# CHAPTER 8

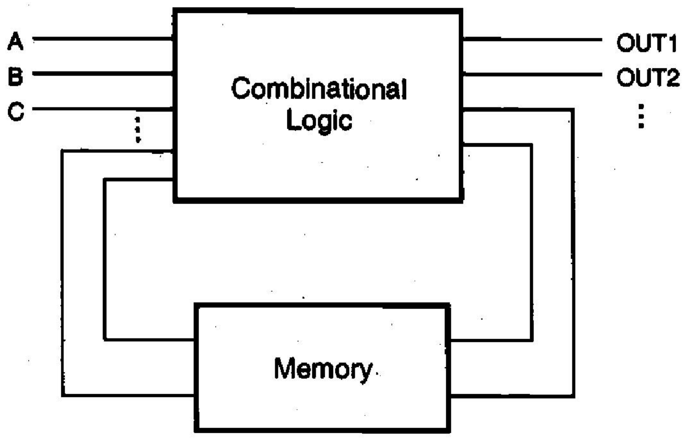
# SEQUENTIAL MOS LOGIC CIRCUITS

## 8.1. Introduction

In all of the combinational logic circuits examined in Chapter 7, if we neglect the propagation delay time, the output levels at any given time point are directly determined as Boolean functions of the input variables applied at that time. Thus, the combinational circuits lack the capability of storing any previous events, or displaying an output behavior which is dependent upon the previously applied inputs. Circuits of this type are also classified as non-regenerative circuits, since there is no feedback relationship between the output and the input.

The other major class of logic circuits is called sequential circuits, in which the output is determined by the current inputs as well as the previously applied input variables. Figure 8.1(a) shows a sequential circuit consisting of a combinational circuit and a memory block in the feedback loop. In most cases, the regenerative behavior of sequential circuits is due to either a direct or indirect feedback connection between the output and the input. Regenerative operation can, under certain conditions, also be interpreted as a simple memory function. The critical components of sequential systems are the basic regenerative circuits, which can be classified into three main groups: bistable circuits, monostable circuits, and astable circuits. The general classification of non-regenerative and regenerative logic circuits is shown in Fig. 8.1.

(a)



(b)

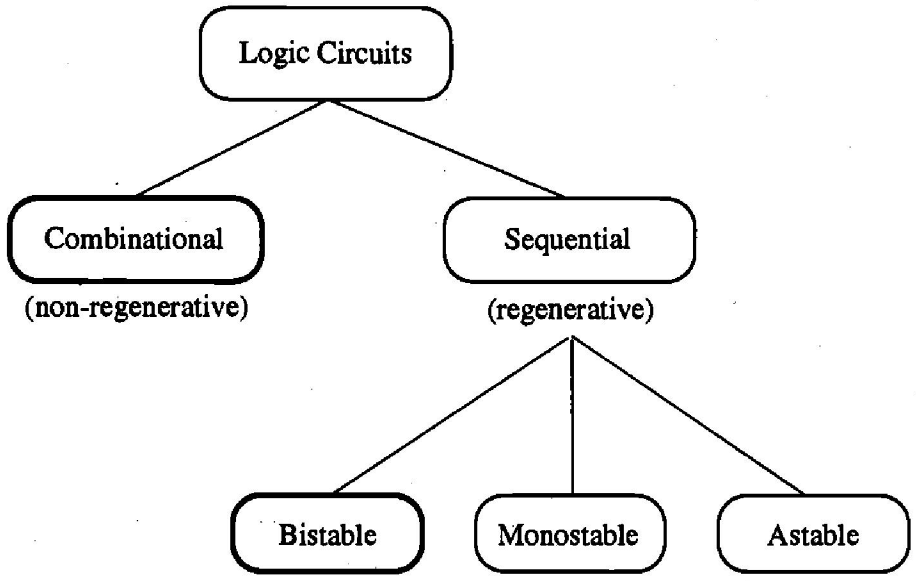


***Figure 8.1.*** (a) Sequential circuit consisting of a combinational logic block and a memory block in the feedback loop. (b) Classification of logic circuits based on their temporal behavior.

Bistable circuits have, as their name implies, two stable states or operation modes, each of which can be attained under certain input and output conditions. Monostable circuits, on the other hand, have only one stable operating point (state). Even if the circuit experiences an external perturbation, the output eventually returns to the single stable state after a certain time period. Finally, in astable circuits, there is no stable operating point or state which the circuit can preserve for a certain time period. Consequently, the output of an astable circuit must oscillate without settling into a stable operating mode. The ring oscillator circuit examined in Chapter 6 would be a typical example of an astable regenerative circuit.

Among these three main groups of regenerative circuit types, the bistable circuits are by far the most widely used and the most important class. All basic latch and flip-flop circuits, registers, and memory elements used in digital systems fall into this category. In the following, we will first examine the electrical behavior of the simple bistable element, and then present some of its useful applications.

## 8.2. Behavior of Bistable Elements

The basic bistable element to be examined in this section consists of two identical cross-coupled inverter circuits, as shown in Fig. 8.2(a). Here, the output voltage of inverter (1) is equal to the input voltage of inverter (2), i.e., $v_{o1} = v_{i2}$, and the output voltage of inverter (2) is equal to the input voltage of inverter (1), i.e., $v_{o2} = v_{i1}$. In order to investigate the static input-output behavior of both inverters, we start by plotting the voltage transfer characteristic of inverter (1) with respect to the $v_{o1}$ - $v_{i1}$ axis pair. Notice that the input and output voltages of inverter (2) correspond to the output and input voltages of inverter (1), respectively. Consequently, we can also plot the voltage transfer characteristic of inverter (2) using the same axis pair, as shown in Fig. 8.2(b).

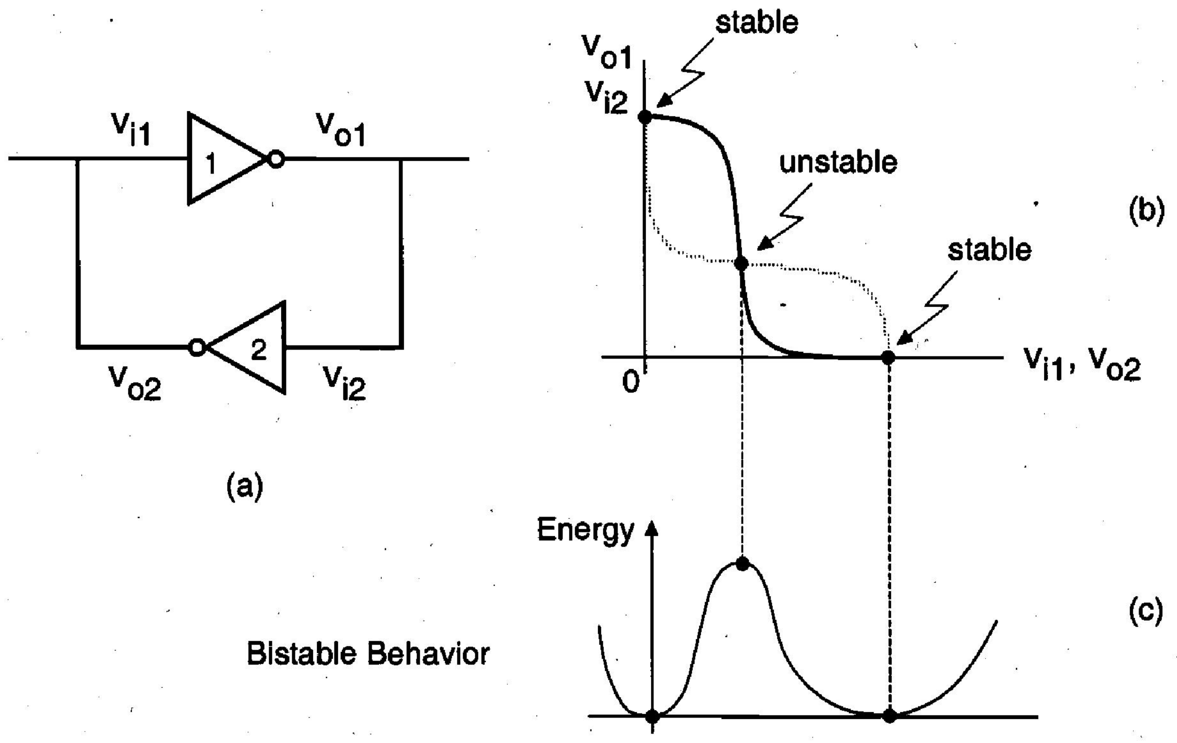


*Figure 8.2.* Static behavior of the two-inverter basic bistable element: (a) Circuit schematic. (b) Intersecting voltage transfer curves of the two inverters, showing the three possible operating points. (c) Qualitative view of the potential energy levels corresponding to the three operating points.

It can be seen that the two voltage transfer characteristics intersect at three points. Simple reasoning can help us to conclude that two of these operating points are stable, as indicated in Fig. 8.2(b). If the circuit is initially operating at one of these two stable points, it will preserve this state unless it is forced externally to change its operating point. Note that the gain of each inverter circuit, i.e., the slope of the respective voltage transfer curves, is smaller than unity at the two stable operating points. Thus, in order to change the state by moving the operating point from one stable point to the other, a sufficiently large external voltage perturbation must be applied so that the voltage gain of the inverter loop becomes larger than unity.

On the other hand, the voltage gains of both inverters are larger than unity at the third operating point. Consequently, even if the circuit is biased at this point initially, a small voltage perturbation at the input of any of the inverters will be amplified, causing the operating point to move to one of the stable operating points. This leads to the conclusion that the third operating point is unstable. The circuit has two stable operating points, hence, it is called bistable.

The bistable behavior of the cross-coupled inverter circuit can also be visualized qualitatively by examining the total potential energy level at each of the three possible operating points (Fig. 8.2(c)). It is seen that the potential energy is at its minimum at two of the three operating points, since the voltage gains of both inverters are equal to zero. By contrast, the energy attains a maximum at the operating point at which the voltage gains of both inverters are maximum. Thus, the circuit has two stable operating points corresponding to the two energy minima, and one unstable operating point corresponding to the potential energy maximum.

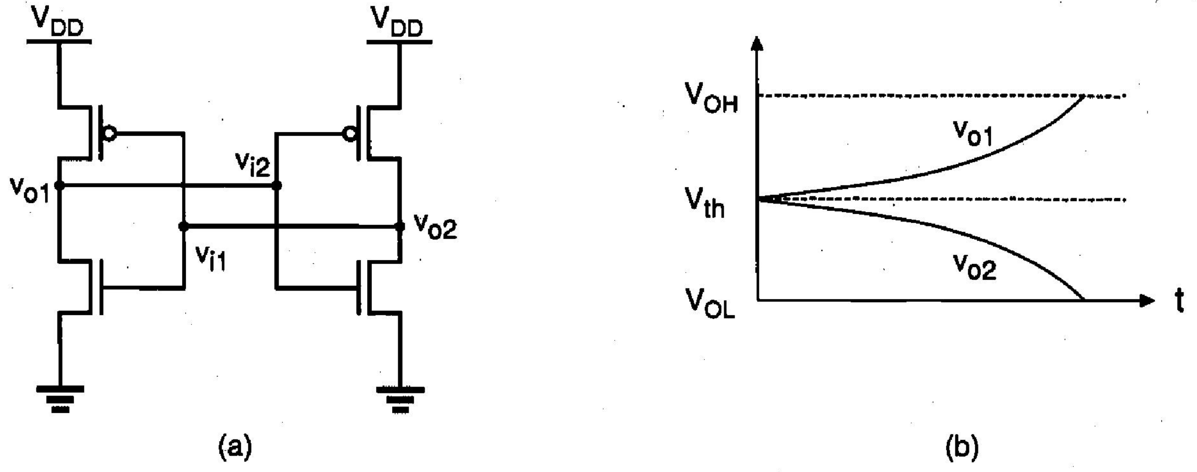


***Figure 8.3.*** (a) Circuit diagram of a CMOS bistable element. (b) One possibility for the expected time-domain behavior of the output voltages, if the circuit is initially set at its unstable operating point.

Figure 8.3(a) shows the circuit diagram of a CMOS two-inverter bistable element. Note that at the unstable operating point of this circuit, all four transistors are in saturation, resulting in maximum loop gain for the circuit. If the initial operating condition is set at this point, any small voltage perturbation will cause significant changes in the operating modes of the transistors. Thus, we expect the output voltages of the two inverters to diverge and eventually settle at $V_{OH}$ and $V_{OL}$, respectively, as illustrated in Fig. 8.3(b). The direction in which each output voltage diverges is determined by the initial perturbation polarity. In the following, we will examine this event more closely using a small-signal analysis approach.

Consider the bistable circuit shown in Fig. 8.4, which is initially operating at $v_{o1} = v_{o2} = V_{th}$, i.e., at the unstable operating point. For our analysis, we will assume that the input (gate) capacitance $C_g$ of each inverter is much larger than its output (drain) capacitance $C_d$, i.e., $C_g >> C_d$.

