CHAPTER
7

Transfer Gate and Dynamic Logic Design

CHAPTER OUTLINE

7.1 Introduction

In previous chapters, we examined the characteristics of *static* logic gates. Two varieties of static gates were presented: conventional CMOS and pseudo-NMOS gates. Standard CMOS gates dissipate far less power than their pseudo-NMOS counterparts but require more area due to the number of complementary PMOS transistors needed, and their size requirements to achieve equal rise/fall delays. Usually the PMOS device is twice the size of the NMOS device in a conventional CMOS inverter. However, the pseudo-NMOS gates require ratioed devices to set the desired value of V_{OL}; typically, the NMOS device is four times the size of the PMOS device in a pseudo-NMOS inverter. Unfortunately, this produces asymmetric rise and fall times. If we increase the size of the PMOS device to try to equalize the delays, the value of V_{OL} increases. In order to strike a balance between conventional CMOS and pseudo-NMOS circuits in terms of area, timing, and power, we turn our attention to *dynamic logic* techniques.

Earlier, we found that all nodes of a static gate have direct paths to V_{DD} or Gnd. So long as the power remains on, the value of the output node is held indefinitely.

Dynamic gates, on the other hand, store their value on a capacitor. The storage nodes are often isolated from the rest of the circuit for long periods of time. Therefore, node values may decay over time if not refreshed or updated periodically. A dynamic node is also more susceptible to noise events than static gates. It is in this sense that the name *dynamic* is applied—the node voltage is held somewhat precariously by the charge stored at the node. This feature renders dynamic logic less desirable than static circuits for most designers. However, if we are careful about how we design this type of circuit, it can outperform static logic circuits. For this reason, we explore dynamic design techniques in this chapter.

One form of dynamic logic uses transfer gates as switches to propagate information through the circuit. When the switches are off, their outputs are held in a *high-impedance* (or high-Z) state implying that the gate is no longer driving the output. In this condition, the previous value is stored as charge on the output capacitance. A second form of dynamic logic relies on additional clocking signals for proper operation. On one part of the clock cycle, all logic gate outputs are *precharged* to an initial value. On the other part of the cycle, the gates *evaluate* their correct output values. We now discuss these two types of dynamic logic circuits in detail.

7.2 Basic Concepts

We begin our study of dynamic circuits by considering the pass transistor or transfer gate (sometimes referred to as a pass gate or transmission gate). These devices will expose the advantages and limitations of using a single transistor as a switch. There are three basic concepts that are important to dynamic circuits. These are charge sharing, feedthrough, and charge leakage.

7.2.1 Pass Transistors

The role of a pass transistor is to transfer an input signal, unaltered, to the output node when the gate is *on*. When the gate is turned *off*, the output enters the high-Z state and holds its previous value. This type of logic gate has an input, an output, and a third terminal that controls whether the device is *on* or *off*. In Figure 7.1, single n-channel and p-channel implementations of the transfer gate are shown under various input conditions. The source and drain nodes serve as inputs and outputs, while the gate node serves as the control input.

Figure 7.1 illustrates the signal transmission capability of each type of transfer gate in the *on* and *off* conditions. Figure 7.1a demonstrates that the n-channel device can successfully pass 0 V when its control input is at V_{DD}. Similarly, the PMOS device propagates V_{DD} from input to output when its control input is 0 V, as in Figure 7.1b. However, both NMOS and PMOS devices have problems passing the opposite signal level. The NMOS device in Figure 7.1c has trouble passing V_{DD}, since it shuts off when the output rises to $V_{DD} - V_{TN}$, with full accounting for body effect. The PMOS has a similar problem with low inputs, as shown in Figure 7.1d. When the control input shuts the device *off*, the output enters the high-Z state, as in Figure 7.1e and Figure 7.1f.

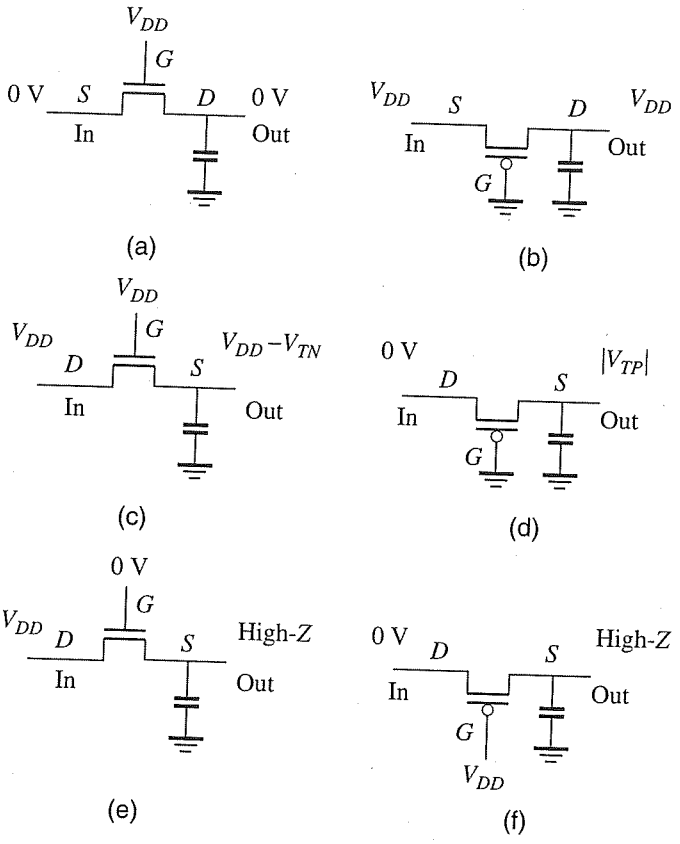


Figure 7.1
NMOS and PMOS pass transistor configurations.

The reason for difference in transmission capability of the NMOS device can be understood by comparing Figure 7.1a and Figure 7.1c. The drain (D), gate (G), and source (S) have been identified to clarify the description. For an NMOS device, the drain is the higher of the two nodes. Since the MOS transistor is a symmetric device, the determination of the drain and source can only be carried out after the node voltages have been assigned. In Figure 7.1c, the drain is connected to V_{DD} since this is guaranteed to be the highest node. If the source node starts at 0 V, the device is *on* and in the saturation of operation. Current flows from drain to source to charge up the output capacitance and the source node voltage begins to increase. It continues to increase until $V_{GS} = V_{TN}$. At this point, the output voltage is $V_{DD} - V_{TN}$. In Figure 7.1a, the drain and source terminals are reversed. The value of V_{GS} is always V_{DD}. As such, the device is always *on* and will pull the output node low until $V_{DS} = 0$, at which point the output is 0 V.

For the p-channel device, we can go through the same exercise for Figure 7.1b and Figure 7.1d. This time the definition of the drain and source are reversed such that the higher voltage is the source and the lower voltage is the drain. In Figure 7.1b, the drain node begins at 0 V and rises as the PMOS device charges it to V_{DD}. The output can reach V_{DD} since the value of V_{GS} is constant at $-V_{DD}$ implying that

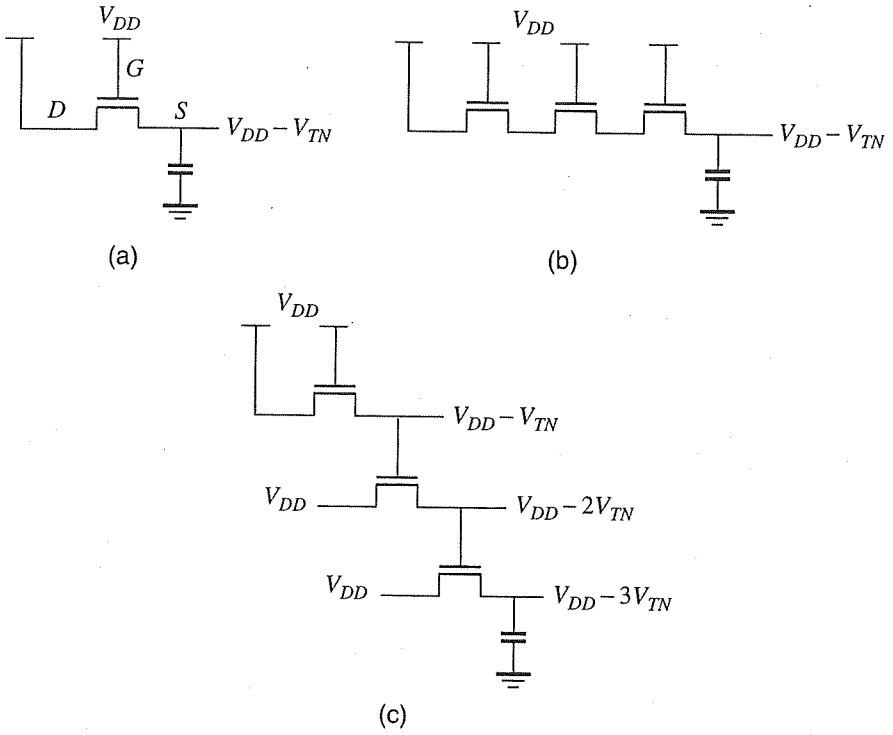


Figure 7.2
NMOS pass gate configurations (ignoring body effect).

the PMOS device is always *on*. In Figure 7.1d, the source node begins at V_{DD} and drops in voltage as the output capacitance is discharged until the value of $V_{GS} = |V_{TP}|$. At this point, the device shuts off. Therefore, the lowest voltage that the output can reach is $|V_{TP}|$.

Figure 7.2 shows three *n*-channel pass transistor configurations to further clarify the operation. Note that the two circuits of Figure 7.2a and Figure 7.2b produce the same output, $V_{DD} - V_{TN}$. At first glance, it would seem that Figure 7.2b with three series transistors would produce an output of $V_{DD} - 3V_{TN}$, but this is incorrect. Each pass transistor is able to pass a value of $V_{DD} - V_{TN}$ from input to output when the gate voltage is V_{DD}. Figure 7.3c illustrates the actual configuration that would produce an output of $V_{DD} - 3V_{TN}$. In each successive pass transistor, the gate voltage is V_{TN} lower than the previous transistor. The output can only climb within a threshold voltage of the gate voltage before the device shuts off. Note that the more accurate value of the output is in fact $V_{DD} - V_{TN1} - V_{TN2} - V_{TN3}$ due to the change in the body effect term in each transistor. The reader should be aware that this example is contrived since designers should avoid using the output of a pass transistor to drive the gate of another pass transistor.

