CHAPTER

6

# DESIGNING SEQUENTIAL LOGIC CIRCUITS

*Implementation techniques for flip-flops, latches, oscillators, pulse generators, and Schmitt triggers*

■

*Static versus dynamic realization*

■

*Avoiding races in sequential circuits*

6.1 Introduction

6.2 Static Sequential Circuits

6.2.1 Bistability

6.2.2 Flip-Flop Classification

6.2.3 Master-Slave and Edge-Triggered FFs

6.2.4 CMOS Static Flip-Flops

6.2.5 Bipolar Static Flip-Flops

6.3 Dynamic Sequential Circuits

6.3.1 The Pseudostatic Latch

6.3.2 The Dynamic Two-Phase Flip-Flop

6.3.3 The C2MOS Latch

6.3.4 NORA-CMOS—A Logic Style for Pipelined Structures

6.3.5 True Single-Phase Clocked Logic (TSPCL)

6.4 Non-Bistable Sequential Circuits

6.4.1 The Schmitt Trigger

6.4.2 Monostable Sequential Circuits

6.4.3 Astable Circuits

6.5 Perspective: Choosing a Clocking Strategy

## 6.1 Introduction

The logic circuits described in Chapters 4 and 5 all have one property in common—the outputs are a logic combination of the **current** input signals. This chapter presents another class of logic circuits called *sequential logic* circuits. In these circuits, the inputs not only depend on the current values of the inputs, but also on **preceding** input values. In other words, a sequential circuit remembers some of the past history of the system—it has memory.

We can implement this memory function in a circuit in two ways. The first approach uses *positive feedback*, or regeneration. Here, one or more output signals are intentionally connected back to the inputs. This results in a class of elements called *multivibrator circuits*. The bistable element, or flip-flop, is its most popular representative, but other elements such as monostable and astable circuits are also frequently used.

A second approach is to use *charge storage* as a means to store signal values. This approach, which is very popular in the MOS world, requires regular refreshing, as charge tends to leak away with time. It is therefore called *dynamic*, in contrast with the regenerative approach, with which a signal value can be held indefinitely and is therefore called *static*.

The bistable elements, both static and dynamic, add an invaluable class of hardware modules to the library, namely the register. This element, which allows for the storage of previous signal values, is an essential component in the design of any digital processor. An elaborate discussion of the design and analysis of bistable sequential circuits is therefore appropriate and constitutes the core of this chapter. Other multivibrators, although less common, are often useful as well. The astable multivibrator acts as an oscillator and can be used as a clock generator, while the monostable multivibrator serves as a pulse generator. A discussion of both of those elements is included at the end of the chapter.

## 6.2 Static Sequential Circuits

### 6.2.1 Bistability

Two inverters connected in cascade are shown in Figure 6.1a, along with a voltage-transfer characteristic typical of such a circuit. Shown plotted are the VTCs of the first inverter, that is, $V_{o1}$ versus $V_{i1}$, and the second inverter ($V_{o2}$ versus $V_{o1}$). The latter plot is rotated to accentuate that $V_{i2} = V_{o1}$. Assume now that the output of the second inverter $V_{o2}$ is connected to the input of the first $V_{i1}$, as shown by the dotted lines in Figure 6.1a. The resulting circuit has only three possible operation points (*A*, *B*, and *C*), as demonstrated on the combined VTC. The following important conjecture is easily proven to be valid:

Under the condition that the gain of the inverter in the transient region is larger than 1, only *A* and *B* are stable operation points, and *C* is a metastable operation point.

This condition holds for every inverter we have discussed in previous chapters. Suppose that the cross-coupled inverter pair is biased at point *C*. A small deviation from this bias