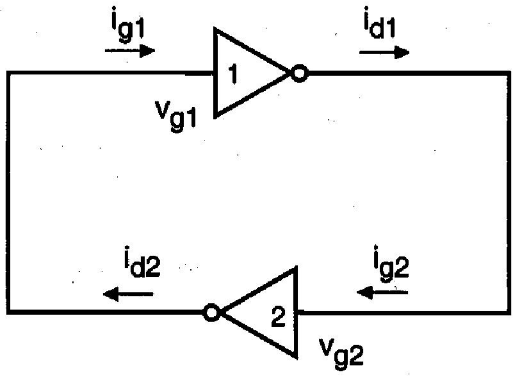


***Figure 8.4.*** Small-signal input and output currents of the inverters.

The small-signal drain current supplied by each inverter (1 and 2) can be expressed, in terms of the small-signal gate voltage of that inverter, as follows. Note that the drain current of each inverter is also equal to the gate current of the other inverter.

$$\begin{aligned} i_{g1} &= i_{d2} = g_m\, v_{g2} \\ i_{g2} &= i_{d1} = g_m\, v_{g1} \end{aligned} \tag{8.1}$$

Here, $g_m$ represents the small-signal transconductance of the inverter. The gate voltages of both inverters can be expressed in terms of the gate charges, $q_1$ and $q_2$.

$$v_{g1} = \frac{q_1}{C_g} \qquad v_{g2} = \frac{q_2}{C_g} \tag{8.2}$$

Note that the small-signal gate current of each inverter can be written as a function of the time derivative of its small-signal gate voltage, as follows.

$$\begin{aligned} i_{g1} &= C_g \frac{dv_{g1}}{dt} \\ i_{g2} &= C_g \frac{dv_{g2}}{dt} \end{aligned} \tag{8.3}$$

Combining (8.1) with (8.3), we obtain:

$$g_m\, v_{g2} = C_g \frac{dv_{g1}}{dt} \tag{8.4}$$

$$g_m\, v_{g1} = C_g \frac{dv_{g2}}{dt} \tag{8.5}$$

Expressing the gate voltages in terms of the gate charges, these two differential equations can also be written as

$$\frac{g_m}{C_g} q_2 = \frac{dq_1}{dt} \tag{8.6}$$

$$\frac{g_m}{C_g} q_1 = \frac{dq_2}{dt} \tag{8.7}$$

Both differential equations given in (8.6) and (8.7) can now be combined to yield a second-order differential equation describing the time behavior of gate charge $q_I$.

$$\frac{g_m}{C_g} q_1 = \frac{C_g}{g_m}\frac{d^2 q_1}{dt^2} \Rightarrow \qquad \frac{d^2 q_1}{dt^2} = \left(\frac{g_m}{C_g}\right)^2 q_1 \tag{8.8}$$

This equation can also be expressed in a more simplified form by using $\tau_0$, the transit time constant.

$$\frac{d^2 q_1}{dt^2} = \frac{1}{\tau_0{}^2} q_1 \qquad\qquad with\ \tau_0 = \frac{C_g}{g_m} \tag{8.9}$$

The time-domain solution of (8.9) for $q_I$ yields

$$q_1(t) = \frac{q_1(0) - \tau_0\, q_1'(0)}{2}\, e^{-\frac{t}{\tau_0}} + \frac{q_1(0) + \tau_0\, q_1'(0)}{2}\, e^{+\frac{t}{\tau_0}} \tag{8.10}$$

where the initial condition is given as:

$$q_1(0) = C_g \cdot v_{g1}(0) \tag{8.11}$$

Note that $v_{g1} = v_{o2}$ and $v_{g2} = v_{o1}$. Replacing the gate charge of both inverters with the corresponding output-voltage variables, we obtain,

$$v_{o2}(t) = \frac{1}{2}\left(v_{o2}(0) - \tau_0\, v_{o2}'(0)\right) e^{-\frac{t}{\tau_0}} + \frac{1}{2}\left(v_{o2}(0) + \tau_0\, v_{o2}'(0)\right) e^{+\frac{t}{\tau_0}} \tag{8.12}$$

$$v_{o1}(t) = \frac{1}{2}\left(v_{o1}(0) - \tau_0\, v_{o1}'(0)\right) e^{-\frac{t}{\tau_0}} + \frac{1}{2}\left(v_{o1}(0) + \tau_0\, v_{o1}'(0)\right) e^{+\frac{t}{\tau_0}} \tag{8.13}$$

For large values of $t$, the time-domain expressions (8.12) and (8.13) can be simplified as follows.

$$v_{o1}(t) \approx \frac{1}{2}\left(v_{o1}(0) + \tau_0\, v_{o1}'(0)\right) e^{+\frac{t}{\tau_0}}$$
$$v_{o2}(t) \approx \frac{1}{2}\left(v_{o2}(0) + \tau_0\, v_{o2}'(0)\right) e^{+\frac{t}{\tau_0}} \qquad (8.14)$$

Note that the magnitude of both output voltages increases exponentially with time. Depending on the polarity of the initial small perturbations $dv_{o1}(0)$ and $dv_{o2}(0)$, the output voltages of both inverters will diverge from their initial value of $V_{th}$ to either $V_{OL}$ or $V_{OH}$. In fact, the polarity of the output-voltage perturbation $dv_{o1}$ must always be opposite to that of $dv_{o2}$, because of the charge-conservation principle. Hence, the two output voltages always diverge into opposite directions, as expected.

$$v_{o1}: \quad V_{th} \rightarrow V_{OH} \text{ or } V_{OL}$$
$$v_{o2}: \quad V_{th} \rightarrow V_{OL} \text{ or } V_{OH} \qquad (8.15)$$

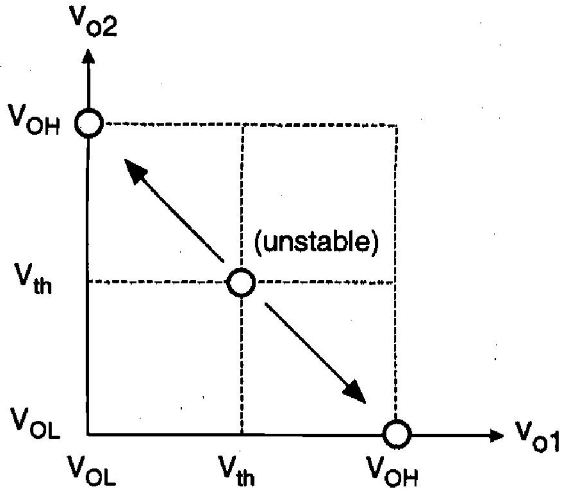


***Figure 8.5.*** Phase-plane representation of the bistable circuit behavior.

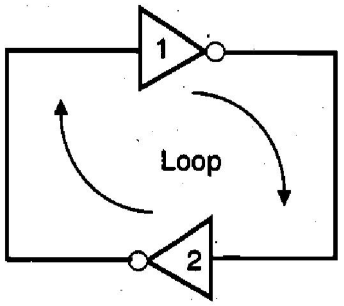
The phase-plane representation of this event, illustrated in Fig. 8.5, shows that the operating point ($v_{o1} = V_{th}$, $v_{o2} = V_{th}$) is unstable. The two operating points ($v_{o1} = V_{OL}$, $v_{o2} = V_{OH}$) and ($v_{o1} = V_{OH}$, $v_{o2} = V_{OL}$) can be shown to be stable, using small-signal models at the corresponding operating points.

In addition to the time-domain analysis presented here, an interesting observation can be made for the two-inverter bistable element: While the bistable circuit is settling from its unstable operating point into one of its stable operating points, we can envision a signal traveling the loop consisting of the two cascaded inverters several times (Fig.

8.6). The time-domain behavior of the output voltage $v_{o1}$ during this period is approximated as follows:

$$\frac{v_{o1}(t)}{v_{o1}(0)} = e^{+\frac{t}{\tau_0}} \tag{8.16}$$

If during a time interval $T$, the signal travels the loop $n$ times, then this is equivalent to the same signal propagating along a cascaded inverter chain consisting of $2n$ inverters. Expressing the loop gain (combined voltage gain of two cascaded inverters) with $A$, we obtain,



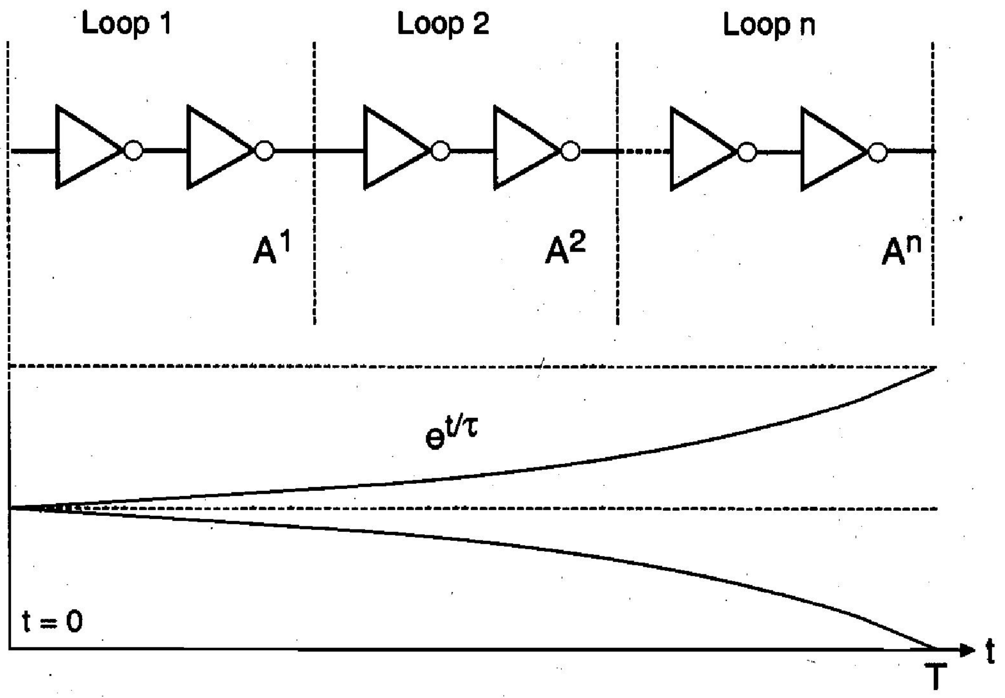


*Figure 8.6.* Propagation of a transient signal in the two-inverter loop during settling.

$$A^n = e^{+\frac{T}{\tau_0}} \qquad (8.17)$$

This expression describes the time-domain behavior of the diverging process until it reaches stable points, as depicted in Fig. 8.6.

## 8.3. The SR Latch Circuit

The bistable element consisting of two cross-coupled inverters (Fig. 8.2) has two stable operating modes, or states. The circuit preserves its state (either one of the two possible modes) as long as the power supply voltage is provided; hence, the circuit can perform a simple memory function of *holding* its state. However, the simple two-inverter circuit examined above has no provision for allowing its state to be changed externally from one stable operating mode to the other. To allow such a change of state, we must add simple switches to the bistable element, which can be used to force or trigger the circuit from one operating point to the other. Figure 8.7 shows the circuit structure of the simple CMOS SR latch, which has two such triggering inputs, S (set) and R (reset). In the literature, the SR latch is also called an SR flip-flop, since two stable states can be switched back and forth. The circuit consists of two CMOS NOR2 gates. One of the input terminals of each NOR gate is used to cross-couple to the output of the other NOR gate, while the second input enables triggering of the circuit.

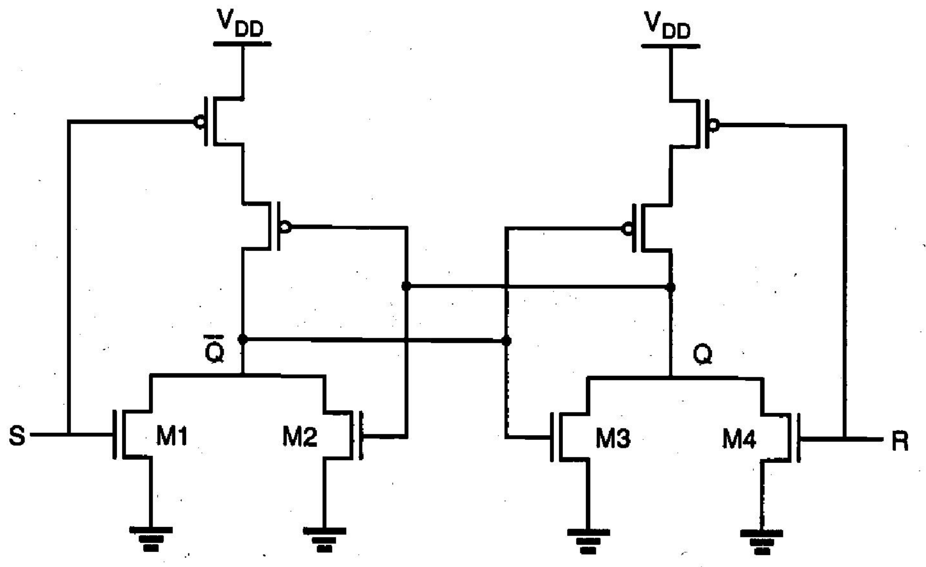


*Figure 8.7.* CMOS SR latch circuit based on NOR2 gates.

The SR latch circuit has two complementary outputs, Q and $\overline{Q}$. By definition, the latch is said to be in its *set* state when Q is equal to logic "1" and $\overline{Q}$ is equal to logic "0." Conversely, the latch is in its *reset* state when the output Q is equal to logic "0" and $\overline{Q}$ is equal to "1." The gate-level schematic of the SR latch consisting of two NOR2 gates, and the corresponding block diagram representation are shown in Fig. 8.8. It can easily be seen that when both input signals are equal to logic "0," the SR latch will operate exactly like the simple cross-coupled bistable element examined earlier, i.e., it will preserve (hold) either one of its two stable operating points (states) as determined by the previous inputs.

If the *set input* (S) is equal to logic "1" and the *reset input* is equal to logic "0," then the output node Q will be forced to logic "1" while the output node $\overline{Q}$ is forced to logic "0." This means that the SR latch will be *set*, regardless of its previous state.