For completeness, the *p*-channel equivalents for Figure 7.2 are given in Figure 7.3. The same arguments for the *n*-channel configurations can be applied to the *p*-channel devices, with the corresponding changes in output voltage levels.

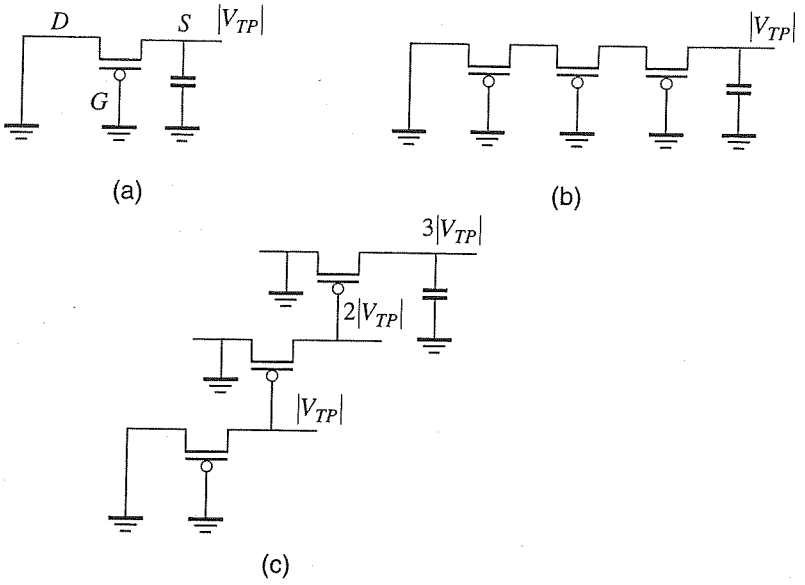


Figure 7.3
PMOS pass gate configurations (ignoring body effect).

7.2.2 Capacitive Feedthrough

The role of the control node of a pass transistor is to place the device in the *on* and *off* states. As such, it is often driven by a clock signal. Ideally, this input should not have any direct influence on the output, except to enable or disable the transfer gate. However, there exists a capacitance, C_f, between the gate and output nodes that causes the clock signal to feed through to the output. This effect is illustrated in Figure 7.4. The control signal undergoes a voltage swing from V_{DD} to 0 V, thereby shutting off the device. This places the output in the high-Z state since $V_{GS} < V_{TN}$. At this point, the two capacitors, C_f and C_{gnd}, are isolated from the rest of the circuit. As the gate voltage decreases in value, the charges associated with the two capacitors

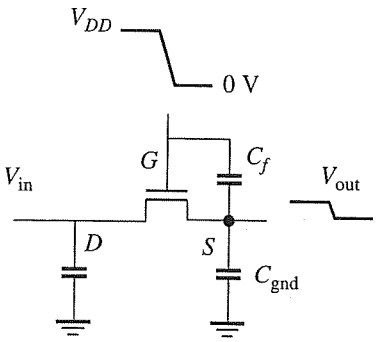
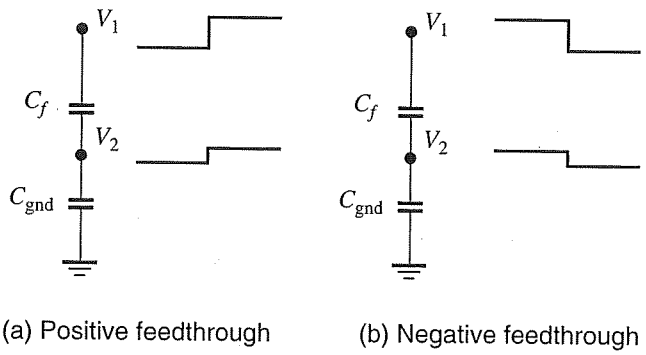


Figure 7.4
Clock feedthrough.



(a) Positive feedthrough (b) Negative feedthrough

Figure 7.5
Dynamic circuit effects of capacitive feedthrough.

redistribute to maintain equality. This redistribution, called displacement current, forces the output node to drop in voltage by an amount determined by the relative values of C_f and C_{gnd}. The fact that a small replica of the clock appears at the output led to the terminology of *clock feedthrough* for this effect. The same effect would be observed at the drain node if it were isolated from the rest of the circuit. However, we have assumed that drain is driven by a static gate in this example.

The capacitive effects associated with feedthrough may occur in the positive or negative direction. We can understand these effects by examining isolated capacitors in series as shown in Figure 7.5. For a series combination of two capacitors, positive and negative feedthrough result in noise injection at an internal node when the external node rises or falls. The change at the external node 1 produces a replica change at the internal node 2, which is a high-impedance node.

The exact voltage change at the internal node can be determined for the idealized cases of Figure 7.5. The charges associated with the two capacitors must be the same at equilibrium:

$$C_f(V_1 - V_2) = C_{gnd}V_2 \tag{7.1}$$

Therefore,

$$V_2 = \frac{C_f V_1}{C_f + C_{gnd}} \tag{7.2}$$

If there is an abrupt change in voltage at node 1, there is a corresponding but smaller change at node 2:

$$\Delta V_2 = \frac{C_f \Delta V_1}{C_f + C_{gnd}} \tag{7.3}$$

In Figure 7.5a, when there is a large positive change in V_1, the internal node voltage will change by an amount dictated by the ratio of C_f to $(C_f + C_{gnd})$. The change at node 2 will be in the positive direction. This is sometimes referred to as

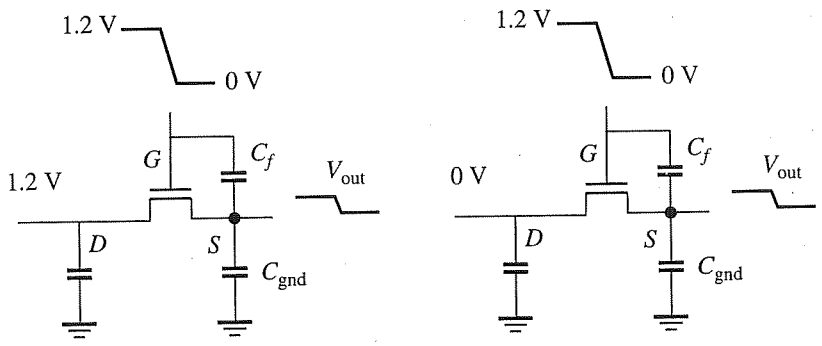
bootstrapping.[1] In Figure 7.5b, we have a negative going step at node 1 that produces a replica negative going step at node 2. Consider the two extreme cases of feedthrough. If C_f is much larger than C_{gnd}, then ΔV_2 approaches ΔV_1. On the other hand, if C_{gnd} is much greater than C_f then ΔV_2 approaches 0. For the pass transistor, the value of C_f is determined by C_{GS} while C_{gnd} is determined by the junction capacitance, C_{BS}, plus the loading capacitance of the next gate. To reduce the feedthrough effect, we should ensure that $C_{gnd} \gg C_f$.

Example 7.1

V_T Drop and Clock Feedthrough Effects

Problem:

In the circuit below with the input at 1.2 V, what is the initial value of the output when the clock is at 1.2 V. Estimate the final value after the clock goes low. Repeat the problem for the case when the input is 0 V. Assume that the device is 4λ/2λ in 0.13 μm technology.



Solution:

Initially, the output will be a threshold voltage below 1.2 V:

$$V_{out} = V_{DD} - V_{TN} = 1.2 - (0.4 + 0.2(\sqrt{0.88 + V_{out}} - \sqrt{0.88}))$$

$$\therefore V_{out} = 0.73 \text{ V}$$

When the clock switches from high to low, the feedthrough effects will act to reduce the value at the output:

$$V_{out} = 0.73 - \Delta V = 0.73 - \frac{C_f(1.2)}{C_f + C_{gnd}}$$

In this example, C_f is due to the overlap capacitance since the transistor is in the cutoff region. Therefore,

$$C_f = C_{OL} = C_{ol}W = (0.25 \text{ fF}/\mu\text{m}) \times 0.2 \ \mu\text{m} = 0.05 \text{ fF}$$

[1] Bootstrapping is usually associated with increasing a node voltage above V_{DD} using active devices. We use the term here more generally to refer to any increase in the voltage due to series capacitor effects.

The value of C_{gnd} is due to the junction capacitance. Therefore,

$$C_{gnd} = C_{eff} \times W = 1 \text{ fF}/\mu\text{m} \times 0.2\ \mu\text{m} = 0.2 \text{ fF}$$

The resulting value of V_{out} is

$$V_{out} = 0.73 - \frac{C_f(1.2)}{C_f + C_{gnd}} = 0.73 - \frac{0.05(1.2)}{0.05 + 0.2} = 0.73 - 0.24 \approx 0.5 \text{ V}$$

For this particular case, the clock feedthrough effect is significant even though we are only considering the overlap capacitance. The resulting voltage is below typical switching thresholds and this would be a problem. In a realistic situation, there is always additional grounded capacitance at the output due to the fanout gates. This would greatly reduce the degree of clock feedthrough.

When the input is 0 V, the output is 0 V. However, in this case, the device is in the linear region of operation. Therefore,

$$C_f = 1/2\ C_gW + C_{ol}W = (1/2)(2 \text{ fF}/\mu\text{m})(0.2\ \mu\text{m}) + (0.25 \text{ fF}/\mu\text{m})(0.2\ \mu\text{m})$$
$$= 0.25 \text{ fF}$$

$$V_{out} = 0 - \frac{C_f(1.2)}{C_f + C_{gnd}} = 0 - \frac{0.25(1.2)}{0.25 + 0.2} = 0 - 0.67 = -0.67 \text{ V}$$

In this case, the voltage drop is substantially more than the first case since the device is in the linear region. However, the device does not shut off until $V_{GS} < V_{TN}$. Therefore, this result is not as accurate as the previous case, but also not as important. The feedthrough equation is intended for use at high-Z nodes only.