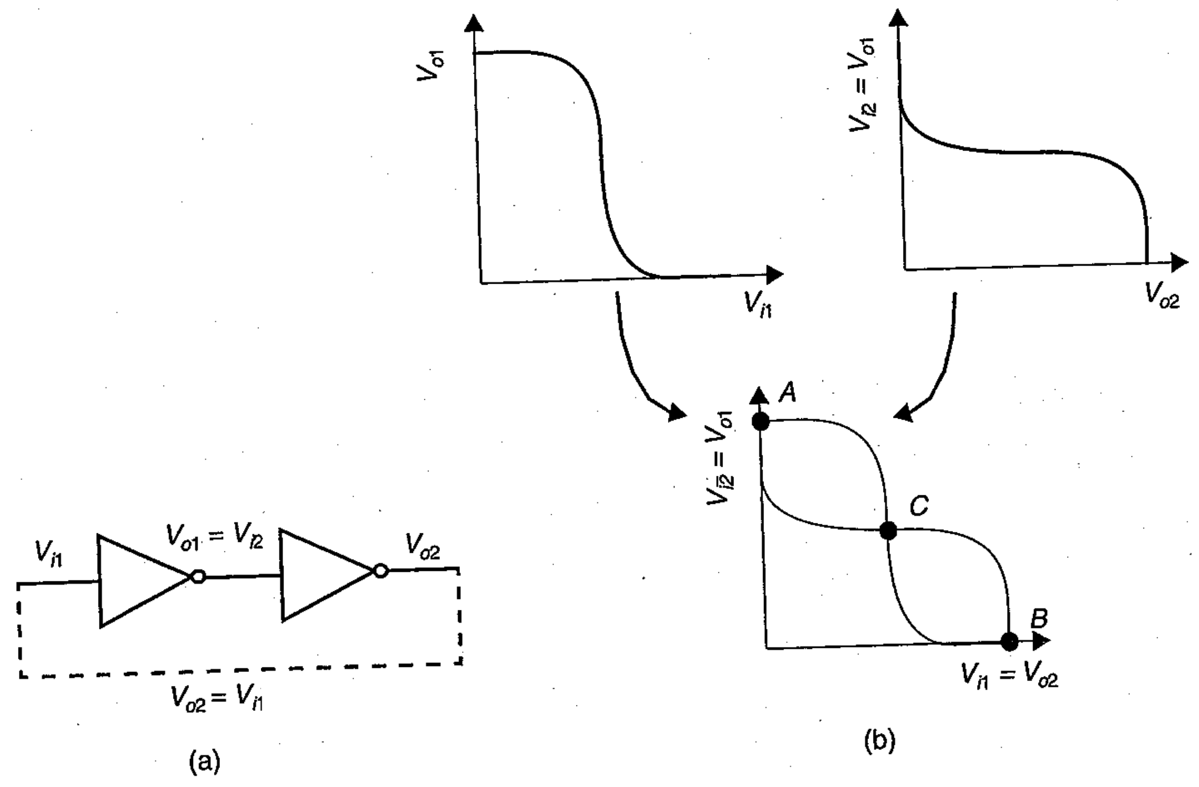


**Figure 6.1** Two cascaded inverters (a) and their VTCs (b).

point, possibly caused by noise, is amplified and *regenerated* around the circuit loop. This is a consequence of the gain around the loop being larger than 1. The effect is demonstrated in Figure 6.2a. A small deviation $\delta$ is applied to $V_{i1}$ (biased in $C$). This deviation is amplified by the gain of the inverter. The enlarged divergence is applied to the second inverter and amplified once more. The bias point moves away from $C$ until one of the operation points $A$ or $B$ is reached. In conclusion, $C$ is an unstable operation point. Every deviation (even the smallest one) causes the operation point to run away from its original bias. The chance is indeed very small that the cross-coupled inverter pair is biased at $C$ and stays there. Operation points with this property are termed *metastable*.

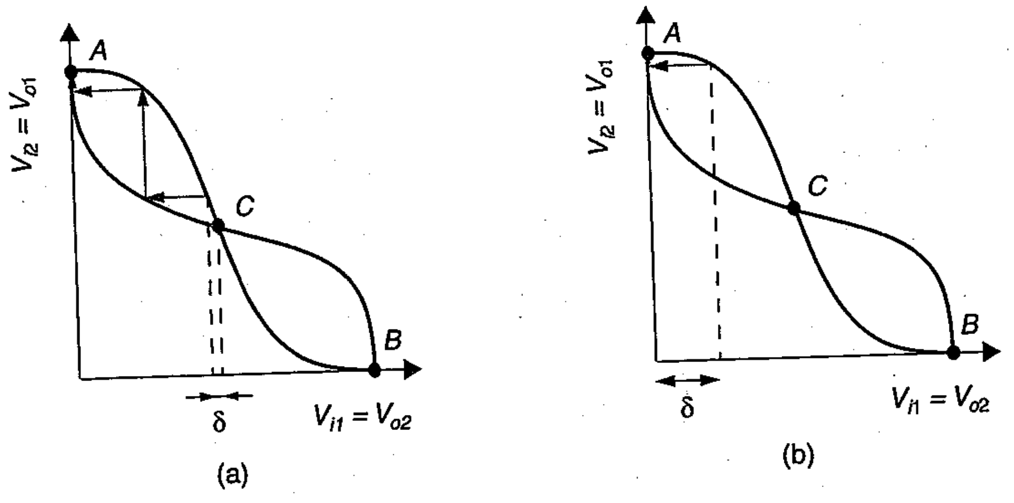


**Figure 6.2** Metastability.

On the other hand, $A$ and $B$ are stable operation points, as demonstrated in Figure 6.2b. In these points, the **loop gain is much smaller than unity.** Even a rather large deviation from the operation point is reduced in size and disappears.

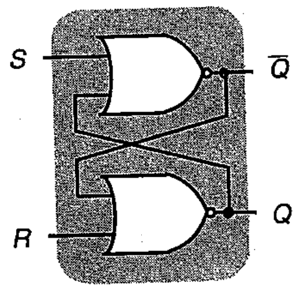
Hence the cross-coupling of two inverters results in a *bistable* circuit, that is, a circuit with two stable states, each corresponding to a logic state. The circuit serves as a memory, storing either a 1 or a 0 (corresponding to positions $A$ and $B$).

In order to change the stored value, we must be able to bring the circuit from state $A$ to $B$ and vice-versa. Since the precondition for stability is that the loop gain $G$ is smaller than unity, we can achieve this by making $A$ (or $B$) temporarily unstable by increasing $G$ to a value larger than 1. This is generally done by applying a trigger pulse at $V_{i1}$ or $V_{i2}$. For instance, assume that the system is in position $A$ ($V_{i1} = 0$, $V_{i2} = 1$). Forcing $V_{i1}$ to 1 causes both inverters to be on simultaneously for a short time and the loop gain $G$ to be larger than 1. The positive feedback regenerates the effect of the trigger pulse, and the circuit moves to the other state ($B$ in this case). The width of the trigger pulse need be only a little larger than the total propagation delay around the circuit loop, which is twice the average propagation delay of the inverters.