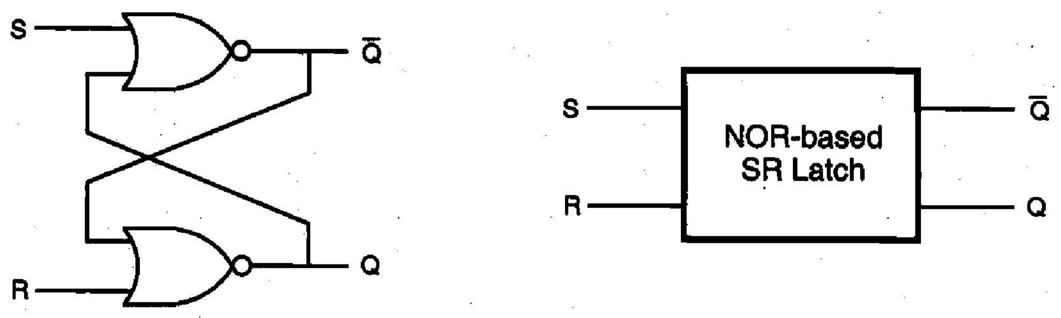


***Figure 8.8.*** Gate-level schematic and block diagram of the NOR-based SR latch.

Similarly, if S is equal to "0" and R is equal to "1," then the output node Q will be forced to "0" while $\overline{Q}$ is forced to "1." Thus, with this input combination, the latch is *reset*, regardless of its previously held state. Finally, consider the case in which both of the inputs S and R are equal to logic "1." In this case, both output nodes will be forced to logic "0," which conflicts with the complementarity of Q and $\overline{Q}$. Therefore, this input combination is not permitted during normal operation and is considered to be a *not-allowed* condition. The truth table of the NOR-based SR latch is summarized in the following:

| $S$ | $R$ | $Q_{n+1}$ | $\overline{Q}_{n+1}$ | Operation |
|---|---|---|---|---|
| 0 | 0 | $Q_n$ | $\overline{Q}_n$ | hold |
| 1 | 0 | 1 | 0 | set |
| 0 | 1 | 0 | 1 | reset |
| 1 | 1 | 0 | 0 | not allowed |

**Table 8.1.** Truth table of the NOR-based SR latch circuit

The operation of the CMOS SR latch circuit shown in Fig. 8.7 can be examined in more detail by considering the operating modes of the four nMOS transistors, M1, M2, M3, and M4. If the set input (S) is equal to $V_{OH}$ and the reset input (R) is equal to $V_{OL}$, both of the parallel-connected transistors M1 and M2 will be on. Consequently, the voltage on node $\overline{Q}$ will assume a logic-low level of $V_{OL} = 0$. At the same time, both M3 and M4 are turned off, which results in a logic-high voltage $V_{OH}$ at node Q. If the reset input (R) is equal to $V_{OH}$ and the set input (S) is equal to $V_{OL}$, the situation will be reversed (M1 and M2 turned off and M3 and M4 turned on).

When both of the input voltages are equal to $V_{OL}$, on the other hand, there are two possibilities. Depending on the previous state of the SR latch, either M2 *or* M3 will be on, while both of the trigger transistors M1 and M4 are off. This will generate a logic-low level of $V_{OL} = 0$ at one of the output nodes, while the complementary output node is at $V_{OH}$. The static operation modes and voltage levels of the NOR-based CMOS SR latch circuit are summarized in the following table. For simplicity, the operating modes of the complementary pMOS transistors are not explicitly listed here.

| $S$ | $R$ | $Q_{n+1}$ | $\overline{Q_{n+1}}$ | Operation |
|---|---|---|---|---|
| $V_{OH}$ | $V_{OL}$ | $V_{OH}$ | $V_{OL}$ | M1 and M2 on, M3 and M4 off |
| $V_{OL}$ | $V_{OH}$ | $V_{OL}$ | $V_{OH}$ | M1 and M2 off, M3 and M4 on |
| $V_{OL}$ | $V_{OL}$ | $V_{OH}$ | $V_{OL}$ | *M*1 and M4 off, M2 on, or |
| $V_{OL}$ | $V_{OL}$ | $V_{OL}$ | $V_{OH}$ | M1 and M4 off, M3 on |

**Table 8.2.** Operation modes of the transistors in the NOR-based CMOS SR latch circuit.

For the transient analysis of the SR latch circuit, we have to consider an event which results in a state change, i.e., either an initially reset latch being set by applying a *set* signal, or an initially set latch being reset by applying the *reset* signal. In either case, we note that both of the output nodes undergo simultaneous voltage transitions. While one output is rising from its logic-low level to logic-high, the other output node is falling from its initial logic-high level to logic-low. Thus, an interesting problem is to estimate the amount of time required for the simultaneous switching of the two output nodes. The exact solution of this problem obviously requires the simultaneous solution of two coupled differential equations, one each for each output node. The problem can, however, be simplified considerably if we assume that the two events described above take place in sequence rather than simultaneously. This assumption causes an overestimation of the switching time.

To calculate the switching times for both output nodes, we first have to find the total parasitic capacitance associated with each node. Simple inspection of the circuit shows that the total lumped capacitance at each output node can be approximated as follows:

$$\begin{aligned} C_Q &= C_{gb,2} + C_{gb,5} + C_{db,3} + C_{db,4} + C_{db,7} + C_{sb,7} + C_{db,8} \\ C_{\overline{Q}} &= C_{gb,3} + C_{gb,7} + C_{db,1} + C_{db,2} + C_{db,5} + C_{sb,5} + C_{db,6} \end{aligned} \tag{8.18}$$

The circuit diagram of the SR latch is shown in Fig. 8.9 together with the lumped load capacitances at the nodes Q and $\overline{\text{Q}}$. Assuming that the latch is initially reset and that

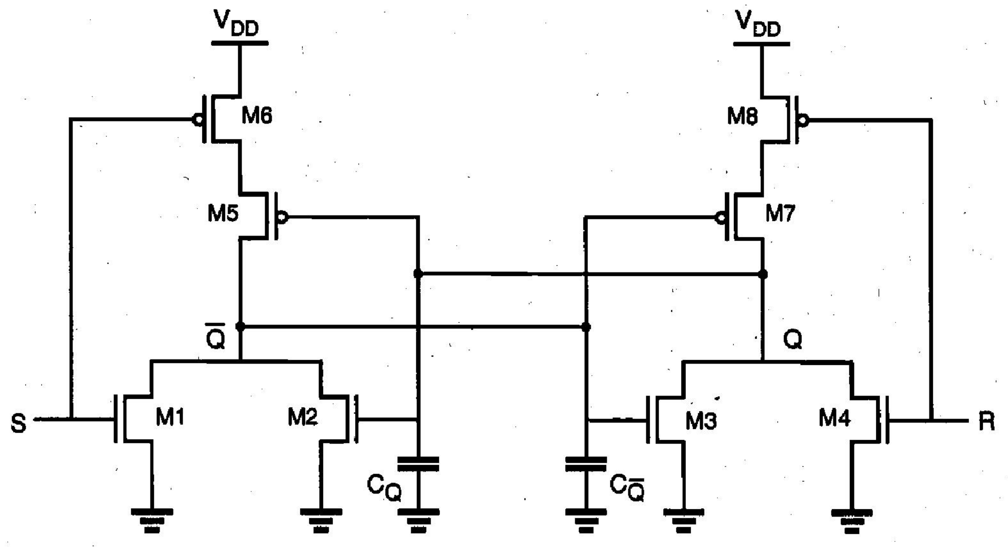


***Figure 8.9.*** Circuit diagram of the CMOS SR latch showing the lumped load capacitances at both output nodes.

a set operation is being performed by applying S = "1" and R = "0," the rise time associated with node Q can now be estimated as follows.

$$\tau_{rise,Q}\left(SR-latch\right)=\tau_{rise,Q}\left(\mathrm{NOR2}\right)+\tau_{fall,\overline{Q}}\left(\mathrm{NOR2}\right) \tag{8.19}$$

Note that the calculation of the switching time $\tau_{rise,Q}$ requires two separate calculations for the rise and fall times of the NOR2 gates. It is obvious that by considering the two events separately, i.e., first, one of the output node voltages ($\overline{Q}$) falling from high to low due to turn-on of M1, followed by the other node voltage (Q) rising from low to high due to turn-off of M3, we are bound to overestimate the actual switching time for the SR latch. Both M2 and M4 can be assumed to be off in this process, although M2 can be turned on as Q rises, thus actually shortening the $\overline{Q}$ node fall time. This approach, however, yields a simpler first-order prediction for the time delay, as opposed to the simultaneous solution of two coupled differential equations.

The NOR-based SR latch can also be implemented by using two cross-coupled depletion-load nMOS NOR2 gates, as shown in Fig. 8.10. From the logic point of view, the operation principle of the depletion-load nMOS NOR-based SR latch is identical to that of the CMOS SR latch. In terms of power dissipation and noise margins, however, the CMOS circuit implementation offers a better alternative, since both of the CMOS NOR2 gates dissipate virtually no static power for preserving a state, and since the output voltages can exhibit a full swing between 0 and $V_{DD}$.

Now consider a different approach for building the basic SR latch circuit. Instead of using two NOR2 gates, we can use two NAND2 gates, as shown in Fig. 8.11. Here, one input of each NAND gate is used to cross-couple to the output of the other NAND gate, while the second input enables external triggering.

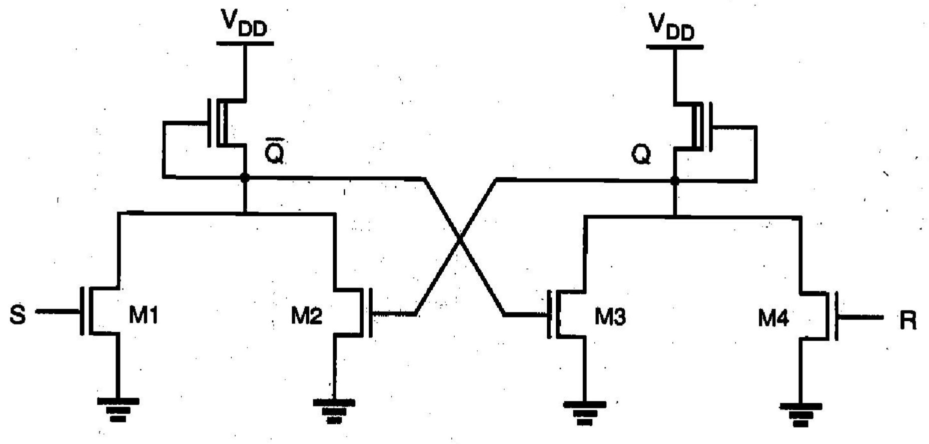


*Figure 8.10.* Depletion-load nMOS SR latch circuit based on NOR2 gates.

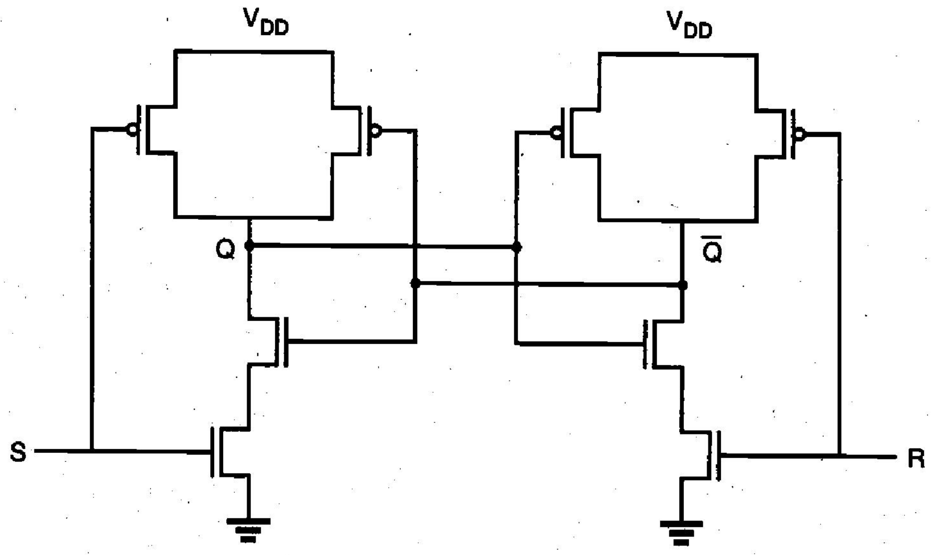
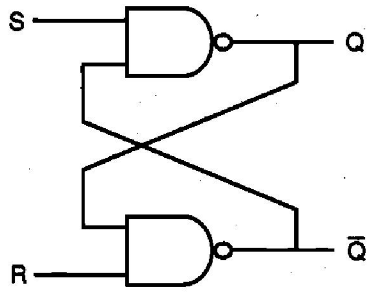


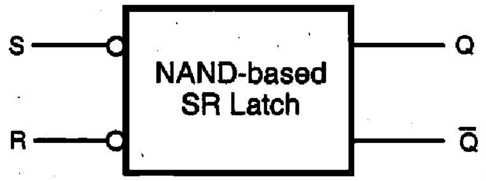
*Figure 8.11.* CMOS SR latch circuit based on NAND2 gates.

A close inspection of the NAND-based SR latch circuit reveals that in order to hold (preserve) a state, both of the external trigger inputs must be equal to logic "1." The operating point or the state of the circuit can be changed only by pulling the *set* input to logic zero or by pulling the *reset* input to zero. We can observe that if S is equal to "0" and R is equal to "1," the output Q attains a logic "1" value and the complementary output $\overline{Q}$ becomes logic "0." Thus, in order to *set* the NAND SR latch, a logic "0" must be applied to the *set* (S) input. Similarly, in order to *reset* the latch, a logic "0" must be applied to the *reset* (R) input. The conclusion is that the NAND-based SR latch responds to *active low* input signals, as opposed to the NOR-based SR latch, which responds to *active high* inputs. Note that if both input signals are equal to logic "0," both output nodes assume a logic-high level, which is not allowed because it violates the complementarity of the two outputs.

The gate-level schematic and the corresponding block diagram representation of the NAND-based SR latch circuit are shown in Fig. 8.12. The small circles at the S and R input terminals indicate that the circuit responds to *active low* input signals. The truth table of the NAND SR latch is also shown in the following. The same approach used in the timing analysis of NOR-based SR latches can be applied to NAND-based SR latches.