7.2.3 Charge Sharing

Another important issue in dynamic circuits is charge sharing which tends to reduce the voltage levels even further. This phenomenon occurs when two isolated nodes with different voltage levels are suddenly connected together when a pass transistor turns on, as shown in Figure 7.6. The key requirement is that the two nodes are in the high-impedance state and store different voltages. When the switch turns on, the charge redistributes until the voltage levels at the two nodes are equal. This reduces the voltage at one node while increasing the voltage at the other. If the voltage is reduced at an output node, it is of concern to us. We can analyze charge sharing by examining the total charge before and after the charge-sharing event.

In Figure 7.6a, the total charge in the closed system is

$$Q_{total} = C_1V_1 + C_2V_2 \tag{7.4}$$

When the gate node is switched from 0 to V_{DD}, as in Figure 7.6b, the transistor turns on and a charge transfer occurs between the two nodes until an equilibrium voltage

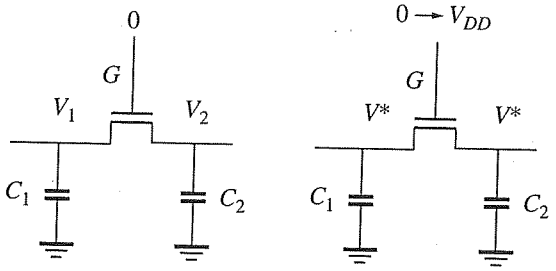


Figure 7.6
Dynamic charge sharing.

is reached. At equilibrium, the total charge in the system remains constant. If V^* is the final voltage after charge exchange is completed, the total charge is given by

$$Q_{\text{total}} = (C_1 + C_2)V^* \tag{7.5}$$

Combining (7.4) and (7.5), we obtain

$$V^* = \frac{(C_1V_1 + C_2V_2)}{C_1 + C_2} \tag{7.6}$$

As an illustration of charge sharing, assume that $V_1 = 0$ and $V_2 = V_{DD} - V_T$ in Figure 7.6a. After the transistor turns on, the voltage at V_2 will drop in value and the voltage at V_1 will increase. Equation (7.6) specifies the final voltage on the two nodes. If V_2 drops below the switching threshold of the next gate, it will trigger the gate to produce an incorrect output. This is a serious consequence of charge sharing. To reduce charge sharing in this case, C_2 should be made much larger than C_1. There are additional techniques that can be employed to reduce the effects of charge sharing that will be described later in this chapter.

Equation (7.6) should be used carefully in conjunction with Figure 7.6 since it is only valid if V^* does not exceed $V_{DD} - V_{TN}$. If the computed voltage goes beyond this value, then V_1 should be set to this value and then V_2 re-calculated with the remaining charge, $Q_{\text{total}} - C_1 (V_{DD} - V_{TN})$. In this case, $V_1 \neq V_2$ after charge sharing.

Charge Sharing Example **Example 7.2**

Problem:

In Figure 7.6a, compute the charge-sharing effects for the following cases assuming 0.13 μm technology parameters:

(a) $C_1 = 100$ fF, $C_2 = 20$ fF, $V_1 = 0$, $V_2 = 1.2$ V

(b) $C_1 = 20$ fF, $C_2 = 20$ fF, $V_1 = 0$, $V_2 = 1.2$ V

(c) $C_1 = 20$ fF, $C_2 = 100$ fF, $V_1 = 0$, $V_2 = 1.2$ V

Solution:

(a) $V^* = \dfrac{(C_1V_1 + C_2V_2)}{C_1 + C_2} = \dfrac{(100\text{ fF} \times 0 + 20\text{ fF} \times 1.2)}{(20\text{ fF} + 100\text{ fF})} = 0.2\text{ V}$

(b) $V^* = \dfrac{(C_1V_1 + C_2V_2)}{C_1 + C_2} = \dfrac{(20\text{ fF} \times 0 + 20\text{ fF} \times 1.2)}{(20\text{ fF} + 20\text{ fF})} = 0.6\text{ V}$

(c) $V^* = \dfrac{(C_1V_1 + C_2V_2)}{C_1 + C_2} = \dfrac{(20\text{ fF} \times 0 + 100\text{ fF} \times 1.2)}{(20\text{ fF} + 100\text{ fF})} = 1.0\text{ V}$

This last solution is not possible since the voltage of node V_1 cannot rise above $V_{DD} - V_{TN}$. Ignoring body effect, the maximum voltage at node V_1 is 0.8 V. The total charge in the system is 120 fC. Node V_1 requires 0.8×20 fF = 16 fC of charge. That leaves 120 − 16 = 104 fC of charge on node V_2. This implies that the voltage at node V_2 = 104 fC/100 fF = 1.04 V.

7.2.4 Other Sources of Charge Loss

When a pass transistor shuts off, the output node enters a high-Z state and retains the value of the node as charge on a capacitance. Unfortunately, the charge tends to leak away over time unless the value is updated or refreshed periodically. If a high value is stored, there are two possible sources of charge leakage. One source is reverse-bias leakage current from the drain junction. This current is rather small as it depends on the area of the junction which continues to shrink as technology scales. Of greater concern is the subthreshold current as described in Chapter 2.

A second source of charge loss is noise injection from neighboring wires. For example, if a wire is routed too closely to a dynamic node, capacitive coupling between the wire and the node could result in noise injection during switching events. The mechanism is similar in nature to feedthrough and bootstrapping. This form of noise injection can have serious consequences and must be considered carefully when designing dynamic circuits. Coupling noise is the subject of Chapter 10 on interconnect.

Another problem is associated with α-particles which are also responsible for data loss in memory circuits. These are high-energy particles that may be present in device packaging materials, such as solder bumps, or due to cosmic rays arriving from space. Particles of this nature can enter the silicon substrate and generate hole-electron pairs that act to discharge the output. This occurs over a very short amount of time and leads to unexpected functional failures in the circuit. The detailed mechanisms are beyond the scope of this book, but the importance of α-particles as a leading cause of so-called *soft errors* in both dynamic and static circuits should be recognized.

7.3 CMOS Transmission Gate Logic

In the previous section, we observed the limitations of using either a single NMOS or a single PMOS device as a transfer gate. Each one is capable of transmitting one

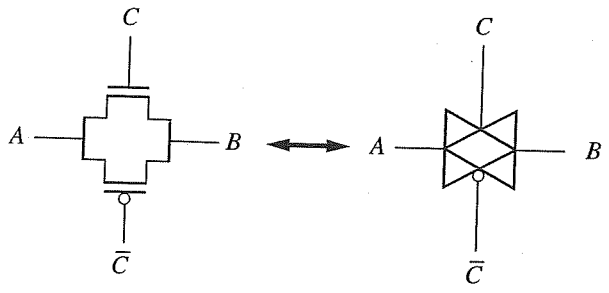


Figure 7.7
CMOS transmission gate and logic symbol.

of the levels accurately but neither device is capable of transmitting both high and low voltages properly, as evidenced by Figure 7.1. A natural improvement can be constructed by placing an NMOS device in parallel with a PMOS device to create a CMOS transmission gate, as shown in Figure 7.7. The two devices in combination can fully transmit any signal value between V_{DD} and Gnd. The cost of this feature is not only an extra transistor, but also an extra inverter since each transistor requires a complementary control input. Therefore, a total of four transistors are necessary for the CMOS version. The corresponding symbol for the transmission gate is also shown in Figure 7.7.

To understand the transmission capability of the CMOS transfer gate, consider the two cases shown in Figure 7.8. The two transistors have been separated out to clearly identify their role in transmitting 0 V or V_{DD}. In Figure 7.8a, a 0 V signal is being transmitted through both devices with an initial output of V_{DD}. The NMOS device has $V_{GS} = V_{DD}$ for the entire duration of the transition of the output from V_{DD} to 0 V. Therefore, it can pull the output to 0 V by itself. The PMOS device can only pull the output to $|V_{TP}|$ since it shuts off at that point. Note the positions of the gate and source terminals. The V_{GS} value of the PMOS device decreases in magnitude

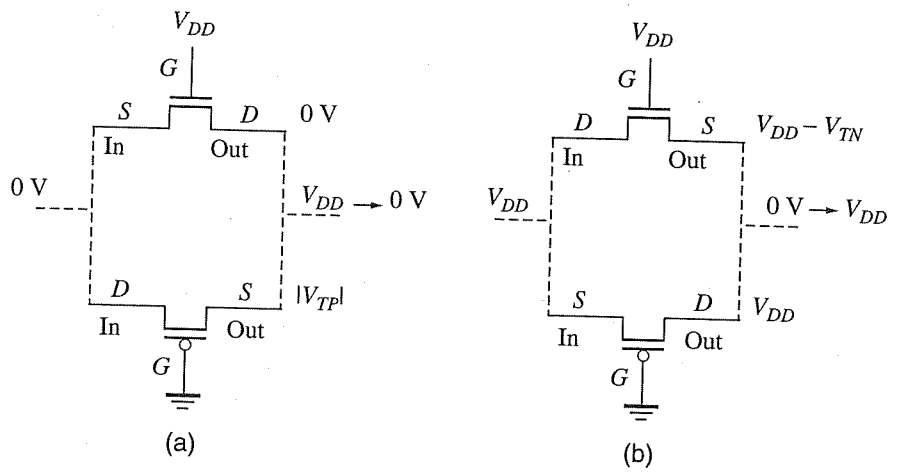


Figure 7.8
Transmitting low and high signals.

as the output falls from V_{DD} to 0 V, and it gets progressively weaker. In fact, the NMOS device pulls the output from $|V_{TP}|$ to 0 V after the PMOS device shuts off completely.