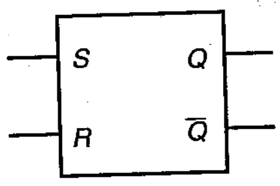
In summary, a bistable circuit has two stable states. In absence of any triggering, the circuit remains in a single state (assuming that the power supply remains applied to the circuit), and hence remembers a value. A trigger pulse must be applied to change the state of the circuit. Another common name for a bistable circuit is *flip-flop*.

### 6.2.2 Flip-Flop Classification

In the logic design world, a number of different flip-flop (FF) types are known. The simplest one is the *SR flip-flop* (or set-reset flip-flop). One possible implementation, using only NOR gates, is shown in Figure 6.3a. The logic symbol for this circuit is given in Figure 6.3b. This circuit is similar to the cross-coupled inverter pair with the inverters replaced by NOR gates. The second input of the NOR gates is connected to the trigger inputs ($S$ and $R$), which makes it possible to force the outputs $Q$ and $\overline{Q}$ to a given state. These outputs are complimentary. When both $S$ and $R$ are 0, the flip-flop is in a quiescent state and both outputs retain their value. If a positive (or 1) pulse is applied to the $S$ input, the $Q$ output is forced into the 1 state (with $\overline{Q}$ going to 0). Vice versa, a 1 pulse on $R$ resets the flip-flop and the $Q$ output goes to 0. The length of the trigger pulse has to be larger than the loop delay of the cross-coupled pair, as we have already noted.



(a) Schematic diagram



(b) Logic symbol

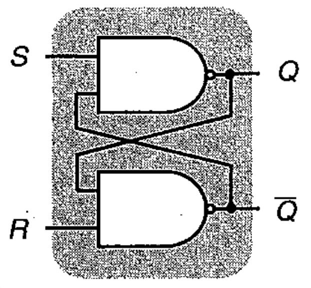
| $S$ | $R$ | $Q$ | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | $Q$ | $\overline{Q}$ |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 |

(c) Characteristic table

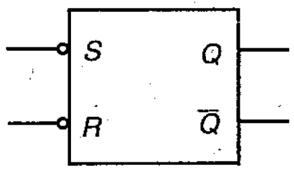
**Figure 6.3** NOR-based *SR* flip-flop.

These results are summarized in the *characteristic table* of the flip-flop, shown in Figure 6.3c. The characteristic table is the truth table of the gate and lists the output states as functions of all possible input conditions. The first three input combinations of the table have already been discussed. When both $S$ and $R$ are high, both $Q$ and $\overline{Q}$ are forced to zero. Since this does not correspond with our constraint that $Q$ and $\overline{Q}$ must be complementary, this input mode is considered to be forbidden. The problem with this operation condition is that when the input triggers return to their zero levels, the resulting state of the latch is unpredictable and depends on whatever input is last to go low.

The $SR$ flip-flop can also be implemented with NAND gates, as shown in Figure 6.4. As we can deduce from the circuit diagram and the characteristic table, the quiescent state of the latch corresponds to both $S$ and $R$ high. A negative-going (or 0) pulse on $S$ or $R$ respectively sets or resets the flip-flop. Having both $S$ and $R$ equal to zero is forbidden. The small circles at the inputs of the NAND-gate $SR$ flip-flop indicate that the gate is triggered by a negative-going pulse, or operates on so-called *negative logic*. This is in contrast with the NOR-based FF, which triggers on positive-going pulses, and is therefore said to operate on *positive logic*.



(a) Schematic diagram



(b) Logic symbol

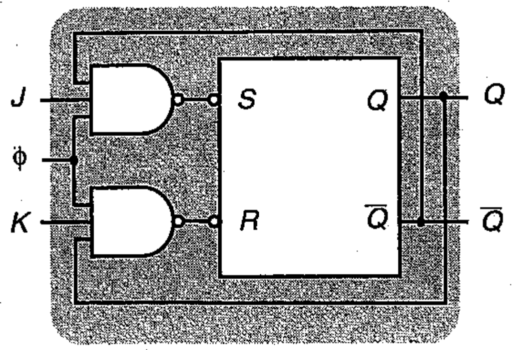
| $S$ | $R$ | $Q$ | $\overline{Q}$ |
|---|---|---|---|
| 1 | 1 | $Q$ | $\overline{Q}$ |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |

(c) Characteristic table

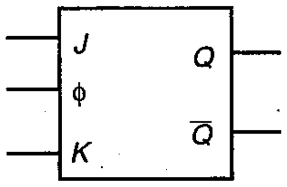
**Figure 6.4** NAND-based *SR* flip-flop

The ambiguity of having a nonallowed mode caused by trigger pulses going active simultaneously can be circumvented by adding two feedback lines to the circuit. The resulting device is called the *JK flip-flop*. An all-NAND version is shown in Figure 6.5a.

An important addition is the *clock input* $\phi$. This ensures that changes in the output logic states of the flip-flops in a design are synchronized with each other. A circuit in which all changes in state are related to a change in a clock signal (or a number of clock



(a) Schematic diagram



(b) Logic symbol

| $J_n$ | $K_n$ | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | $Q_n$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Q}_n$ |

(c) Characteristic table

**Figure 6.5** *JK* flip-flop.

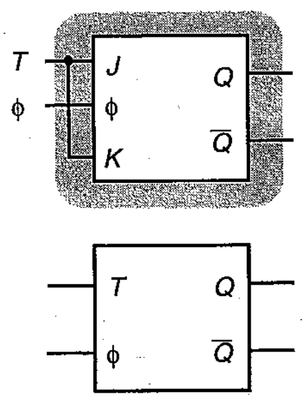
signals) is called a *synchronous* circuit. The majority of the currently designed circuits belong to this class.