The NAND-based SR latch can also be implemented by using two cross-coupled depletion-load NAND2 gates, as shown in Fig. 8.13. While the operation principle is identical to that of the CMOS NAND SR latch (Fig. 8.11) from the logic point of view, the CMOS circuit implementation again offers a better alternative in terms of static power dissipation and noise margins.





| $S$ | $R$ | $Q_{n+1}$ | $\overline{Q_{n+1}}$ | Operation |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | not allowed |
| 0 | 1 | 1 | 0 | set |
| 1 | 0 | 0 | 1 | reset |
| 1 | 1 | $Q_n$ | $\overline{Q_n}$ | hold |

***Figure 8.12.*** Gate-level schematic and block diagram of the NAND-based SR latch.

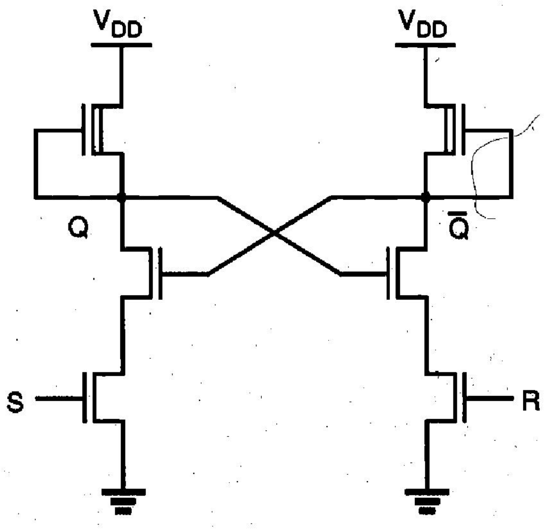


***Figure 8.13.*** Depletion-load nMOS NAND-based SR latch circuit.

### *Clocked SR Latch*

All of the SR latch circuits examined in the previous section are essentially asynchronous sequential circuits, which will respond to the changes occurring in input signals at a circuit-delay-dependent time point during their operation. To facilitate synchronous operation, the circuit response can be controlled simply by adding a gating clock signal to the circuit, so that the outputs will respond to the input levels only during the active period of a clock pulse. For simple reference, the clock pulse will be assumed to be a periodic square waveform, which is applied simultaneously to all clocked logic gates in the system.

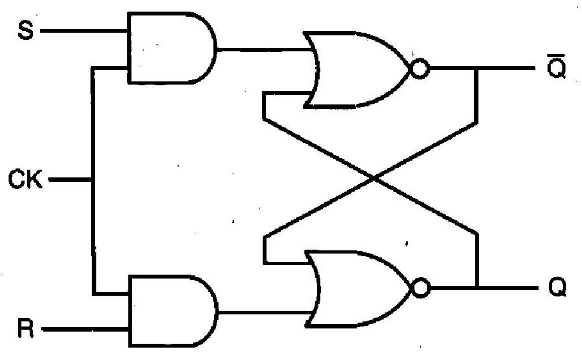


***Figure 8.14.*** Gate-level schematic of the clocked NOR-based SR latch.

The gate-level schematic of a clocked NOR-based SR latch is shown in Fig. 8.14. It can be seen that if the clock (CK) is equal to logic "0," the input signals have no influence upon the circuit response. The outputs of the two AND gates will remain at logic "0," which forces the SR latch to hold its current state regardless of the S and R input signals. When the clock input goes to logic "1," the logic levels applied to the S and R inputs are permitted to reach the SR latch, and possibly change its state. Note that as in the non-clocked SR latch, the input combination S = R = "1" is not allowed in the clocked SR latch. With both inputs S and R at logic "1," the occurrence of a clock pulse causes both outputs to go momentarily to zero. When the clock pulse is removed, i.e., when it becomes "0," the state of the latch is indeterminate. It can eventually settle into either state, depending on slight delay differences between the output signals.

To illustrate the operation of the clocked SR latch, a sample sequence of CK, S, and R waveforms, and the corresponding output waveform Q are shown in Fig. 8.15. Note that the circuit is strictly *level-sensitive* during active clock phases, i.e., any changes occurring in the S and R input voltages when the CK level is equal to "1" will be reflected onto the circuit outputs. Consequently, even a narrow spike or glitch occurring during an active clock phase can set or reset the latch, if the loop delay is shorter than the pulse width.

Figure 8.16 shows a CMOS implementation of the clocked NOR-based SR latch circuit, using two simple AOI gates. Notice that the AOI-based implementation of the circuit results in a very small transistor count, compared with the alternative circuit realization consisting of two AND2 and two NOR2 gates.

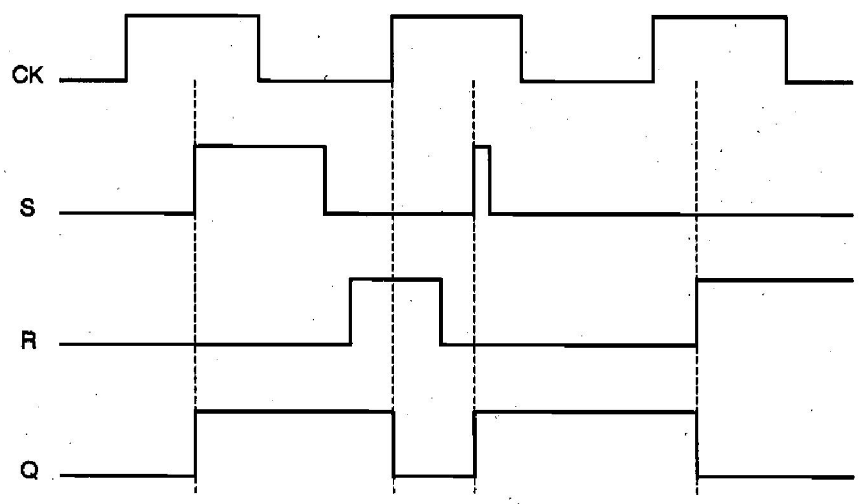


***Figure 8.15.*** Sample input and output waveforms illustrating the operation of the clocked NOR-based SR latch circuit.

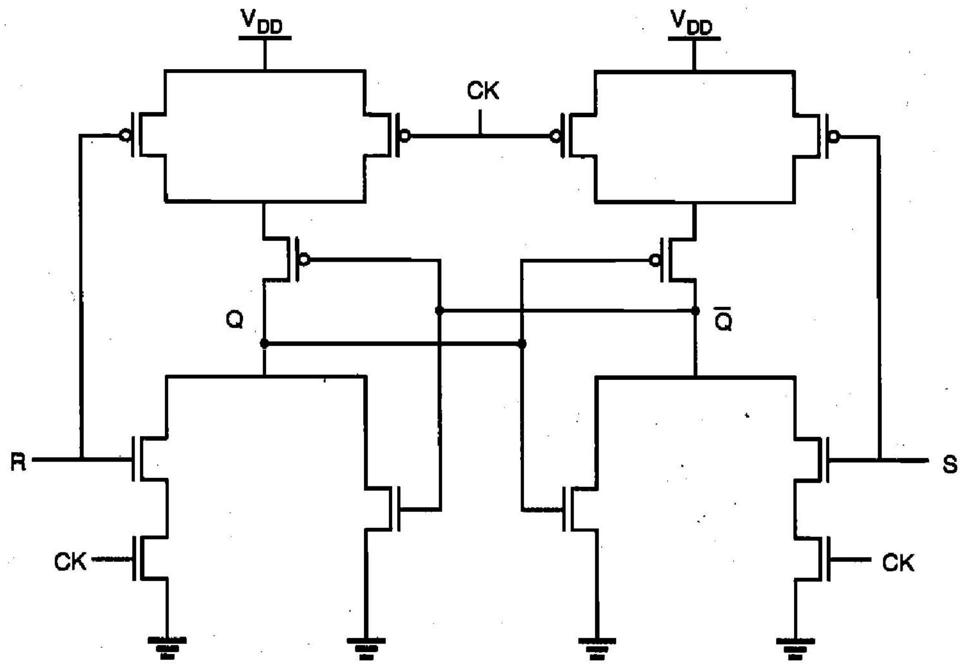


***Figure 8.16.*** AOI-based implementation of the clocked NOR-based SR latch circuit.

The NAND-based SR latch can also be implemented with gating clock input, as shown in Fig. 8.17. It must be noted, however, that both input signals S and R as well as the clock signal CK are *active low* in this case. This means that changes in the input signal levels will be ignored when the clock is equal to logic "1," and that inputs will influence the outputs only when the clock is active, i.e., CK = "0." For the circuit implementation of this clocked NAND-based SR latch, we can use a simple OAI structure, which is essentially analogous to the AOI-based realization of the clocked NOR SR latch circuit.

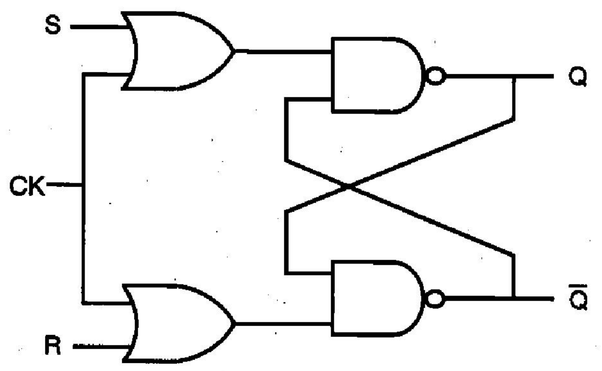


***Figure 8.17.*** Gate-level schematic of the clocked NAND-based SR latch circuit, with active low inputs.

A different implementation of the clocked NAND-based SR latch is shown in Fig. 8.18. Here, both input signals and the CK signal are *active high*, i.e., the latch output Q will be set when CK = "1," S = "1," and R = "0." Similarly, the latch will be reset when CK = "1," S = "0," and R = "1." The latch preserves its state as long as the clock signal is inactive, i.e., when CK = "0." The drawback of this implementation is that the transistor count is higher than the *active low* version shown in Fig. 8.17.

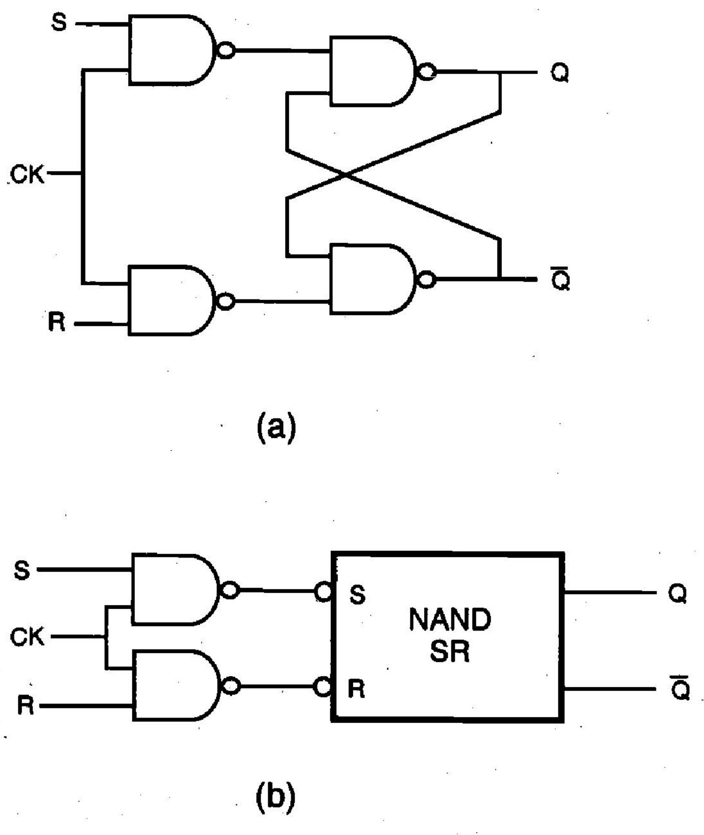


***Figure 8.18.*** (a) Gate-level schematic of the clocked NAND-based SR latch circuit, with active high inputs. (b) Partial block diagram representation of the same circuit.

## *Clocked JK Latch*

All simple and clocked SR latch circuits examined to this point suffer from the common problem of having a not-allowed input combination, i.e., their state becomes indeterminate when both inputs S and R are activated at the same time. This problem can be overcome by adding two feedback lines from the outputs to the inputs, as shown in Fig. 8.19. The resulting circuit is called a JK latch. Figure 8.19 shows an all-NAND implementation of the JK latch with active high inputs, and the corresponding block diagram representation. The JK latch is commonly called a JK flip-flop.