The roles of the two devices are reversed in Figure 7.8b, where the transmission gate is attempting to pass an input of V_{DD} to the output with an initial condition of 0 V. In this case, the NMOS device becomes progressively weaker as the output increases since its V_{GS} value is decreasing. The PMOS device has a V_{GS} value that is constant at $-V_{DD}$. Therefore, it remains on during the entire output transition. When the output reaches $V_{DD} - V_{TN}$, the NMOS device shuts off and the PMOS device pulls the output the rest of the way to V_{DD}. By examining these two cases, it is clear that the weakness of one device is overcome by the strength of the other device, whether the output is transmitting a high or low value. This is a clear advantage of the CMOS transfer gate over the single transistor counterpart.

Another advantage of the CMOS transfer gate is that the degree of capacitive feedthrough is reduced. This is because it is clocked with the true and complement values of the enable signal. Referring to Figure 7.7, when the signal C falls, there may be feedthrough at node B, but it would be canceled by the transition to a high voltage at $\overline{C}$. If the target voltage of B is low, feedthrough due to C would only make it lower. So this case is not a problem. If the intended voltage at B was high, then feedthrough would normally make the voltage lower. But since the p-channel device remains *on* for slightly longer than the n-channel device, it is able to pull the output back to V_{DD}.

7.3.1 Multiplexers Using CMOS Transfer Gates

CMOS transmission gate logic can be used to reduce the number of transistors needed to implement certain logic functions. They can be used in a variety of situations, but the most popular is in the role of a multiplexer as shown in Figure 7.9. The control signal S turns the transfer gates *on* and *off* depending on its value. Specifically, when $S = 1$, the upper transfer gate is *on* and that allows A to flow to the output. When $S = 0$, the lower transfer gate is *on*, and that allows B to flow from input to output. Therefore, the function of this configuration is $F = AS + B\overline{S}$. This

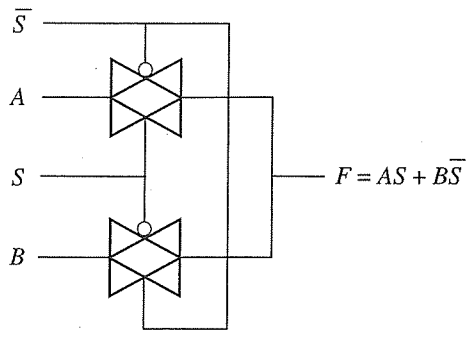


Figure 7.9
Multiplexer configuration.

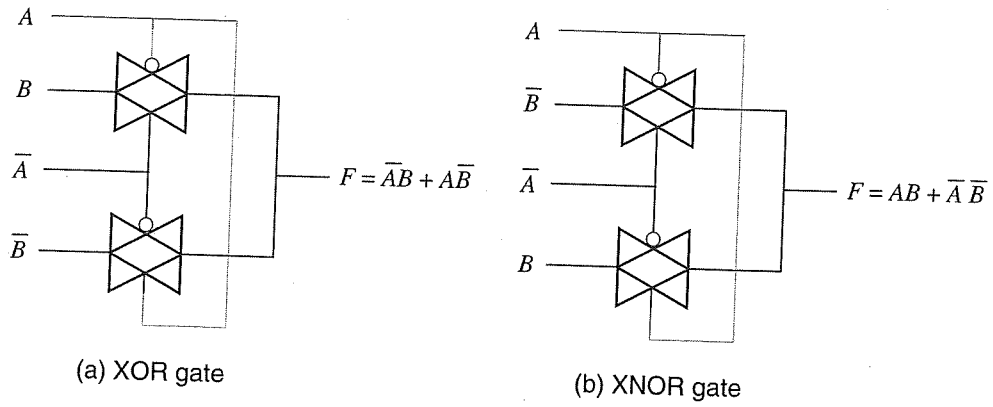


Figure 7.10
XOR and XNOR gates using CMOS transmission gates.

is a multiplexer where one of two input signals is selected based on the value of a control signal.

Assuming that certain rules are followed, it is a simple matter to construct other useful circuits using the multiplexer of Figure 7.9. By specifying the inputs and control signals in a particular pattern, we can construct XOR and XNOR gates as shown in Figure 7.10. A static CMOS gate version would require 16 transistors whereas the transfer gate version requires only 8 transistors (including all necessary inverters).

Another example is shown in Figure 7.11. Here, two multiplexer configurations that select one-of-four inputs are shown. In Figure 7.11a, a two-level strategy is used, while in Figure 7.11b a single-level is used. The two-level approach requires

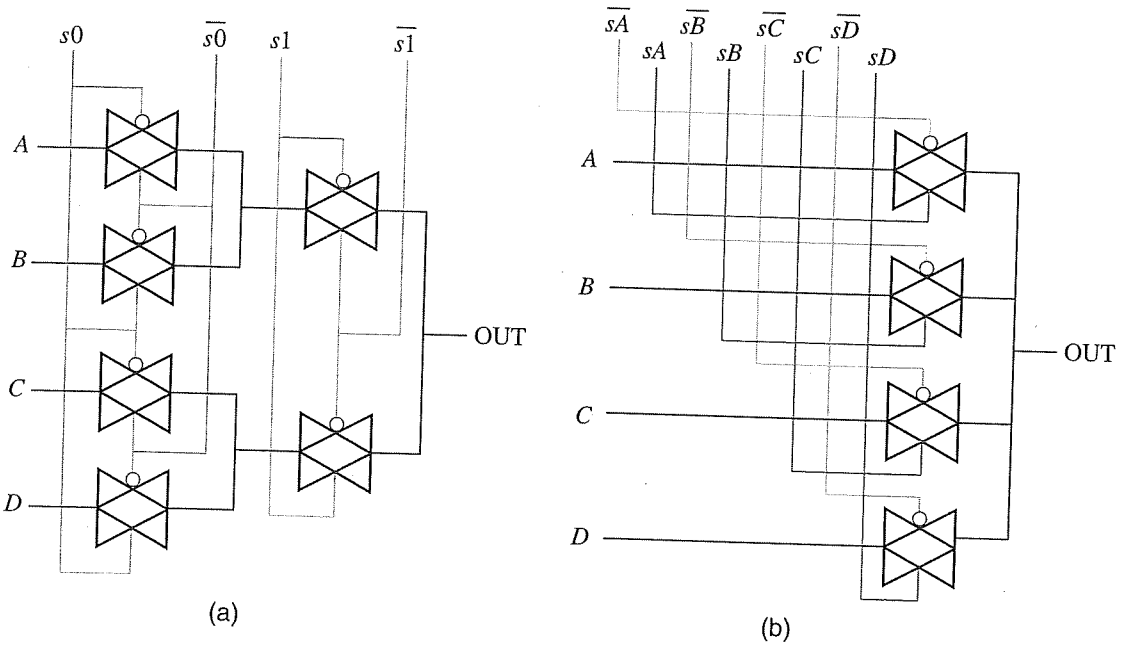


Figure 7.11
Two-level and single-level multiplexer configurations.

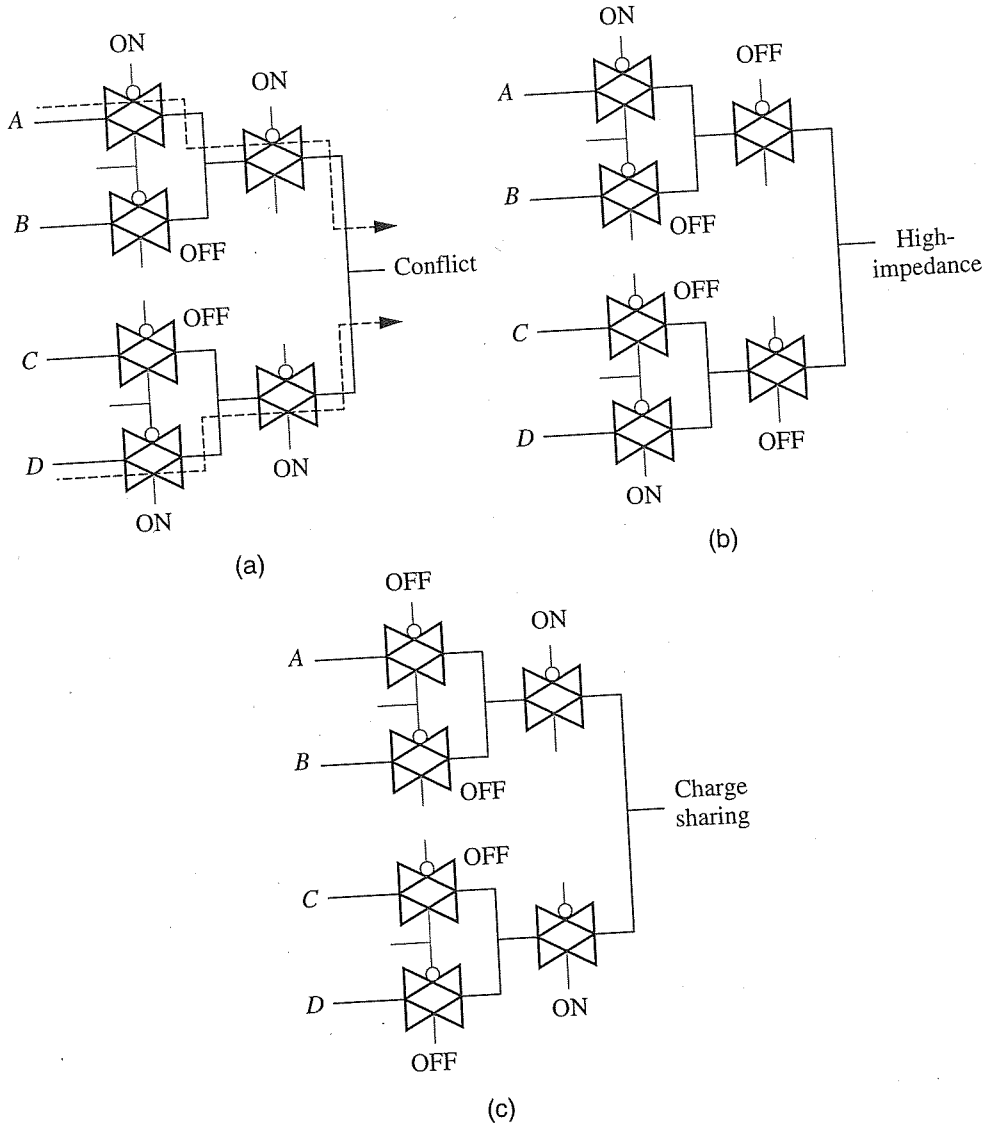


Figure 7.12
Improper conditions for multiplexer style logic.

only two control signals, $s0$ and $s1$, and their complements. The single-level approach looks more attractive but requires four control signals, sA, sB, sC, and sD, and their complements. There are more inverters required and more significant signal routing issues in this case. We will study the speed of the two approaches in a later section.