The logic symbol and characteristic table of the *JK* flip-flop are given in Figure 6.5b and c. The setting and resetting of the flip-flop is performed as for the *SR* FF, except that these events are now synchronized with respect to the clock by way of the two NAND gates at the inputs of the FF. As long as the clock $\phi$ is low, the $S$ and $R$ inputs of the FF are both at 1, and the state of the flip-flop remains unchanged regardless of the values of the $J$ and $K$ inputs. Input-signal events can only propagate to the output when the clock is high. Another major difference between the *SR* and *JK* flip-flops is that the forbidden state is eliminated. When both $J$ and $K$ inputs are high and the clock $\phi$ goes high, the feedback of the output values $Q$ and $\overline{Q}$ causes the flip-flop to toggle its state. For instance, when the flip-flop is in the set state ($\overline{Q} = 0$, $Q = 1$), only $R$ goes low as a result of the feedback, and the flip-flop moves to the reset state.

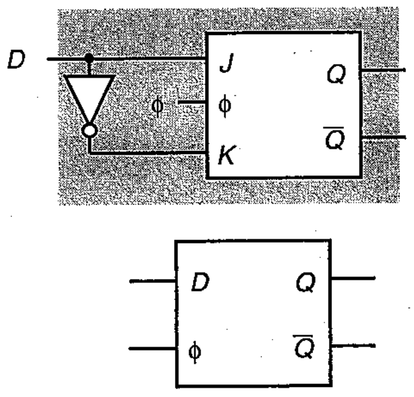
There is a catch, however. In the above example, the feedback disables the $K$ and enables the $J$ input after toggling. If the clock is still high, this causes the flip-flop to change state again. This repeated and unwanted toggling puts some stringent constraints on the clock pulse width. To function correctly, the pulse width of the clock has to be less than the propagation delay of the FF. This restriction is removed in other FF structures.

The characteristic table of the *JK* FF is similar to the one of the *SR* FF, but the forbidden mode is now replaced by the toggle mode. This is expressed in the characteristic table by stating that the next value of $Q$ (denoted as $Q_{n+1}$) is the reverse of the current value of $Q$ (indicated as $Q_n$). Note that input changes can only propagate when the clock $\phi$ is high, or in other words, when the inputs are synchronized.

Some derived forms of the *JK* FF are commonly used and have received proper names. One is the *toggle* or *T flip-flop*. This is a *JK* FF where the only allowed operation mode is the toggle operation, which can be achieved by tying both $J$ and $K$ permanently together, as shown in Figure 6.6a. Another special FF is the *delay* or *D flip-flop,* which is used copiously in digital systems for the temporary storage of data. Here, the $Q$ output of the gate is a replica of the $D$ input. An inverter between the $S$ and $R$ inputs of the FF ensures that the inputs are always complementary Figure 6.6(b). Therefore, a 1 input results in a setting of the FF, while a 0 resets the device. As suggested by its name, the D-FF simply generates a delayed version of the input signal that is synchronized with the clock $\phi$.



(a) *T* flip-flop



(b) *D* flip-flop

**Figure 6.6** Derived flip-flops.

### 6.2.3 Master-Slave and Edge-Triggered FFs

The *JK* flip-flop presented above is also called a *latch*. A flip-flop is a latch if the gate is transparent while the clock is high (low) [Hill74]. Any change at the input is reflected at the output after a nominal delay. The latch is said to open with the rising of the clock. Data is accepted continuously until the clock goes down and the latch closes.

The transparent nature of the latch can cause some severe problems. Consider the simple circuit of Figure 6.7. As long as the clock is high, the output of the flip-flop oscillates back and forth between the 0 and the 1 states. This phenomenon is called a *race* (or *race-around)* condition and can only be avoided by making the pulse width of $\phi$ smaller than the propagation delay of the loop. Since the loop delay in the example is small, and probably smaller than the pulse width, this situation has a major chance of occurring. The result of this repetitive toggling is that the output is undetermined when the clock goes low. Observe that a *JK* flip-flop has an intrinsic race problem when *J* and *K* are high, as discussed earlier.