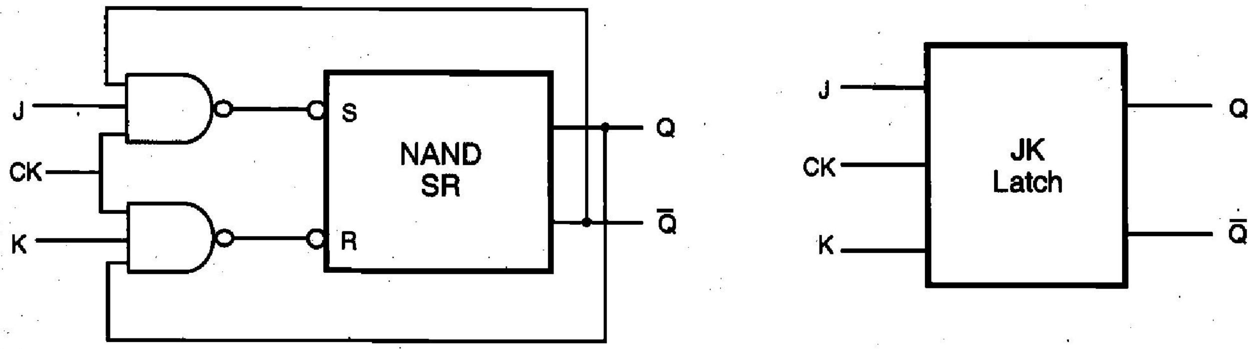


***Figure 8.19.*** Gate-level schematic of the clocked NAND-based JK latch circuit.

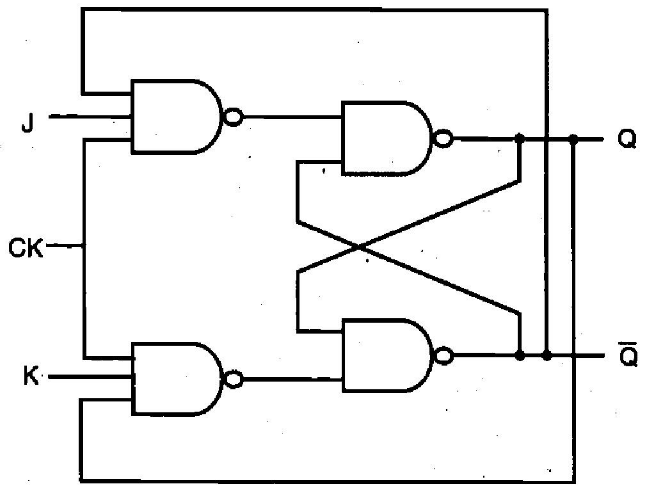


***Figure 8.20.*** All-NAND implementation of the clocked JK latch circuit.

The J and K inputs in this circuit correspond to the set and reset inputs of the basic SR latch. When the clock is active, the latch can be set with the input combination (J = "1," K = "0"), and it can be reset with the input combination (J = "0," K = "1"). If both inputs are equal to logic "0," the latch preserves its current state. If, on the other hand, both inputs are equal to "1" during the active clock phase, the latch simply switches its state due to feedback. In other words, the JK latch does not have a not-allowed input combination. As in the other clocked latch circuits, the JK latch will hold its current state when the clock is inactive (CK = "0"). The operation of the clocked JK latch is summarized in the truth table (Table 8.3).

Figure 8.21 shows an alternative, NOR-based implementation of the clocked JK latch, and CMOS realization of this circuit. Note that the AOI-based circuit structure results in a relatively low transistor count, and consequently, a more compact circuit compared to the all-NAND realization shown in Fig. 8.20.

| $J$ | $K$ | $Q_n$ | $\overline{Q_n}$ | $S$ | $R$ | $Q_{n+1}$ | $\overline{Q_{n+1}}$ | *Operation* |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | *hold* |
| | | 1 | 0 | 1 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | *reset* |
| | | 1 | 0 | 1 | 0 | 0 | 1 | |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | *set* |
| | | 1 | 0 | 1 | 1 | 1 | 0 | |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | *toggle* |
| | | 1 | 0 | 1 | 0 | 0 | 1 | |

**Table 8.3.** Detailed truth table of the JK latch circuit.

While there is no not-allowed input combination for the JK latch, there is still a potential problem. If both inputs are equal to logic "1" during the active phase of the clock pulse, the output of the circuit will oscillate (toggle) continuously until either the clock becomes inactive (goes to zero), or one of the input signals goes to zero. To prevent this undesirable timing problem, the clock pulse width must be made smaller than the

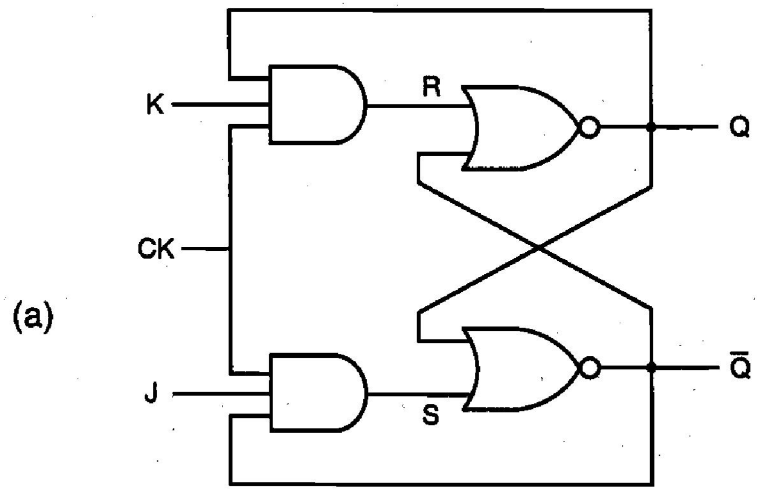


*Figure 8.21.* (a) Gate-level schematic of the clocked NOR-based JK latch circuit.

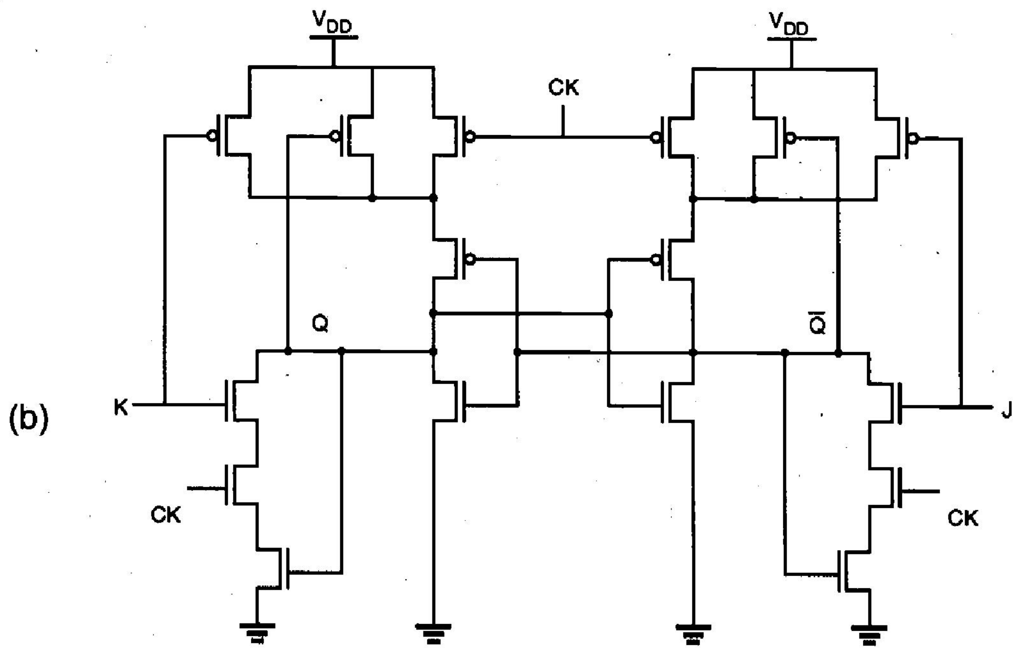


*Figure 8.21. (continued)* (b) CMOS AOI realization of the JK latch.

input-to-output propagation delay of the JK latch circuit. This restriction dictates that the clock signal must go low before the output level has an opportunity to switch again, which prevents uncontrolled oscillation of the output. However, note that this clock constraint is difficult to implement for most practical applications.

Assuming that the clock timing constraint described above is satisfied, the output of the JK latch will toggle (change its state) only once for each clock pulse, if both inputs are equal to logic "1" (Fig. 8.22). A circuit which is operated exclusively in this mode is called a *toggle switch*.

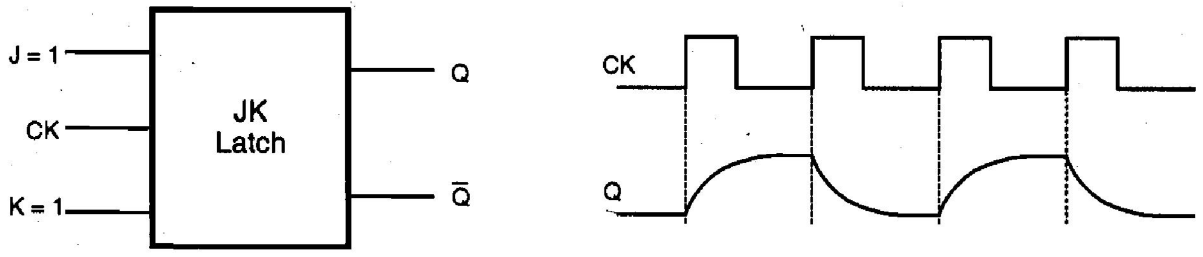


*Figure 8.22.* Operation of the JK latch as a toggle switch.

### *Master-Slave Flip-Flop*

Most of the timing limitations encountered in the previously examined clocked latch circuits can be prevented by using two latch stages in a cascaded configuration. The key operation principle is that the two cascaded stages are activated with opposite clock phases. This configuration is called the *master-slave flip-flop*. Our definition of flip-flop is designed to distinguish it from latches discussed previously, although they are mostly used interchangeably in the literature.

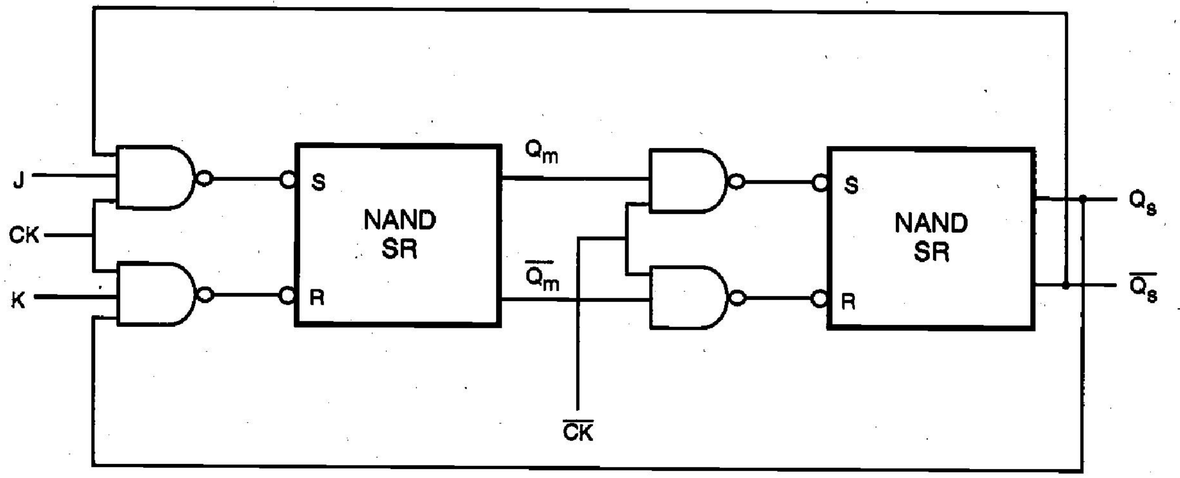


*Figure 8.23.* Master-slave flip-flop consisting of NAND-based JK latches.

The input latch in Fig. 8.23, called the "master," is activated when the clock pulse is high. During this phase, the inputs J and K allow data to be entered into the flip-flop, and the first-stage outputs are set according to the primary inputs. When the clock pulse goes to zero, the master latch becomes inactive and the second-stage latch, called the "slave," becomes active. The output levels of the flip-flop circuit are determined during this second phase, based on the master-stage outputs set in the previous phase.

Since the master and the slave stages are effectively decoupled from each other with the opposite clocking scheme, the circuit is never *transparent*, i.e., a change occurring in the primary inputs is never reflected directly to the outputs. This very important property clearly separates the master-slave flip-flop from all of the latch circuits examined earlier in this section. Figure 8.24 shows a sample set of input and output waveforms associated with the JK master-slave flip-flop, which can help the reader to study the basic operation principles.

Because the master and the slave stages are decoupled from each other, the circuit allows for toggling when J = K = "1," but it eliminates the possibility of uncontrolled oscillations since only one stage is active at any given time. A NOR-based alternative realization for the master-slave flip-flop circuit is shown in Fig. 8.25.

Figure 8.24 also shows that the master-slave flip-flop circuit examined here has the potential problem of "one's catching." When the clock pulse is high, a narrow spike or glitch in one of the inputs, for instance a glitch in the J line (or K line), may set (or reset) the master latch and thus cause an unwanted state transition, which will then be propagated into the slave stage during the following phase. This problem can be eliminated to a large extent by building an edge-triggered master-slave flip-flop, which will be examined in the following section.