For this type of MUX function to work properly, the control and data inputs must be carefully specified. First, the data inputs to the multiplexer must all be valid, and second, the control signals must turn on only one path at a time. If these rules are not followed, conflicts or charge sharing may occur and corrupt the value at the output. For example, if there are multiple paths to the output, as in Figure 7.12a, a

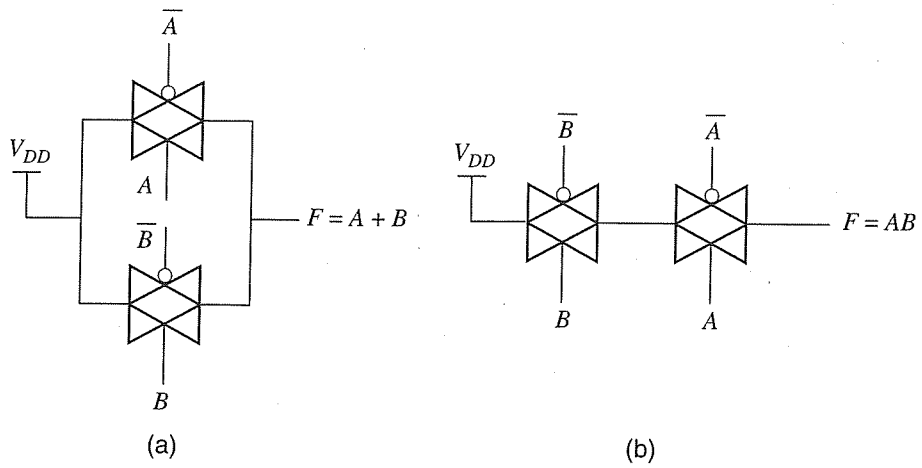


Figure 7.13
OR and AND functions using CMOS transmission gates.

conflict would arise at the output if A and D had different values. If there is no path to the output due to *off* devices in all paths, the output will assume a high-impedance state as in Figure 7.12b. If intermediate gates are *on*, as in Figure 7.12c, then an unknown may result at the output (possibly due to charge sharing).

To build general logic functions using transmission gates, the first step is to select the control signals and build a truth table for all the possible combinations. Then, the truth table is converted to a multiplexer-style design by creating a signal path in the circuit for each row of the truth table. Next, the desired output values specified in the truth table are routed from the data inputs to the output. Any needed OR and AND operations can be realized using parallel and series transmission gates as shown in Figure 7.13. Finally, optimization of the design may be carried out to combine paths or remove unnecessary transistors.

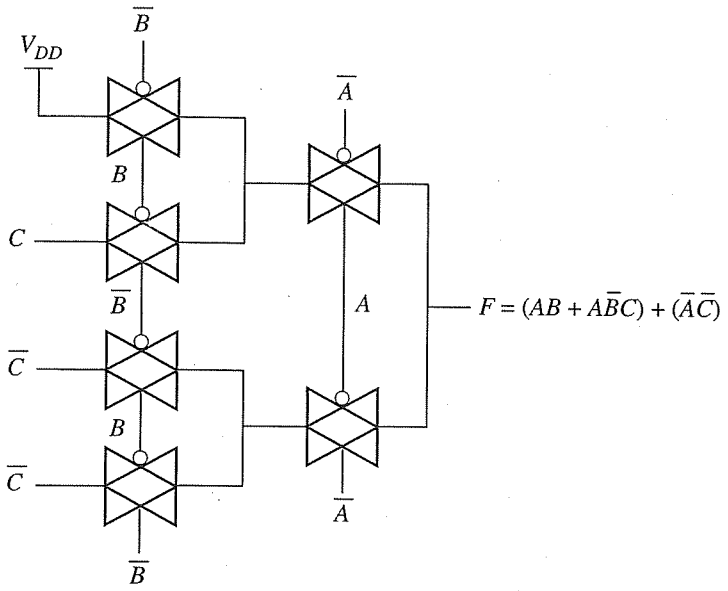
Implement the function $F = AB + A\overline{B}C + \overline{A}\,\overline{C}$ using transmission gates. **Example 7.3**

Solution:

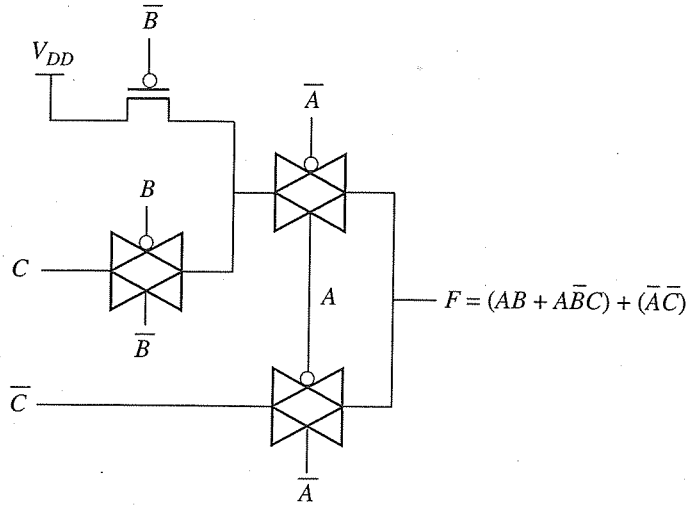
Identify A and B as control signals and build a truth table.

A	B	F
0	0	$\overline{C}$
0	1	$\overline{C}$
1	0	C
1	1	1

This truth table can be directly implemented as shown below using F as data input.



The first two lines of the truth table are associated with the lower part of the multiplexer. They can be combined to simplify the implementation. In addition, the transfer gate connected to V_{DD} does not need an n-channel device since it is only passing a high value. With these two optimizations, the following circuit results:



CMOS transmission gates can also be used with combinational blocks to implement certain functions efficiently. The first circuit in Figure 7.14 is a six-transistor XOR gate. The second circuit simply illustrates the use of transmission gate logic with standard CMOS logic to implement a complex function. In order to quickly extract the function F from the given circuit, follow the NMOS side of the transmission gates.

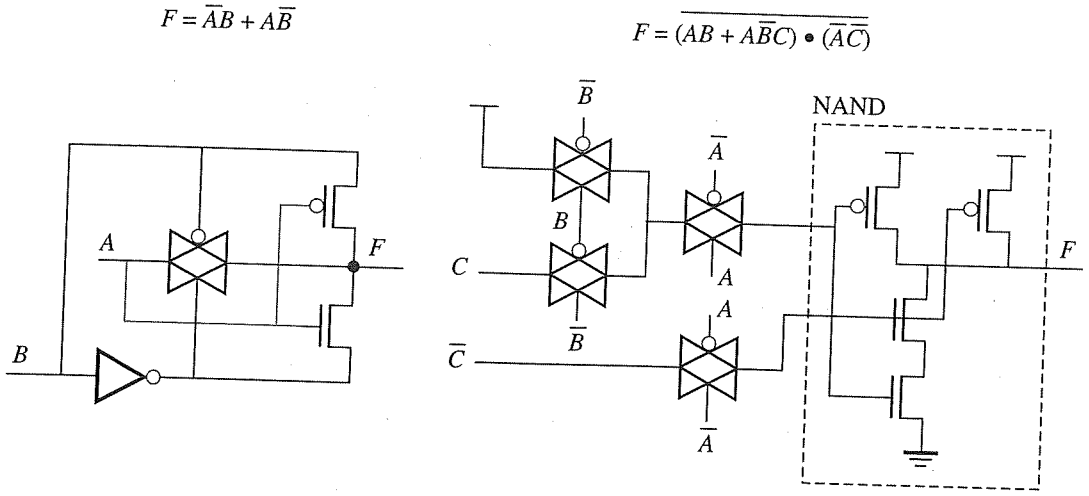


Figure 7.14
Transmission gates and standard gate combinations.

In this case, the NAND operation is being performed on the two inputs. One input is $\bar{A}\bar{C}$, while the other input is $AB + A\bar{B}C$. The resulting function F is the NAND of these two terms as shown in the diagram.

Describe the operation of the XOR gate of Figure 7.14. Discuss its advantages and disadvantages relative to other XOR circuits considered so far in this textbook. **Exercise 7.1**

7.3.2 CMOS Transmission Gate Delays

In order to evaluate the timing performance of circuits containing transmission gates, an *RC* model of these gates must be developed. The model for a CMOS transfer gate (TG) consists of an on-resistance, R_{TG}, and two capacitances, C_1 and C_2, as shown in Figure 7.15. The on-resistance of the transfer gate is a parallel combination of the on-resistances of the NMOS and PMOS devices. The value of the resistances will depend on whether a 0 or a 1 is being propagated through the gate. The

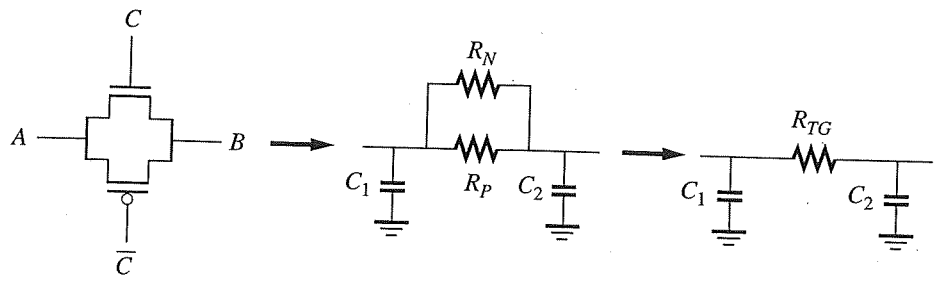


Figure 7.15
RC model for CMOS transmission gate.

resulting *RC* model must be combined with the *RC* models for the driver and load of the transfer gate. It should be noted here that transmission gates do not, by themselves, have any drive capability. Their role is to transmit signals from input to output under gate control. Some type of gate must be connected at the input as the driver for the signal in all circuits that use transfer gates.