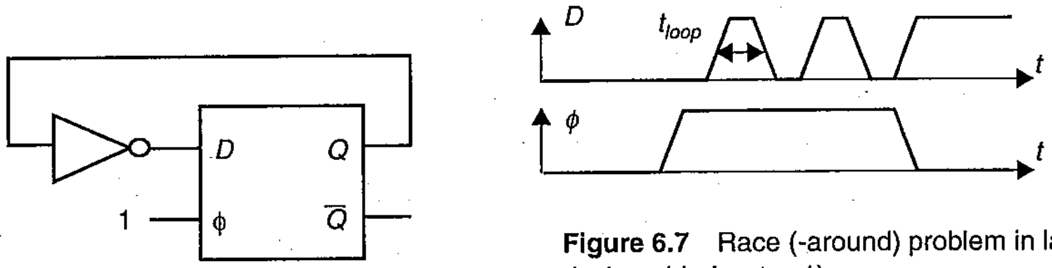
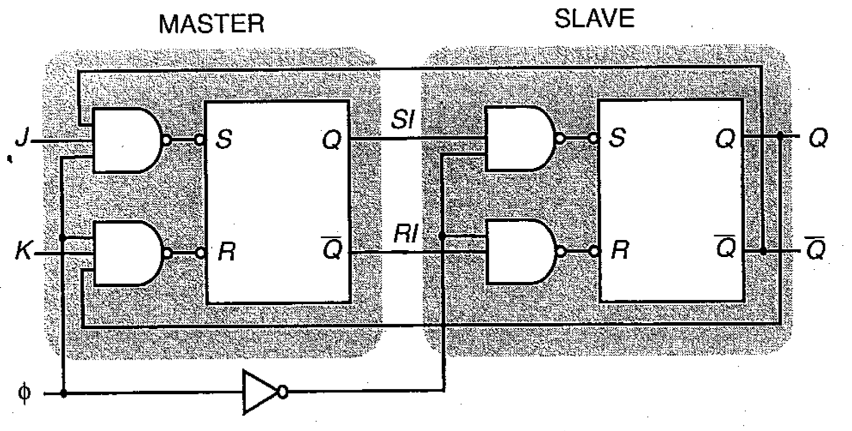


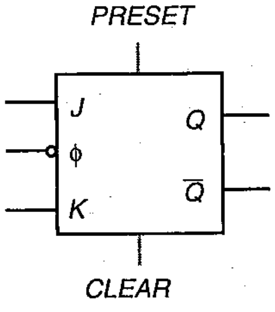
**Figure 6.7** Race (-around) problem in latch-based designs (during $\phi = 1$).

One way to avoid a race is to use the *master-slave* approach. Master-slave flip-flops are built by cascading two basic flip-flops with opposite clock phases, as illustrated in Figure 6.8. The first flip-flop, called the master, becomes operational when the clock $\phi$ goes high. During this period, inputs *J* and *K* are enabled, and the intermediate signals *SI* and *RI* can be changed. The feedback of the *Q* and $\overline{Q}$ signals ensures that the flip-flop acts as a *JK* flip-flop that toggles when both *J* and *K* are high. The $\overline{\phi}$ clock is low in this time period, putting the second FF, called the slave, in the *hold* mode.This prevents the changes in *SI* and *RI* from propagating to the *Q* and $\overline{Q}$ outputs. On the falling edge of the clock, $\phi$ goes down, freezing the state of the master latch. A very short time later, the NAND input gates of the slave latch are enabled, and the changes in *SI* and *RI* are propagated to the outputs. Due to this master-slave operation, there is no limitation on the width of the clock pulse, since the master latch is disabled at the time the outputs are changing, and no racing around can occur. In other words, the master-slave principle makes sure that the feedback path of the flip-flop is always interrupted either at the master or slave side. The pulse lengths of $\phi$ and $\overline{\phi}$ have to be larger than the propagation delays of the master (slave) latches for this structure to be functional.

The logic symbol of the master-slave flip-flop is given in Figure 6.8b. The little circle on the $\phi$ input indicates that the state of the flip-flop changes at the falling edge of the clock. Very often, the basic flip-flop circuit is extended with some additional inputs, as indicated by gray lines on the symbol of Figure 6.8b (not included in the logic diagram). These are the *asynchronous* inputs, which can change the state of the flip-flop regardless



(a) Schematic diagram



(b) Logic symbol

**Figure 6.8** Master-slave flip-flop.

of the clock state. Typical direct inputs are the *PRESET* and *CLEAR* inputs that initialize the flip-flop state to either 1 or 0.

A problem with the master-slave approach is that the circuit is sensitive to changes in the input signals as long as $\phi$ is high. In other words, the input signals must stay constant when the clock is active. Assume that the flip-flop is reset ($\overline{Q} = 1$). This means that the $J$-input NAND gate of the master FF is enabled. Any *spike* or *glitch* on this input, possibly caused by the switching of a neighboring signal, might cause the master latch to be set. Bringing the $J$ input back to zero does not change the state of the latch, because it can only be reset by a pulse on the $K$ input, which is currently disabled! Thus one can say that the $J$ input has "caught" a 1 that is subsequently transferred to the slave when $\phi$ goes low. This problem is therefore known as *one-catching,* or *level-sensitive,* and it can be avoided by keeping the length of the $\phi$ pulse as small as possible. This might not always be possible due to constraints imposed by the rest of the system. One way to circumvent this problem is to make use of *edge-triggered* devices.

The idea behind the edge-triggered approach is to allow the state of the flip-flop to change only at the rising (or falling) edge of the clock. Input events at other points in time are ignored, so that one-catching is avoided. This is generally achieved by ensuring that the $S$ and $R$ pulses, as presented to the actual latch, are of a controlled and narrow width and occur synchronously with a clock transition. The width of the pulses can be manipulated in many different ways, all of which require a timing element as a reference. The delay of an $RC$ network or the propagation delay of a logic gate are commonly used timing references.