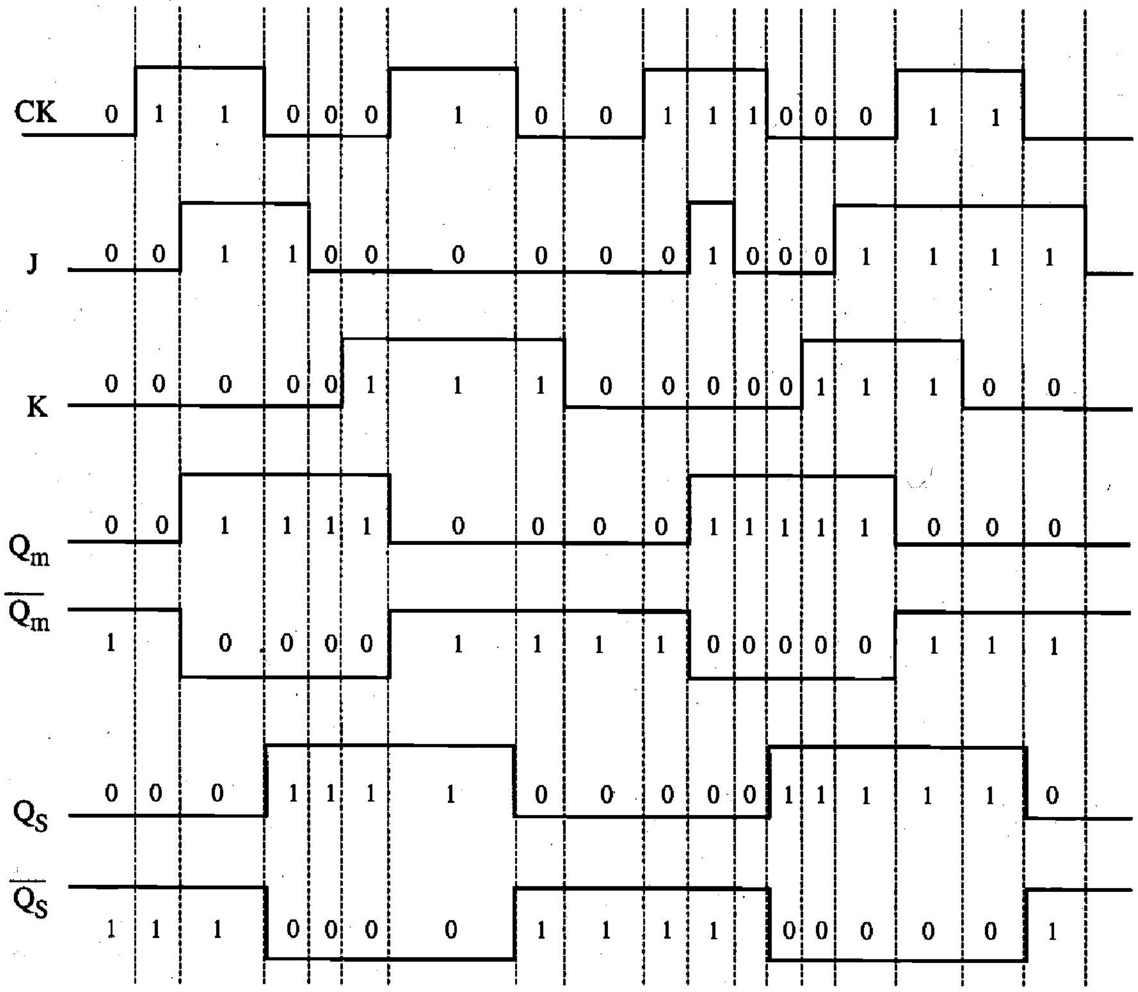


*Figure 8.24.* Sample input and output waveforms of the master-slave flip-flop circuit.

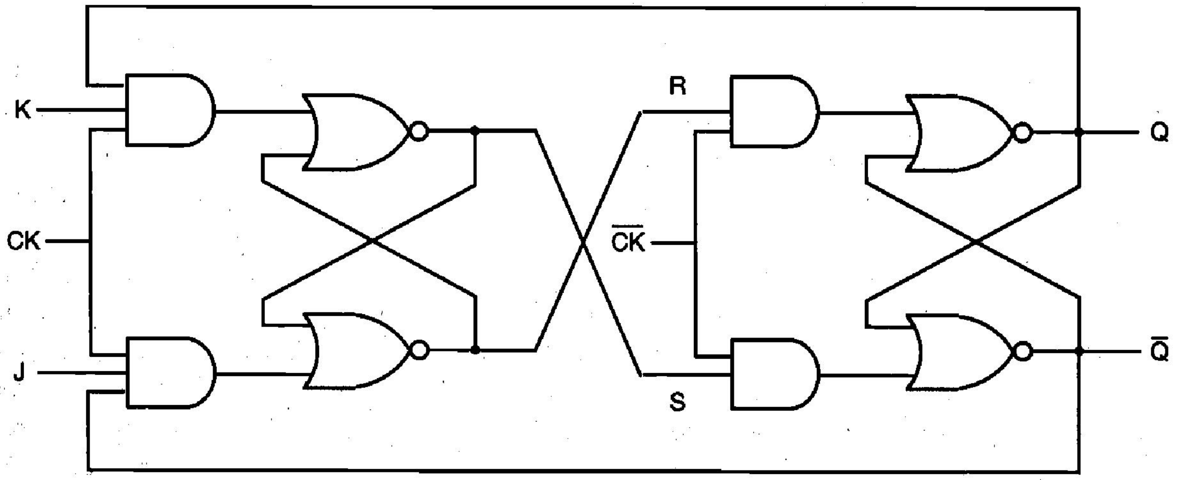


*Figure 8.25.* NOR-based realization of the JK master-slave flip-flop.

## 8.5. CMOS D-Latch and Edge-Triggered Flip-Flop

With the widespread use of CMOS circuit techniques in digital integrated circuit design, a large selection of CMOS-based sequential circuits have also gained popularity and prominence, especially in VLSI design. Throughout this chapter, we have seen examples showing that virtually all of the latch and flip-flop circuits can be implemented with CMOS gates, and that their design is quite straightforward. However, direct CMOS implementations of conventional circuits such as the clocked JK latch or the JK master-slave flip-flop tend to require a large number of transistors.

In this section, we will see that specific versions of sequential circuits built primarily with CMOS transmission gates are generally simpler and require fewer transistors than the circuits designed with conventional structuring. As an introduction to the issue, let us first consider the simple D-latch circuit shown in Fig. 8.26. The gate-level representation of the D-latch is simply obtained by modifying the clocked NOR-based SR latch circuit. Here, the circuit has a single input D, which is directly connected to the S input of the latch. The input variable D is also inverted and connected to the R input of the latch. It can be seen from the gate-level schematic that the output Q assumes the value of the input D when the clock is active, i.e., for CK = "1." When the clock signal goes to zero, the output will simply preserve its state. Thus, the CK input acts as an enable signal which allows data to be accepted into the D-latch.

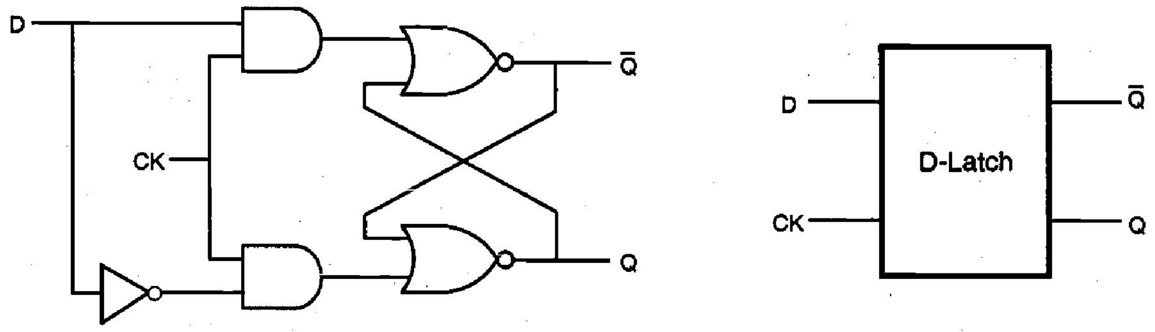


*Figure 8.26.* Gate-level schematic and the block diagram view of the D-latch.

The D-latch finds many applications in digital circuit design, primarily for temporary storage of data or as a delay element. In the following, we will examine its simple CMOS implementation. Consider the circuit diagram given in Fig. 8.27, which shows a basic two-inverter loop and two CMOS transmission gate (TG) switches.

The TG at the input is activated by the CK signal, whereas the TG in the inverter loop is activated by the inverse of the CK signal, $\overline{\text{CK}}$. Thus, the input signal is accepted (latched) into the circuit when the clock is high, and this information is preserved as the state of the inverter loop when the clock is low. The operation of the CMOS D-latch circuit can be better visualized by replacing the CMOS transmission gates with simple switches, as shown in Fig. 8.28. A timing diagram accompanying this figure shows the time intervals during which the input and the output signals should be valid (unshaded).

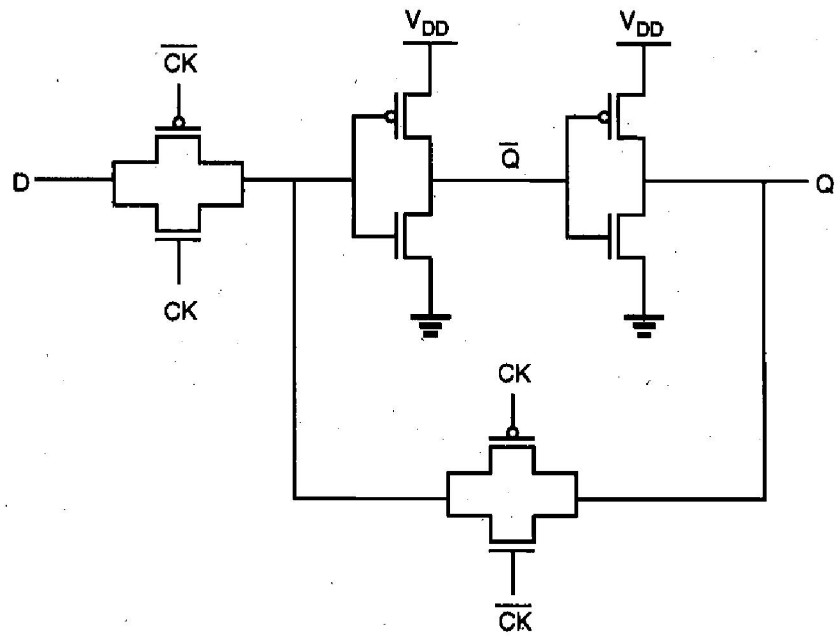


***Figure 8.27.*** CMOS implementation of the D-latch (version 1).

Note that the valid D input must be stable for a short time before (*setup time,* $t_{setup}$) and after (*hold time,* $t_{hold}$) the negative clock transition, during which the input switch opens and the loop switch closes. Once the inverter loop is completed by closing the loop switch, the output will preserve its valid level. In the D-latch design, the requirements for setup time and hold time should be met carefully. Any violation of such specifications can cause *metastability* problems which lead to seemingly chaotic transient behavior, and can result in an unpredictable state after the transitional period.

The D-latch shown in Fig. 8.27 is not an edge-triggered storage element because the output changes according to the input, i.e., the latch is transparent, while the clock is high. The transparency property makes the application of this D-latch unsuitable for counters and some data storage implementations.

Figure 8.29 shows a different version of the CMOS D-latch. The circuit contains two tristate inverters, driven by the clock signal and its inverse. Although the circuit appears to be quite different from that shown in Fig. 8.27, the basic operation principle of the circuit is the same as that shown in Fig. 8.28. The first tri-state inverter acts as the input switch, accepting the input signal when the clock is high. At this time, the second tristate inverter is at its high-impedance state, and the output Q is following the input signal. When the clock goes low, the input buffer becomes inactive, and the second tristate inverter completes the two-inverter loop, which preserves its state until the next clock pulse.

Finally, consider the two-stage master-slave flip-flop circuit shown in Fig. 8.30, which is constructed by simply cascading two D-latch circuits. The first stage (master) is driven by the clock signal, while the second stage (slave) is driven by the inverted clock signal. Thus, the master stage is positive level-sensitive, while the slave stage is negative level-sensitive.

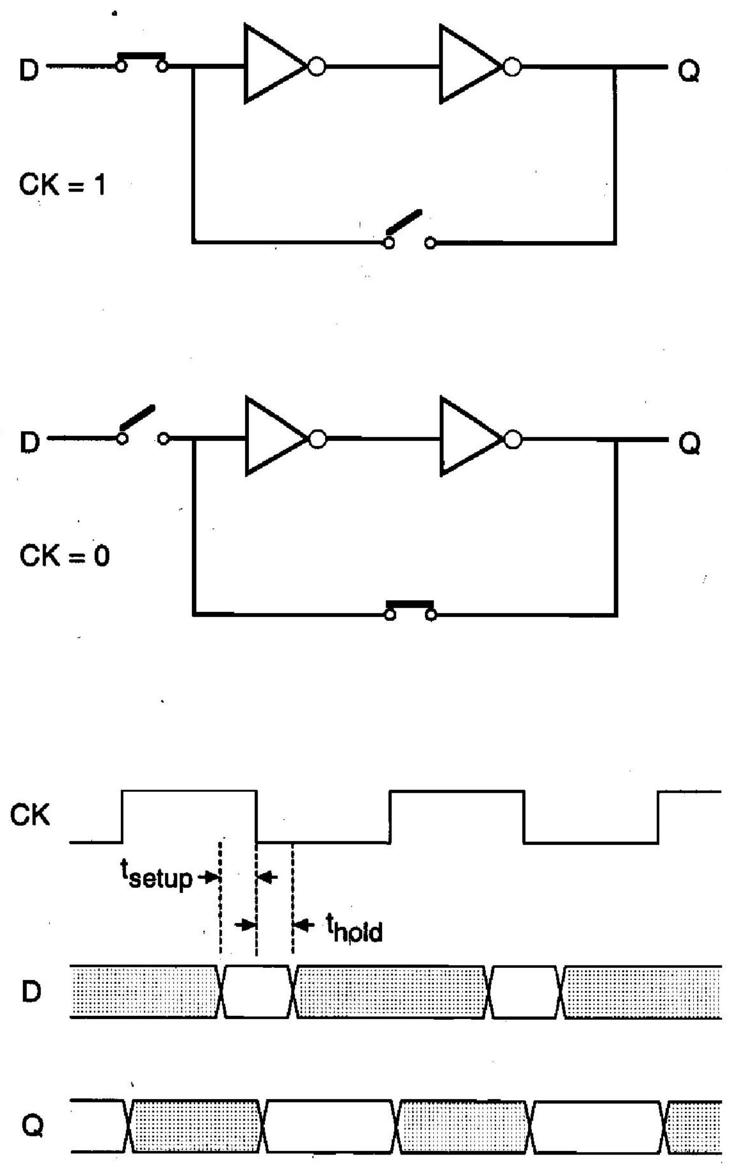


*Figure 8.28.* Simplified schematic view and the corresponding timing diagram of the CMOS D-latch circuit, showing the setup time and the hold time.

When the clock is high, the master stage follows the D input while the slave stage holds the previous value. When the clock changes from logic "1" to logic "0," the master latch ceases to sample the input and stores the D value at the time of the clock transition. At the same time, the slave latch becomes transparent, passing the stored master value $Q_m$ to the output of the slave stage, $Q_s$. The input cannot affect the output because the master stage is disconnected from the D input. When the clock changes again from logic "0" to "1," the slave latch locks in the master latch output and the master stage starts sampling the input again. Thus, this circuit is a negative edge-triggered D flip-flop by virtue of the fact that it samples the input at the falling edge of the clock pulse.