A detailed analysis of the on-resistance can be performed by considering the two cases shown previously in Figure 7.8. The large-signal resistance can be computed using the equation:

$$R_{\text{on}} \approx \frac{V_{DS}}{I_{DS}}$$

If we return to Figure 7.8a, the input is 0 V and the output undergoes a transition from V_{DD} to 0 V. The NMOS device begins in saturation and then switches to the linear region as the output voltage drops to 0 V. When in saturation, the on-resistance is

$$R_N = \frac{V_{\text{out}}}{I_{\text{Dsat},n}}$$

When the device switches to the linear region, the on-resistance is

$$R_N = \frac{V_{\text{out}}}{I_{\text{Dlin},n}}$$

The PMOS device is biased in the saturation region until the output reaches $|V_{TP}|$. At this point, it enters cutoff and the resistance increases to infinity. In the saturation region, the resistance is given by

$$R_P = \frac{V_{\text{out}}}{I_{\text{Dsat},p}}$$

The on-resistances for unit size NMOS and PMOS devices, and their parallel combination are plotted in Figure 7.16a. Note that the resistance of the PMOS device increases dramatically as the output voltage switches from V_{DD} to $|V_{TP}|$, while the NMOS resistance drops linearly over the same range. However, the parallel combination of the two resistances is relatively constant. It is convenient to model this resistance by a constant value based on the average during the transition.

The same analysis can be carried out for the second case in Figure 7.8b where the input is V_{DD} and the output transitions from 0 V to V_{DD}. The PMOS device begins in saturation and then switches to the linear region as the output voltage rises to V_{DD}. When in saturation, the on-resistance is

$$R_P = \frac{V_{DD} - V_{\text{out}}}{I_{\text{Dsat},p}}$$

When the device switches to the linear region, the on-resistance is

$$R_P = \frac{V_{DD} - V_{\text{out}}}{I_{\text{Dlin},p}}$$

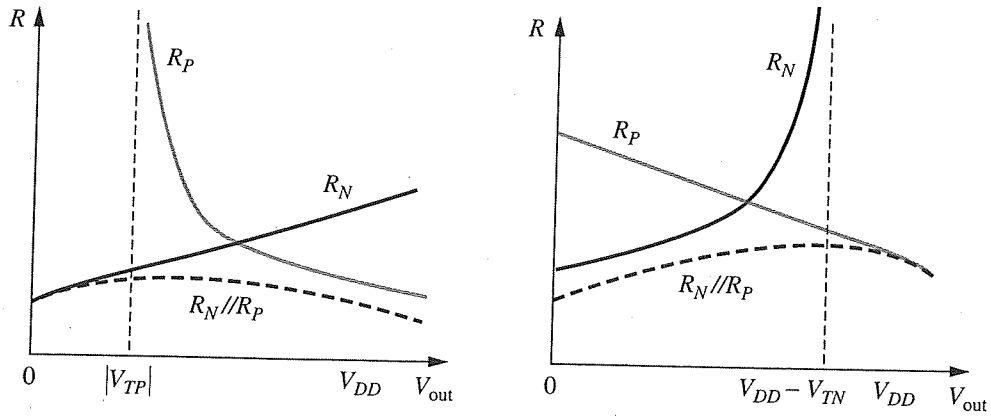


Figure 7.16
NMOS and PMOS on-resistances for falling and rising cases.

The NMOS device stays in the saturation region until the output reaches $V_{DD} - V_{TN}$. At this point, it enters cutoff and the resistance increases to infinity. In the saturation region, the resistance is given by

$$R_N = \frac{V_{DD} - V_{\text{out}}}{I_{\text{Dsat},n}}$$

The on-resistance for unit size NMOS and PMOS devices, and their parallel combination for the second case is shown in Figure 7.16b. Here, the resistance of the NMOS device increases to infinity as the output voltage approaches $V_{DD} - V_{TN}$, while the PMOS resistance decreases linearly over the same interval. Again, the parallel combination of the two resistances is relatively constant and of similar magnitude as the falling case.

A simplified analysis can be carried out to estimate the on-resistance of the combined devices for the rising and falling cases. If we are propagating V_{DD}, the PMOS device behaves as a regular pullup with an on-resistance of approximately $R_{eqp} = 30\ \text{k}\Omega/\square$. For a unit size device, the resistance is 30 kΩ. On the other hand, the *n*-channel device is trying to pass a high value and its on-resistance will increase as its source node voltage approaches V_{DD}. Since the NMOS device is shutting off, the resistance will eventually reach infinity. The average on-resistance will be roughly *double* its nominal value of $R_{eqn} = 12.5\ \text{k}\Omega/\square$. For a unit device, this value is 25 kΩ. The parallel combination of the two devices produces a value that is close to the NMOS on resistance:

$$R_{TG} = R_N//R_P = 2R_{eqn}//R_{eqp} \approx 2R_{eqn}//2.4R_{eqn} = 1.1R_{eqn} \approx R_{eqn}$$

Now consider the case when we are propagating 0 V through the transmission gate. In this case, the *n*-channel device is *on* and has a resistance of R_{eqn}. The *p*-channel

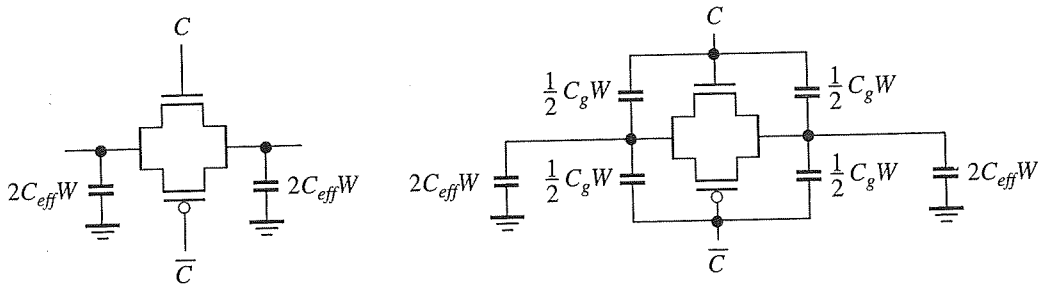


Figure 7.17
Transfer gate capacitances in *OFF* and *ON* states.

device is initially *on* and then begins to shut off as the voltage drops to 0 V. Therefore, its resistance is roughly $2R_{eqp}$, which is about five times R_{eqn}. The parallel combination of the two values produces

$$R_{TG} = R_N//R_P = R_{eqn}//2R_{eqp} \approx R_{eqn}//4.8R_{eqn} = 0.83R_{eqn} \approx R_{eqn}$$

In effect, the on-resistance of the CMOS transfer gate is approximately R_{eqn} regardless of the rising or falling conditions. Therefore, we will use this value for all resistance calculations with suitable adjustment for different W/L values:

$$R_{TG} = R_{eqn}\left(\frac{L}{W}\right) \tag{7.7}$$

We now turn our attention to the capacitance calculation. The transfer gate capacitance can be determined by examining the capacitances at the input and output for the *on* and *off* cases. When the transfer gate is *off*, as in Figure 7.17a, the C_{GS} and C_{GD} capacitances for both devices are zero, except for the overlap capacitances (not shown). In this condition, the input and output capacitances are due to the junction capacitances of the *n*-channel and *p*-channel devices.

Using C_{eff}, the input and output capacitances are

$$C_{in} = C_{out} = C_{eff}(W_n + W_p) \tag{7.8}$$

Since the two devices are the same size this simplifies to

$$C_{in} = C_{out} = C_{eff}2W \tag{7.9}$$

Next, consider a transfer gate in the *on* state, as shown in Figure 7.17b. Here the devices are assumed to be in the linear region (which is true for some part of the transition for both devices). Therefore, the input and output nodes see half the gate capacitance from each device along with the junction capacitances (and overlap

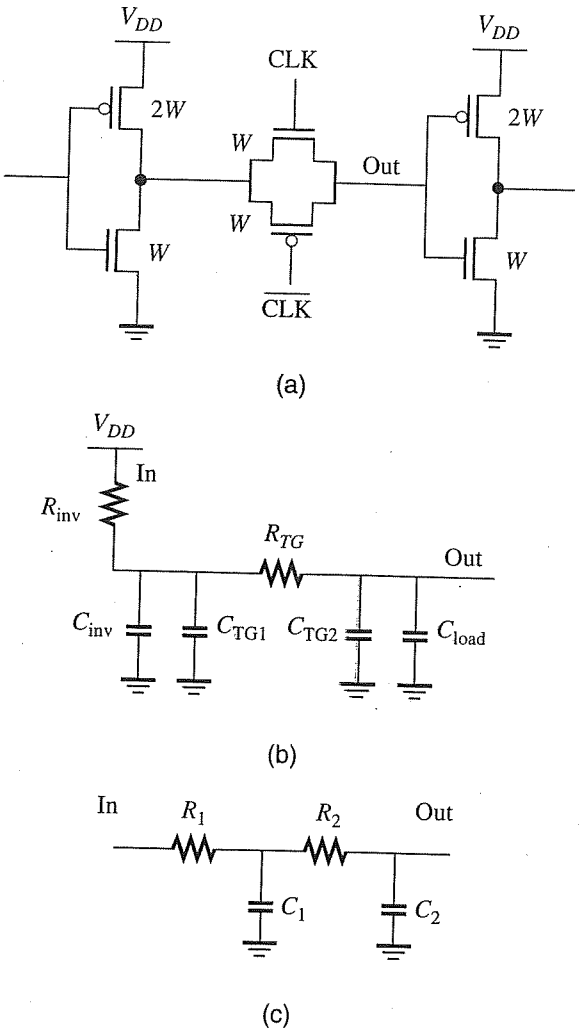


Figure 7.18
Transmission gate with driver and load.

capacitances). The total input and output capacitances are equal and can be computed as

$$C_{\text{in}} = C_{\text{out}} = C_{\text{eff}}(W_n + W_p) + \frac{1}{2}(C_g W_n + C_g W_p) \tag{7.10}$$

Note that both devices are the same size so this can be simplified to

$$C_{\text{in}} = C_{\text{out}} = C_{\text{eff}}\, 2W + C_g W \tag{7.11}$$

With the complete *RC* model defined as above, it is now possible to compute the path delay of circuits containing transmission gates. However, accurate delay calculation must involve the characteristics of the driver and load of the transmission gate. This is illustrated in Figure 7.18a where an inverter is driving the transmission

gate which is loaded by a second inverter. To compute the delay accurately, the *RC* model shown in Figure 7.18b is required. Here the inverter *RC* model is connected at the input of the transfer gate and the load capacitance is connected at the output of the transfer gate. The resulting circuit is an *RC* ladder.