Consider, for instance, the circuit of Figure 6.9a. When $\phi$ is high, the output of gate $N_2$ is always 1 (independent of *In*), which means that the gate is disabled. During the same period, the input *In* can propagate through $N_1$. As shown on the waveform diagram of Figure 6.9b, a high input causes $X$ to go low. On the falling edge of $\phi$, $N_1$ is disabled and $N_2$ enabled. $X$ is forced to 1. This happens only after a period of time equal to the propagation delay of $N_1$. During a short time interval, $N_2$ sees both of its inputs at 0 and goes low. $X$ eventually reaches the 1 value, and *Out* goes back to 1. Consequently, a short, low-going pulse appears at the output of $N_2$ with a length approximately equal to the propagation delay of $N_1$.

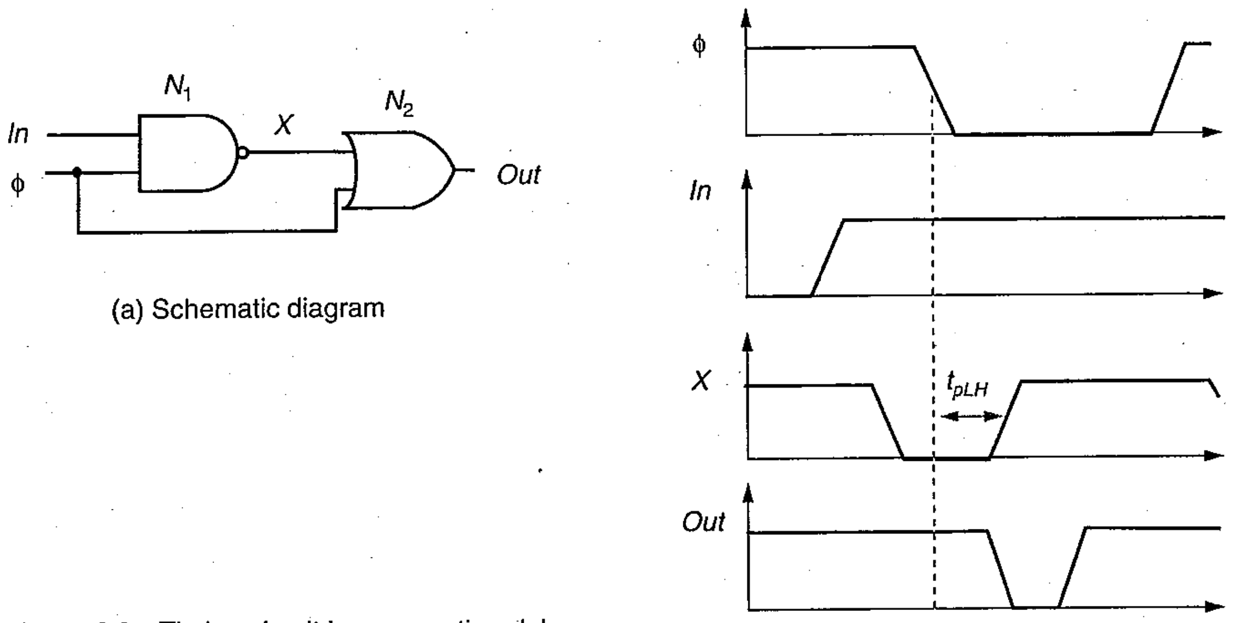
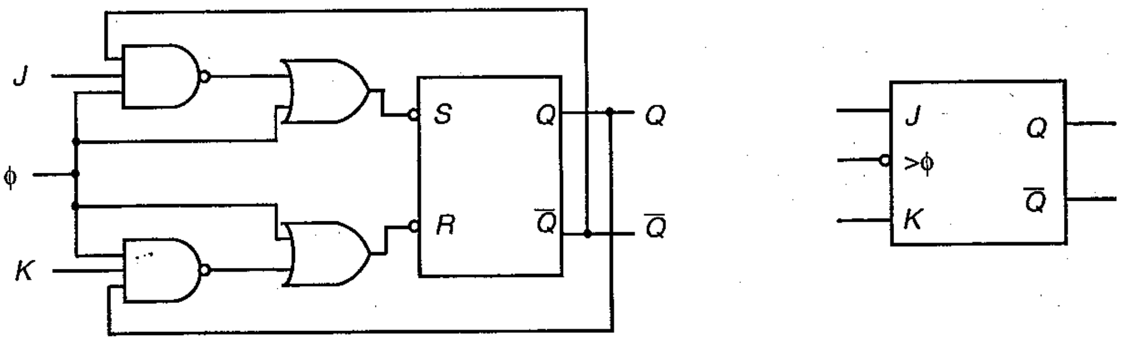


**Figure 6.9** Timing circuit in propagation-delay-based, edge-triggered flip-flop.

(b) Timing diagram

This idea is applied in the edge-triggered *JK* flip-flop of Figure 6.10. The values of *J* and *K* are sampled at the low-going edge of $\phi$ and generate short pulses at the *S* or *R* inputs of the latch. The state of the flip-flop thus only changes when the clock goes low, and the output state is determined by the values of the *J* and/or *K* inputs just prior to the clock going low.



(a) Schematic diagram (b) Logic symbol

**Figure 6.10** Negative edge-triggered *JK* flip-flop.

In order to function correctly, the edge-triggered flip-flop requires the inputs to be stable some time before the clock goes low. This period is called the *set-up time* of the FF and in this case is approximately equal to the propagation delay of the input gate. Other flip-flop designs also require the state of the *J* and *K* lines to be held for some time after the clock edge, known as the *hold* time. The definitions of set-up and hold times are illustrated in the timing diagram of Figure 6.11. The performance of an FF is furthermore qualified by its *propagation delay* $t_{pFF}$, which equals the time it takes for the output to change after the occurrence of a clock edge.