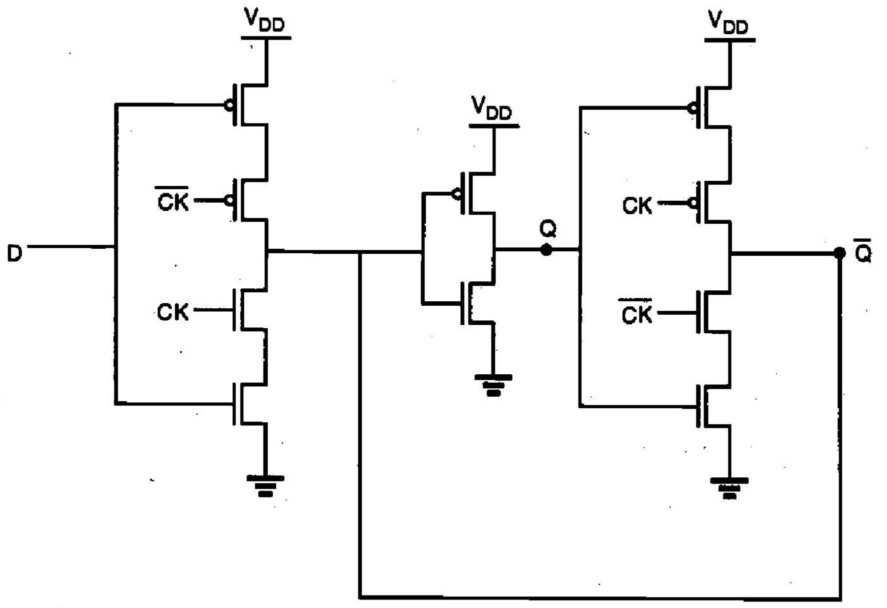


***Figure 8.29.*** CMOS implementation of the D-latch (version 2).

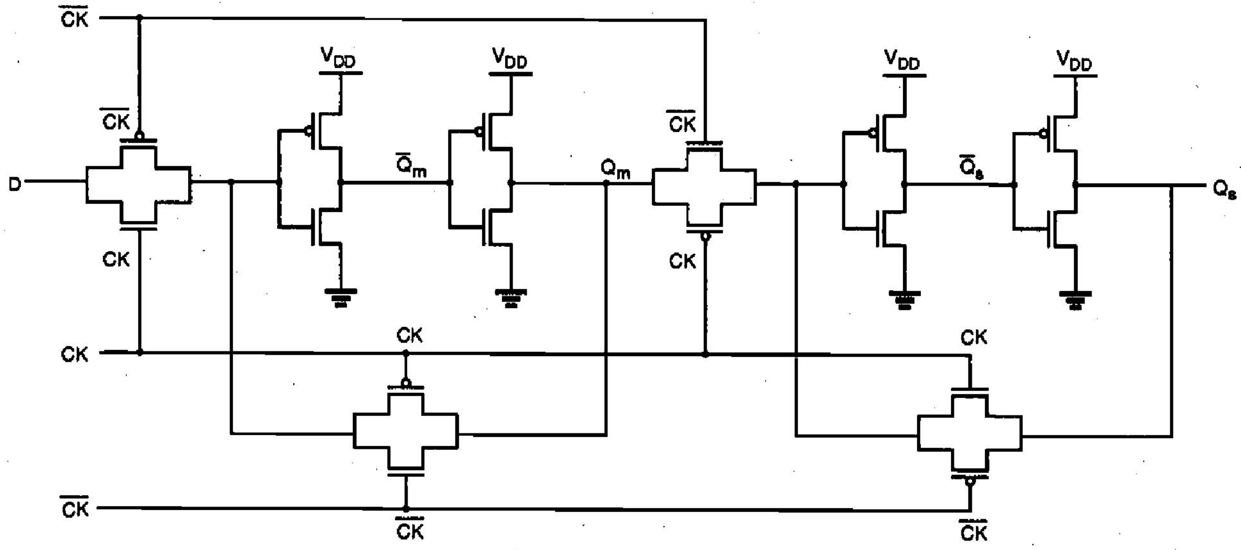


***Figure 8.30.*** CMOS negative (falling) edge-triggered master-slave D flip-flop (DFF).

Figure 8.31 shows the simulated input and output waveforms of the CMOS negative edge-triggered D-type flip-flop. The output of the master stage latches the applied input (D) when the clock signal is "1", and the output of the slave stage becomes valid when the clock signal drops to "0". Thus, the D-type flip-flop (DFF) essentially samples the input at every falling edge of the clock pulse.

It should be emphasized that the operation of the DFF circuit can be seriously affected if the master stage experiences a set-up time violation. This situation is illustrated in Fig. 8.32, where the input D switches from "0" to "1" immediately before

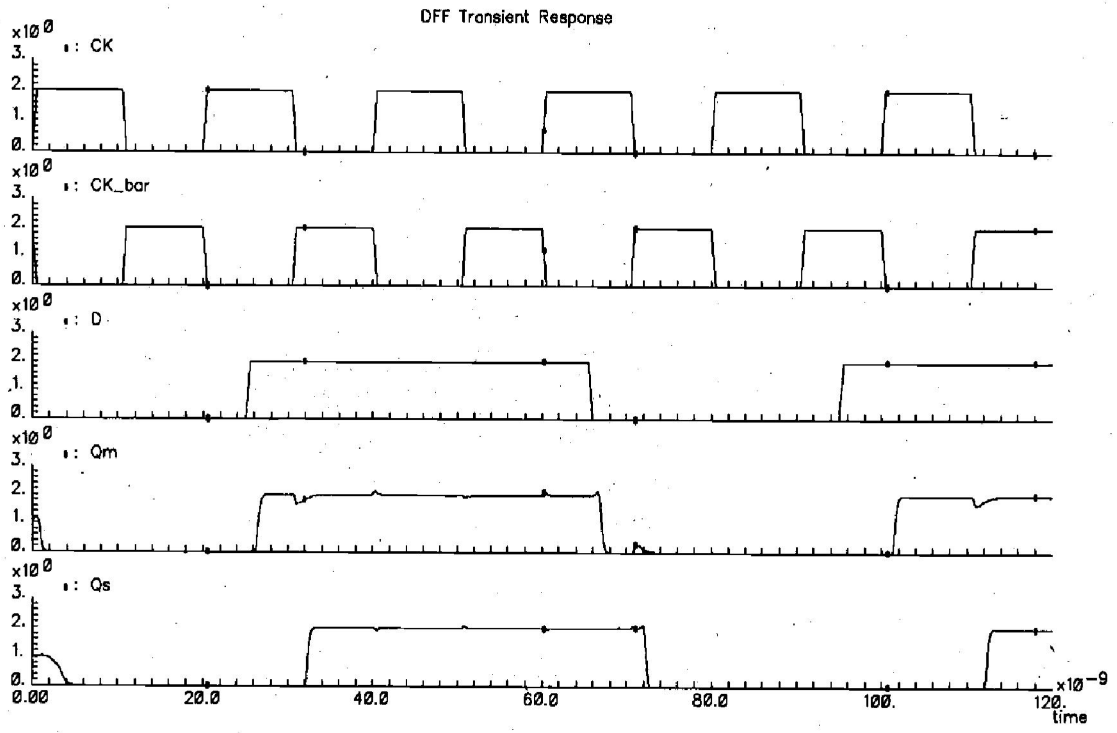


***Figure 8.31.*** Simulated input and output waveforms of the CMOS DFF circuit in Fig. 8.30.

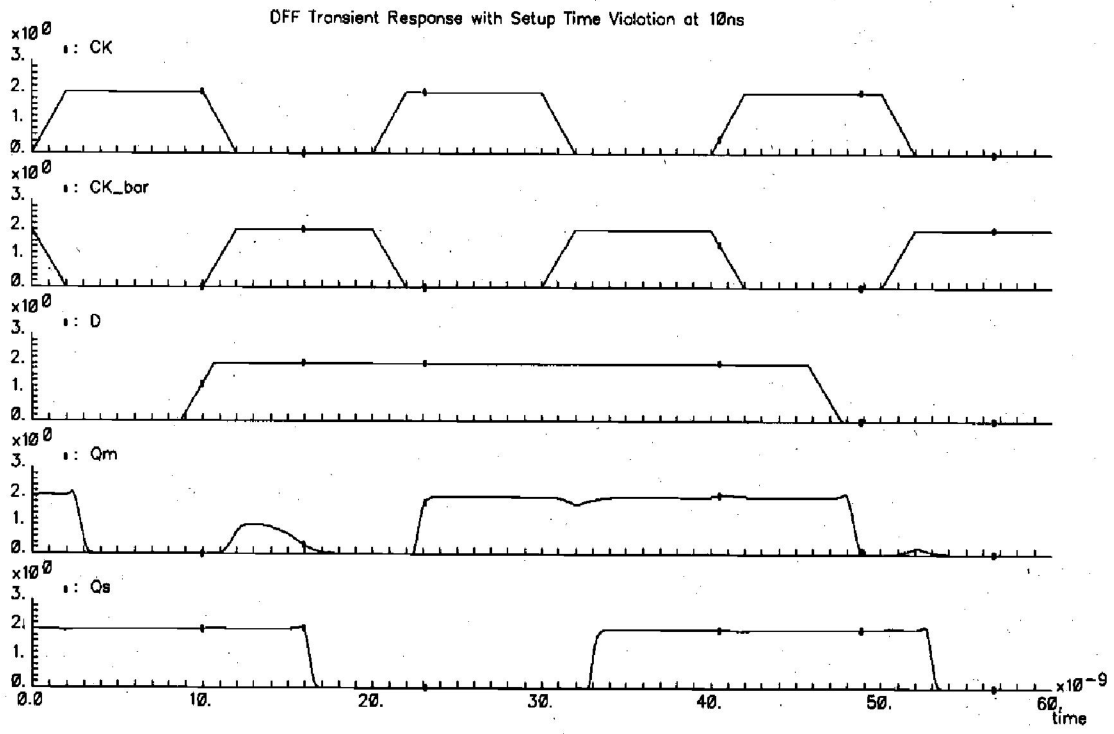


***Figure 8.32.*** Simulated waveforms of the CMOS DFF circuit, showing a set-up time violation for the master stage input at 10 ns. The output of the master stage fails to settle at the correct level.

the clock transition occurs (set-up time violation). As a result, the master stage fails to latch the correct value, and the slave stage produces an erroneous output. The relative timing of the input and clock signals are carefully synchronized to avoid such situations. The layout of the CMOS DFF circuit is given in Fig. 8.33.

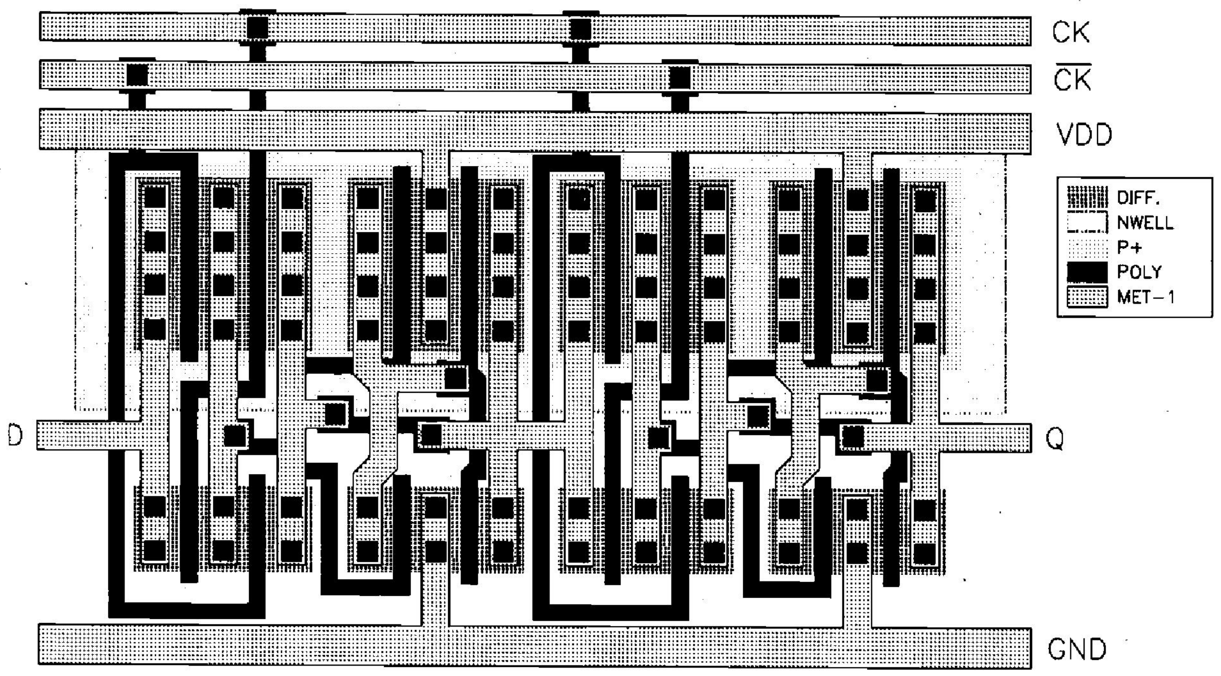


***Figure 8.33.*** Layout of the CMOS DFF shown in Fig. 8.30.

Another implementation of edge-triggered D flip-flop is shown in Fig. 8.34, which consists of six NAND3 gates. This D flip-flop is positive edge-triggered as illustrated in the waveform chart in Fig. 8.35. Initially, all the signal values except for S are 0, i.e., (S, R, CK, D) = (1, 0, 0, 0), and Q = 0. In the second phase, both D and R switch to 1, i.e., (S, R, CK, D) = (1, 0, 1, 1), but no change in Q occurs and the Q value remains at 0. However, in the third phase, if CK goes to high, i.e., (S, R, CK, D) = (1, 1, 1, 1), the output of gate 2 switches to 0, which in turn sets the output of the last stage SR latch to 1. Thus, the output of this D flip-flop switches to 1 at the positive-going edge of the clock signal, CK. However, as can be observed in the ninth phase of the waveform diagram chart, the Q output is not affected by the negative-going edge of CK, nor by other signal changes.

