:    *After occurrence of NGT

[^1]:    *As stated in Chapter 3, this section and all sections covering PLDs and HDLs may be skipped if desired.

[^2]:    REVIEW QUESTIONS

[^3]:    *Answers to problems marked with an asterisk can be found in the back of the text.