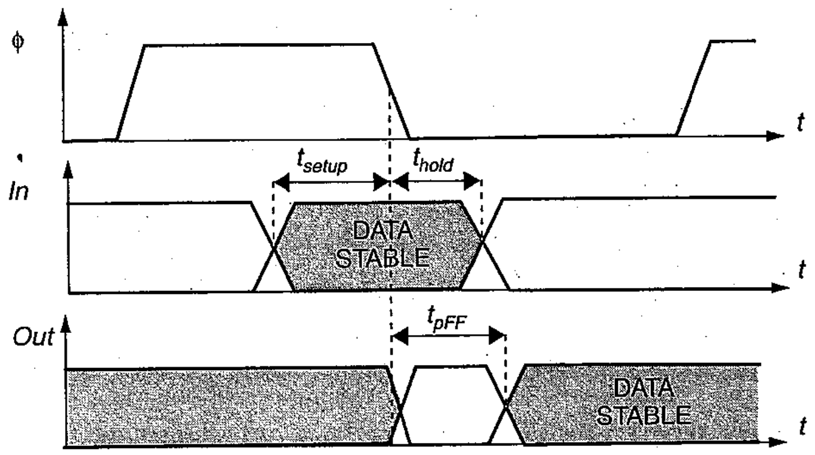


Figure 6.11 Definition of set-up time, hold time, and propagation delay of a flip-flop.

This collection of timing parameters helps to determine the speed of a sequential machine, which consists of a block of combinational logic and a number of (edge-triggered) flip-flops storing the state, as in Figure 6.12. With $t_{p,comb}$ the longest propagation delay through the combinational network, the following relation must hold for the circuit to operate correctly:

$$t_{pFF} + t_{p,comb} + t_{setup} < T \tag{6.1}$$

where $T$ is the period of the clock-signal. Eq. (6.1) determines the maximum clock speed for which the circuit still performs correctly.

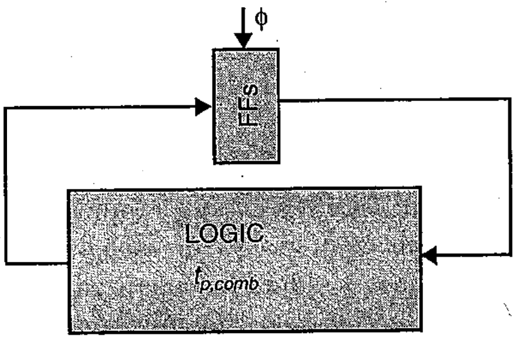


Figure 6.12 Maximum speed of sequential machine using edge-triggered FFs.

The logic symbol of the edge-triggered flip-flop is shown in Figure 6.10b. The symbol is differentiated from the master-slave by the small > sign at the $\phi$ input. The presence or absence of a small circle at the clock input indicates whether the outputs change on the falling or the rising edge of the clock. The former case is called *negative edge-triggered* (with dot), while the latter is called *positive edge-triggered.*

The subsequent sections discuss the implementation of the static flip-flop in both the CMOS and bipolar technologies.

### 6.2.4 CMOS Static Flip-Flops

The simplest way to implement the static flip-flop structures introduced in the previous sections would be to replace the gates by their static CMOS implementations. This produces complex compositions that use an excessive numbers of transistors, and are in general large and slow. Observe from Eq. (6.1) that the FF set-up and propagation times have a direct impact on the global circuit performance. As a reference, the straightforward implementation of the master-slave flip-flop of Figure 6.8 in complementary CMOS consumes 38 transistors.

More efficient implementations are obviously preferable. One possible realization of a clocked SR flip-flop is given in Figure 6.13. It essentially consists of a cross-coupled inverter pair. The extra transistors are used to drive the FF from one stable state to the other upon application of the $S$ or $R$ pulses and a high clock signal. The transistors of the FF must be dimensioned carefully, or the latch might not be switchable. Consider the case where $Q$ is high and an $R$ pulse is applied. The combination of transistors $M_4$, $M_7$, and $M_8$ forms a ratioed inverter. In order to make the FF switch, we must succeed in bringing $Q$ below the switching threshold of the inverter $M_1$-$M_2$. Once this is achieved, the positive feedback causes the flip-flop to invert states. This requirement forces us to increase the sizes of transistors $M_5$, $M_6$, $M_7$, and $M_8$.

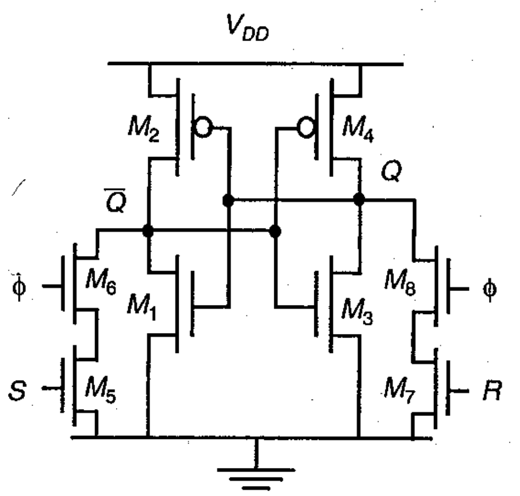


**Figure 6.13** CMOS clocked *SR* flip-flop.

It is important to realize that this flip-flop does not consume any static power. Due to the feedback, one inverter is in the high state, while the other resides in the low state. No static paths between $V_{DD}$ and $GND$ can exist except during switching.