The Elmore delay equations for *RC* ladders and *RC* trees are described in Chapter 10. For the purposes of our analysis here, we simply use the results derived in that chapter. Specifically, the delay of the two-stage *RC* ladder of Figure 7.18c using the Elmore delay equation is

$$t_{\text{Elmore}} = R_1C_1 + (R_1 + R_2)C_2$$

Therefore, for the *RC* network of Figure 7.18b, we would compute the delay as

$$t_{\text{Elmore}} = R_{\text{inv}}(C_{\text{inv}} + C_{\text{TG1}}) + (R_{\text{inv}} + R_{\text{TG}})(C_{\text{TG2}} + C_{\text{load}})$$

Example 7.4 For the two multiplexers in Figure 7.11, which one is faster, assuming that there exists a path from input to output? Assume that a 1X inverter drives each input and the fanout of the multiplexer is *f* times the 1X inverter.

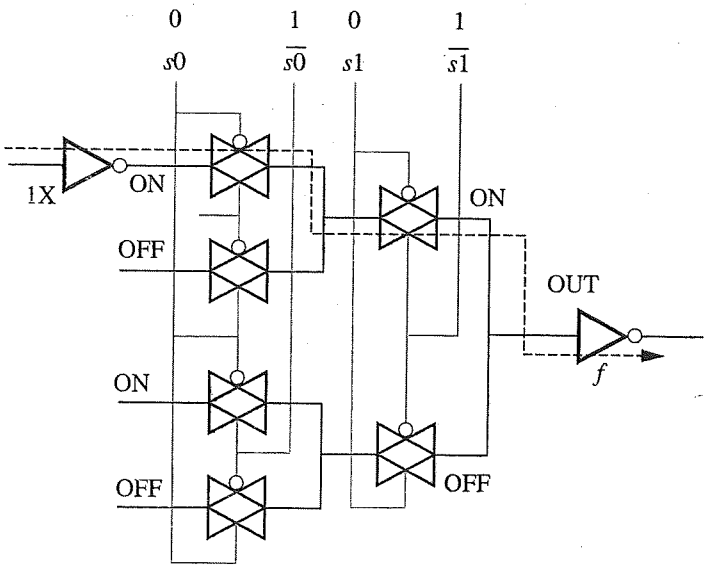
Solution:

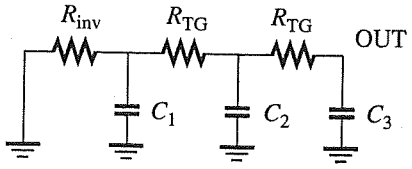
In order to compute the delays, we need to make use of the Elmore delay calculation. For a three-stage *RC* ladder, the delay has the form

$$t_{\text{Elmore}} = R_1C_1 + (R_1 + R_2)C_2 + (R_1 + R_2 + R_3)C_3$$

Using this formulation, we can derive the *RC* circuits and delays for the two cases as follows:

MUX1:





$t_1 = R_{\text{inv}}C_1 + (R_{\text{inv}} + R_{\text{TG}})C_2 + (R_{\text{inv}} + 2R_{\text{TG}})C_3$

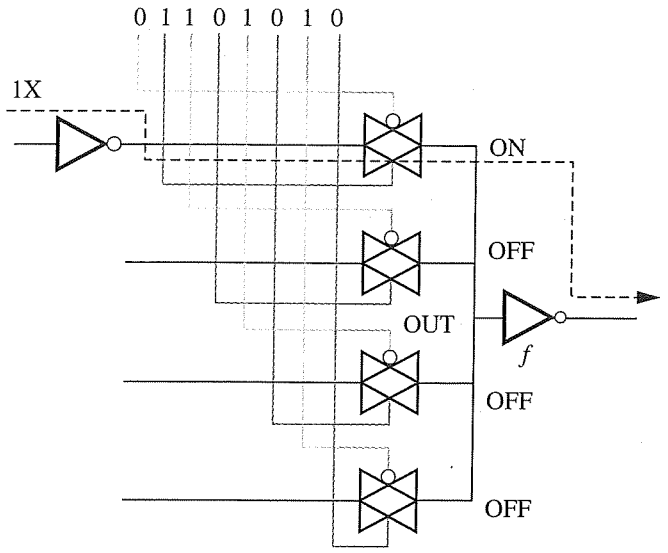
$C_1 = 3C_{eff}W + C_gW + 2C_{eff}W \qquad R_{\text{inv}} = R$

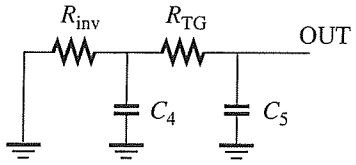
$C_2 = C_gW \times 2 + 2C_{eff}W \times 3 \qquad R_{\text{TG}} = R$

$C_3 = 2C_{eff}W \times 2 + C_gW + 3fC_gW$

$t_1 = R(29C_{eff} + 8C_g + 9fC_g)W$

MUX2:



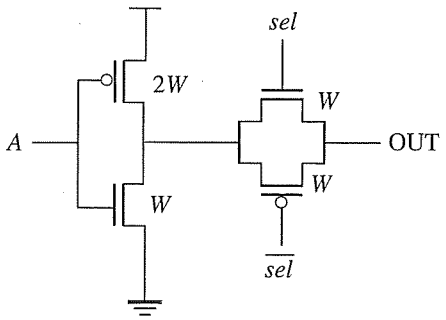


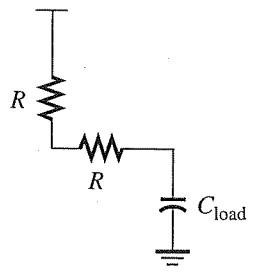
$$t_2 = R_{inv}C_4 + (R_{inv} + R_{TG})C_5$$
$$C_4 = 3C_{eff}W + 2C_{eff}W + C_gW \qquad R_{inv} = R$$
$$C_5 = 2C_{eff}W \times 4 + C_gW + 3fC_gW \qquad R_{TG} = R$$
$$t_2 = R(21C_{eff} + 3C_g + 6fC_g)W$$

Comparing the two cases, MUX1 is slower than MUX2. If we have too many transfer gates without any buffering, the delay increases very quickly. Also, we will eventually drive a large load, and transfer gates do not have the needed driving capability. However, MUX2 requires more routing resources than MUX1.

7.3.3 Logical Effort with CMOS Transmission Gates

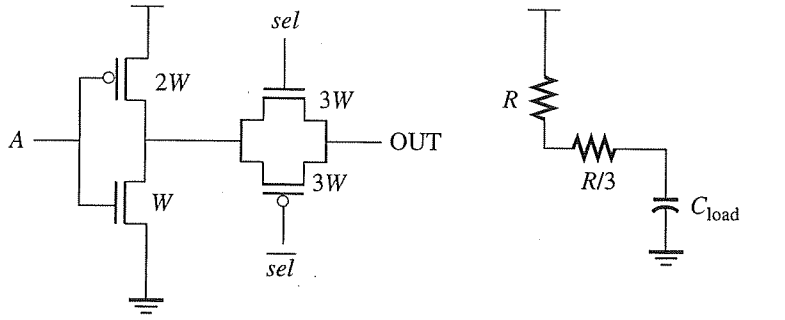
We now address the topic of logical effort (LE) and its application to CMOS transmission gates. Consider the circuit of Figure 7.19 with a minimum size transmission gate. This is another type of tristate driver which performs a similar function to the tristate inverter seen in Section 4.9 of Chapter 4. Here, we have three inputs: *A*, *sel*, and $\overline{sel}$. Since both *sel* and $\overline{sel}$ are related, we can compute the LE for the *sel*





$$\text{LE input } A = \frac{3W(2R)}{3WR} = 2 \qquad \text{LE input } sel = \frac{W(2R)}{3WR} = 2/3$$

Figure 7.19
Logical effort computation for transmission gates.



$$\text{LE input } A = \frac{3W(4R/3)}{3WR} = 4/3 \qquad \text{LE input } sel = \frac{3W(4R/3)}{3WR} = 4/3$$

Figure 7.20
LE computation with a 3X transmission gate.