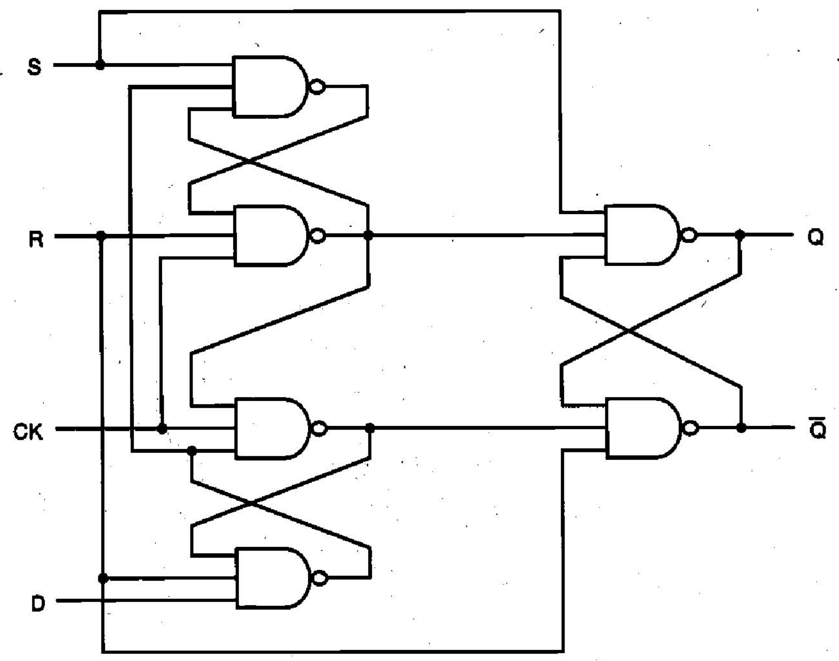


*Figure 8.34.* NAND3-based positive edge-triggered D flip-flop circuit.

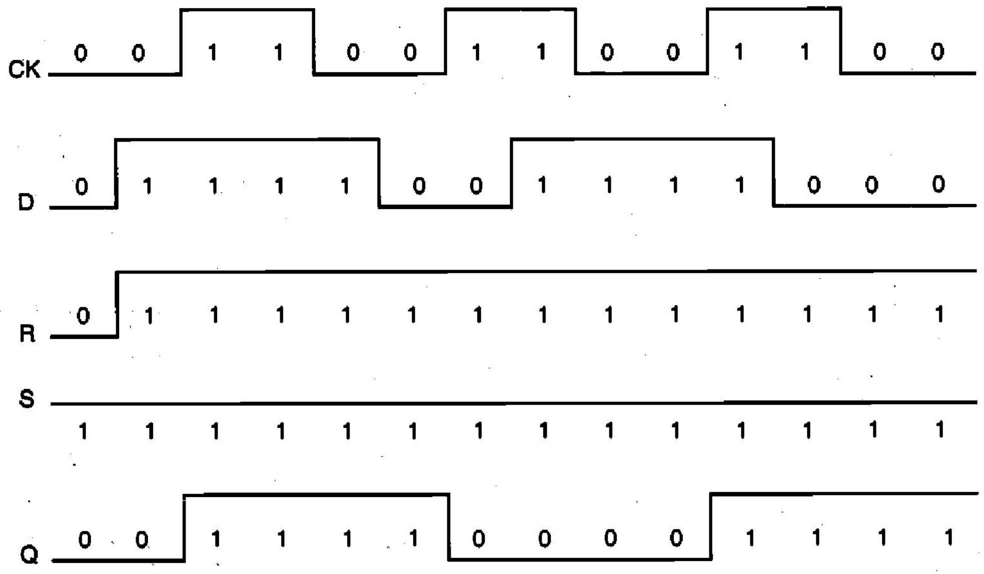


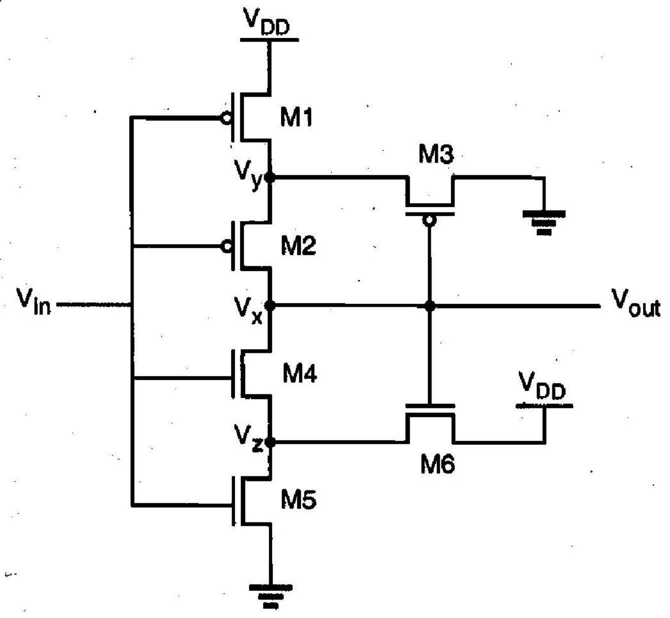
*Figure 8.35.* Timing diagram of the positive edge-triggered D flip-flop.

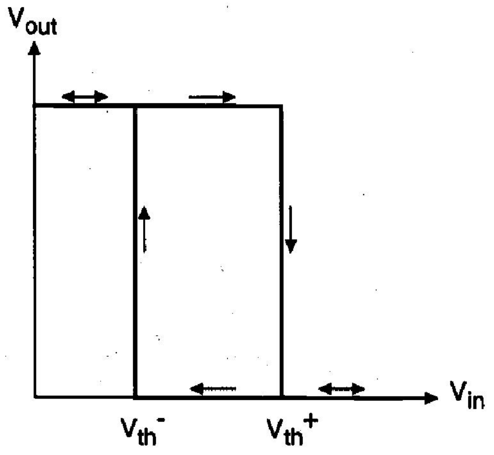
## Appendix

### Schmitt Trigger Circuit

In the following, we will examine the Schmitt trigger circuit, which is a very useful regenerative circuit. The Schmitt trigger has an inverter-like voltage transfer characteristic, but with two different logic threshold voltages for increasing and for decreasing input signals. With this unique property, the circuit can be utilized for the detection of low-to-high and high-to-low switching events in noisy environments.

The circuit diagram of a CMOS Schmitt trigger and the typical features of its voltage transfer characteristic (VTC) are shown below. Also listed here is the corresponding SPICE circuit input file. In the following, we will first calculate the important points in the VTC, and then compare our results with SPICE simulation.





```
CMOS Schmitt Trigger DC analysis
vdd 5 0 dc 5V
vin 1 0 dc 1v
m5 2 1 0 0 mn l=1u w=1u
m4 3 1 2 0 mn l=1u w=2.5u
m6 5 3 2 0 mn l=1u w=3u
m1 4 1 5 5 mp l=1u w=1u
m2 3 1 4 5 mp l=1u w=2.5u
m3 0 3 4 5 mp l=1u w=3u
.model mn nmos vto=1 gamma=0.4 kp=2.5e-5
.model mp pmos vto=-1 gamma=0.4 kp=1.0e-5
.dc vin 0 5 0.1
.print dc v(3)
.end
```

We start our step-by-step analysis by considering a positive input sweep, i.e., assuming that the input voltage is increasing from 0 to $V_{DD}$.

i) At $V_{in} = 0$ V:
M1 and M2 are turned on, then

$$V_x = V_y = V_{DD} = 5 \text{ V}$$

At the same time, M4 and M5 are turned off. M3 is off; M6 is on and operates in the saturation region. Calculating the threshold voltage of M6 with $2\phi_F = -0.6$ V,

$$V_z = V_{DD} - V_{T,6} = 3.5 \text{ V}$$

ii) At $V_{in} = V_{T0,n} = 1.0$ V:
M5 starts to turn on, M4 is still off.

$$V_x = 5 \text{ V}$$

iii) At $V_{in} = 2.0$ V:
Assume M4 is off, while both M5 and M6 operate in the saturation region.

$$\frac{1}{2}k'\left(\frac{W}{L}\right)_5 \left(V_{in} - V_{T0,n}\right)^2 = \frac{1}{2}k'\left(\frac{W}{L}\right)_6 \left(V_{DD} - V_z - V_{T,6}\right)^2$$

$$(2-1)^2 = 3\left(5 - V_z - \left[1 + 0.4\left(\sqrt{0.6 + V_z} - \sqrt{0.6}\right)\right]\right)^2$$

Solving this equation for $V_z$, we find that there is only one physically reasonable root.

$$V_z = 2.976 \text{ V}$$

Now, we check our assumption made above, i.e., M4 is indeed turned off:

$$V_{GS,4} = 2 - 2.976 = -0.976 < V_{T0,n} = 1$$

iv) At $V_{in} = 3.5$ V:

$V_z$ continues to decrease. Assuming M5 in linear region and M6 in saturation, we arrive at the following current equation.

$$\frac{1}{2}k'\left(\frac{W}{L}\right)_5 \left[2\left(V_{in} - V_{T0,n}\right)V_z - V_z^2\right] = \frac{1}{2}k'\left(\frac{W}{L}\right)_6 \left(V_{DD} - V_z - V_{T,6}\right)^2$$

$$\left[2(3.5-1.0)V_z - V_z^2\right] = 3\left(5 - V_z - \left[1 + 0.4\left(\sqrt{0.6+V_z} - \sqrt{0.6}\right)\right]\right)^2$$

Solving this equation for $V_z$, we obtain, $V_z = 2.2$ V. Now determine the gate-to-source voltage of M4 as

$$V_{GS,4} = 3.5 - 2.2 = 1.3 > V_{T0,n} = 1$$

It is seen that at this point, M4 is already on. Thus, the analysis above, which is based on the assumption that M4 is not conducting, can no longer be valid. At this input voltage, node x is being pulled down toward "0." This can also be seen clearly from the simulation results. We conclude that the upper logic threshold voltage $V_{th}^+$ is approximately equal to 3,5 V.

Next, we consider a negative input sweep, i.e., assume that the input voltage is decreasing from $V_{DD}$ to 0.

i) At $V_{in} = 5.0$ V:

M4 and M5 are on, so that the output voltage is $V_x = 0$ V. The pMOS transistors M1 and M2 are off, and M3 is in saturation, thus,

$$\frac{1}{2}k'\left(\frac{W}{L}\right)_3 \left(0 - V_y - V_{T,3}\right)^2 = 0$$

$$V_y = -V_{T,3} = -\left[V_{T0,p} - 0.4\left(\sqrt{0.6 + V_{DD} - V_y} - \sqrt{0.6}\right)\right]$$

$$V_y = 1.5\ [\text{V}]$$

ii) At $V_{in} = 4.0$ V:

M1 is at the edge of turning on, M2 is off, and M3 is in saturation. The output voltage is still unchanged.

iii) At $V_{in} = 3.0$ V:
M1 is on and in saturation region. M3 is also in saturation, thus,

$$\frac{1}{2}k'\left(\frac{W}{L}\right)_1 \left(V_{in} - V_{DD} - V_{T0,p}\right)^2 = \frac{1}{2}k'\left(\frac{W}{L}\right)_3 \left(0 - V_y - V_{T,3}\right)^2$$

$$\left[3 - 5 - (-1)\right]^2 = 3\left(0 - V_y - \left[-1 - 0.4\left(\sqrt{0.6+5-V_y} - \sqrt{0.6}\right)\right]\right)^2$$

The solution of this equation yields:

$$V_{in} = 2.02 \text{ V}$$

Now we determine the gate-to-source voltage of M2 as

$$V_{GS,2} = 3.0 - 2.02 = 0.98 > V_{T0,p} = -1$$

which indicates that M2 is still turned off at this point.

iv) At $V_{in} = 1.5$ V:
If M2 is still off, M1 is in the linear region, and M3 is in the saturation region:

$$\frac{1}{2}k'\left(\frac{W}{L}\right)_1\left(2\left(V_{in} - V_{DD} - V_{T0,p}\right)\left(V_y - V_{DD}\right) - \left(V_y - V_{DD}\right)^2\right)$$
$$= \frac{1}{2}k'\left(\frac{W}{L}\right)_3\left(0 - V_y - V_{T,3}\right)^2$$

$$2(1.5 - 5 + 1)\left(V_y - 5\right) - \left(V_y - 5\right)^2$$
$$= 3\left(-V_y - \left[-1 - 0.4\left(\sqrt{0.6 + 5 - V_y} - \sqrt{0.6}\right)\right]\right)^2$$

Solving this quadratic equation yields

$$V_y = 2.79 \text{ V}$$

It can be shown that at this point, the pMOS transistor M2 is already turned on. Consequently, the output voltage is being pulled up to $V_{DD}$. We conclude that the lower logic threshold voltage $V_{th}^-$ is approximately equal to 1.5 V.

The SPICE simulation results are plotted below for both increasing and decreasing input voltages. The expected hysteresis behavior and the two switching thresholds are clearly seen in the simulation results.

## References

1. C. Mead and L. Conway, *Introduction to VLSI Systems*, Reading, MA: Addison-Wesley Publishing Company, Inc., 1980.

2. F.J. Hill and G.R. Peterson, *Computer-Aided Logical Design with Emphasis on VLSI*, fourth edition, John Wiley & Sons, Inc., New York, 1993.