**Example 6.1 Transistor Sizing of Clocked SR Flip-Flop**

Assume that the cross-coupled inverter pair is designed with minimum-size transistors, and that those transistors are scaled so that the inverter threshold $V_M$ is located at $V_{DD}/2$. For the 1.2 μm CMOS technology, the following transistor sizes were selected: $(W/L)_{M1} = (W/L)_{M3} = (1.8/1.2)$, and $(W/L)_{M2} = (W/L)_{M4} = (5.4/1.2)$.

Assuming $Q = 1$, determine the minimum sizes of $M_5$, $M_6$, $M_7$, and $M_8$ to make the device switchable.

To switch the FF from the $Q = 1$ to the $Q = 0$ state, it is essential that the low level of the ratioed, pseudo-NMOS inverter $(M_7$-$M_8)$-$M_4$ be below the switching threshold of the inverter $M_1$-$M_2$ that equals $V_{DD}/2$. It is reasonable to assume that as long as $V_Q > V_M$, $V_{\overline{Q}}$ equals 0, and the gate of transistor $M_4$ is grounded. The boundary conditions on the transistor sizes can be derived by equating the currents in the inverter for $V_Q = V_{DD}/2$, as given in Eq. (6.2). We assume that $M_7$ and $M_8$ have identical sizes. Under that condition, the pull-down network can be modeled by a single transistor $M_{78}$, whose length is twice the length of the individual devices.

$$k_{n,M78}\left((V_{DD}-V_{Tn})\frac{V_{DD}}{2}-\frac{V_{DD}^2}{8}\right) = k_{p,M4}\left((V_{DD}-|V_{Tp}|)\frac{V_{DD}}{2}-\frac{V_{DD}^2}{8}\right) \quad (6.2)$$

For $V_{Tn} \approx |V_{Tp}|$, this results in the condition that $k_{n,M78} \geq k_{p,M4}$, which translates to the following size constraint:

$$(W/L)_{M78} \geq (\mu_p/\mu_n)\,(W/L)_{M4} = (W/L)_{M3}$$

The latter part of the expression stems from the requirement that in order to set $V_M$ of the inverter $M_3$-$M_4$ to $V_{DD}/2$, it is necessary to make the PMOS device $M_4$ $(\mu_n/\mu_p)$ times wider than the NMOS transistor $M_3$. This finally results into minimum size requirements for $M_7$ and $M_8$ (and by considering symmetry, $M_5$ and $M_6$).

$$(W/L)_{M7} = 2(W/L)_{M78} \geq 2(W/L)_{M3}$$

or $(W/L)_{M7} \geq (3.6/1.2)$.

Figure 6.14 plots the simulated voltage-transfer characteristic ($V_Q$ versus $V_R$) of the FF for various values of the $(W/L)$ ratios of transistors $M_7$-$M_8$. Observe that the flip-flop barely switches for $(W/L) = (3.6/1.2)$. However, once the switching point is reached ($V_R \approx 4.7$ V), the device switches very abruptly. This is due to the positive feedback that comes into effect even when the inverter $M_1$-$M_2$ is turned on ever so lightly. This positive feedback effect was ignored in our manual analysis, which explains the small deviation between the predicted minimum size (3.6/1.2) versus the simulated one (3.3/1.2). The simulation further demonstrates that a device size of (1.8/1.2) fails to switch the gate, while a larger device size of (7.2/1.2) places the switching point somewhere in the middle of the voltage range. This is clearly the desirable solution, as it maximizes the noise margins and increases the switching speed.

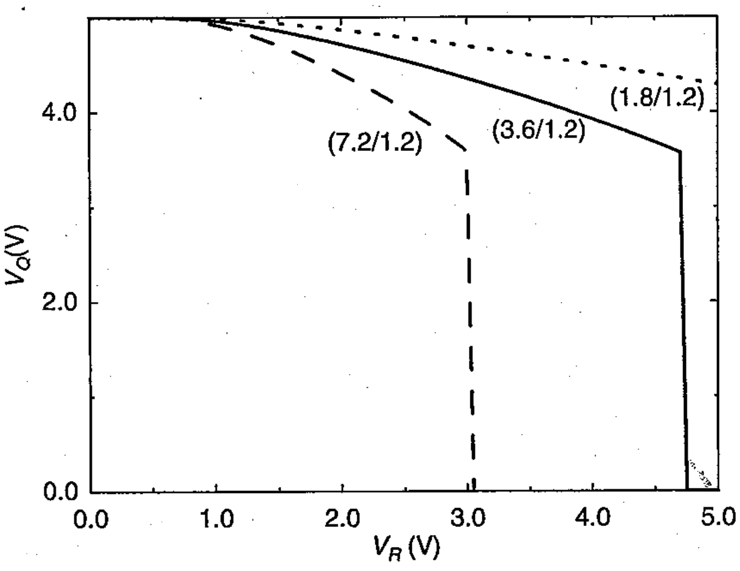


**Figure 6.14** Simulated voltage-transfer characteristic of *SR* FF ($V_Q$ versus $V_R$).

The positive feedback effect makes a manual derivation of propagation delay of the *SR* FF (and other FFs) nontrivial. Some simplifications are therefore necessary. Consider,