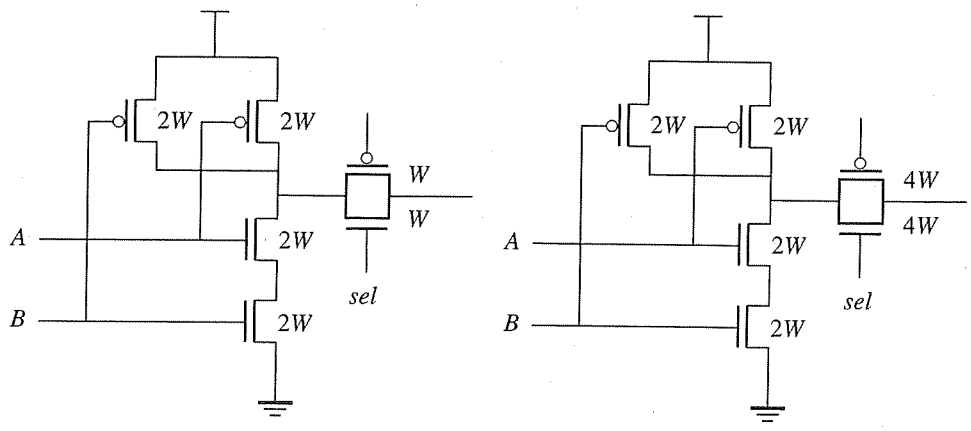
and A inputs. For input A, we apply the ratio of τ_{gate}/τ_{inv} as prescribed by the LE method. This involves the product of C_{in} with R_{eff} as described in Chapter 6. The expression for C_{in} is $3WC_g$ for the inverter. R_{eff} is $2R$, given that the inverter and TG each have a resistance of R. Therefore, $LE_A = 2$. For input *sel*, the input capacitance is WC_g and the total path resistance is $2R$ (including the inverter). Therefore, $LE_{sel} = 2/3$.

If we increase the size of the transmission gate to three times the minimum size, we see from Figure 7.20 that the LE of the A input decreases while the LE for the *sel* input increases. This makes sense intuitively since input A experiences a smaller output resistance without any change in its input capacitance, while input *sel* experiences an increase of three times its input capacitance but not a reduction of three times its output resistance.

Example 7.5 **Logical Effort for NAND Driving CMOS TG**

Problem:

For the two designs shown below, compute the logical effort for input A and input *sel*.



Solution:

$$\text{LE input } A = \frac{4W(2R)}{3W\,R} = 8/3 \qquad \text{LE input } A = \frac{4W(5R/4)}{3W\,R} = 5/3$$

$$\text{LE input } sel = \frac{W(2R)}{3W\,R} = 2/3 \qquad \text{LE input } sel = \frac{4W(5R/4)}{3W\,R} = 5/3$$

7.4 Dynamic D-Latches and D Flip-Flops

The operation and design considerations of latches and flip-flops were described in Chapter 5. The D flip-flop and D-latch are perhaps the most widely used circuits in CMOS design. In this section, we describe design improvements associated with these types of sequential elements when implemented with transfer gates. The major advantage in using transfer gates is that the circuits require many fewer transistors than the configurations shown in Chapter 5.

We begin with the simplest possible D-latch, shown in Figure 7.21a. Surprisingly, a single NMOS pass transistor functions as a D-latch. It is transparent since the value of D is propagated to Q while CLK is high. When the clock transitions from high to low, the last value is stored on the capacitor C_2. However, there are a number of obvious problems with this circuit based on material presented earlier in

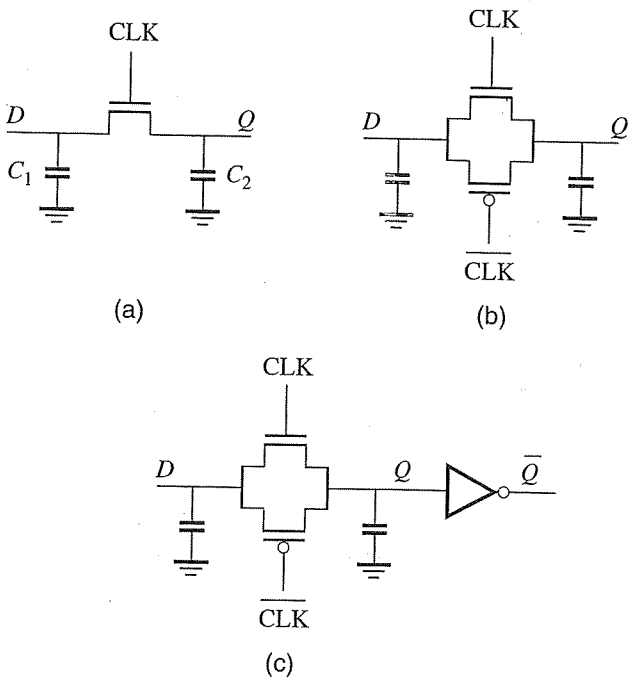


Figure 7.21
Progression of simple D-latches.

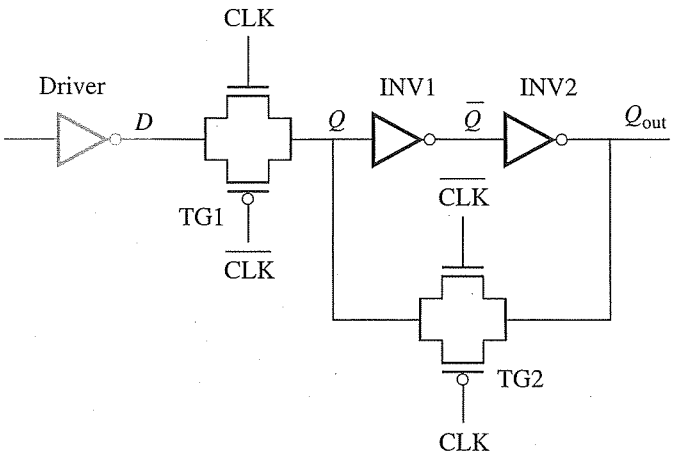


Figure 7.22
CMOS D-latch using transfer gates.

this chapter. First, the voltage at the output can only rise to $V_{DD} - V_T$. Second, there is clock feedthrough at Q when the clock goes low. The level of feedthrough is given by Equation (7.3). Third, there is no $\overline{Q}$ output available. Finally, and perhaps most importantly, the output is in a high-Z state after the clock goes low so it is susceptible to all of the charge loss mechanisms.

Most of these problems can easily be resolved. The circuit of Figure 7.21b is a CMOS transmission gate (TG) that resolves the V_T drop and clock feedthrough issues, as we know by now. The generation of $\overline{Q}$ is simply a matter of adding an inverter as shown in Figure 7.21c. While this "improved" D-latch will work as required, its reliability is still questionable due to the dynamic storage at node Q.

To solve this problem, we need a feedback loop to statically hold the value when the latch is *off*. This is illustrated in Figure 7.22. Here, we added another inverter to generate Q_{out} (since we cannot drive a load from the internal node Q), and introduced another CMOS TG to connect back to the initially generated Q. Note that the clock on the new TG is reversed with respect to the first TG. With this clocking scheme, TG1 is *on* when we are in *transparent* mode, while TG2 is *on* when we are in *hold* mode. When the latch is in *hold* mode, TG2 acts to recirculate Q back to the input of the first inverter, which takes advantage of the regenerative nature of the inverters. When the circuit is in transparent mode, TG2 is *off* and the value of D passes through to Q_{out} after a delay of TG1 and two inverters.

While this circuit removes the problem of dynamic storage, it introduces another small problem. When the CLK signal goes high, we note that there is a delay before $\overline{\text{CLK}}$ goes low. In that situation, the n-channel device of TG2 is still *on*. If D is different from the previously stored Q, then there will be a conflict at node Q for a short duration. TG1 is attempting to apply a new value at Q while TG2 is recirculating the old value through the loop. The problem can be resolved with proper sizing

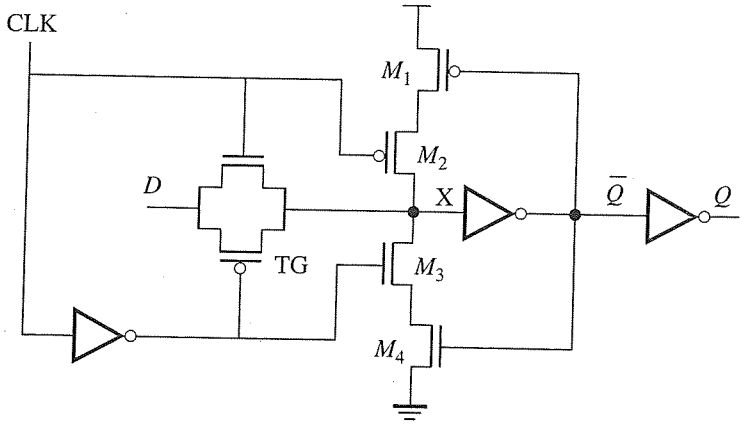


Figure 7.23
Typical D-latch implementation in CMOS.

of the INV2, TG2, TG1, and the gate that is driving *D*. The basic idea is to ensure that the forward path is "stronger" than the feedback path. Therefore, the driver and TG1 must be large enough such that the devices driving the forward path can swing the voltage at *Q* past the switching threshold V_S when competing with INV2 and TG2. If they are not sized properly, the old value will remain latched until TG2 turns *off*.

A more popular version of the D-latch is shown in Figure 7.23. This circuit employs the same principles as the one shown in Figure 7.22 but avoids any conflicts at internal nodes. The operation of this latch is as follows. When the CLK is high, the latch operates in transparent mode. Any changes at the input *D* are reflected at the output after a delay of TG + 2 INV. If the input data is high, then the value of $\overline{Q}$ is low. Therefore, M_1 is *on* and M_4 is *off*. When CLK goes low, then M_2 is turned *on* while M_3 is turned *off*. The internal node *X* is pulled high through M_1 and M_2. Note that there is no conflict situation in this circuit, and node *X* is not a dynamic node. A similar analysis can be done with a low input data. This time M_3 and M_4 would act to pull the internal node low after the clock goes low. Again, there are no conflicts in this configuration.

The circuit for a master-slave D flip-flop can be constructed from two D-latches, as shown in Figure 7.24. The flip-flop is simply a cascade of two D-latches. The connection of the clock signals to the master and slave are important to note. The positive edge of the clock shuts off the first latch and enables the second latch. The operation of the flip-flop is as follows: when the clock is low, the first latch is transparent, while the second latch is disabled. The data will flow up to TG2, but no further since TG1 is *on* and TG2 is *off*. When CLK goes high, TG1 shuts off and holds *D* at the internal node *Q*. Either the pull-up or pull-down path for the internal *Q* is enabled depending on the data value. Meanwhile, TG2 turns on and allows *Q* to flow to the outputs. Since *D* is only transmitted to the output on the rising edge of CLK, this is a positive edge-triggered flip-flop.

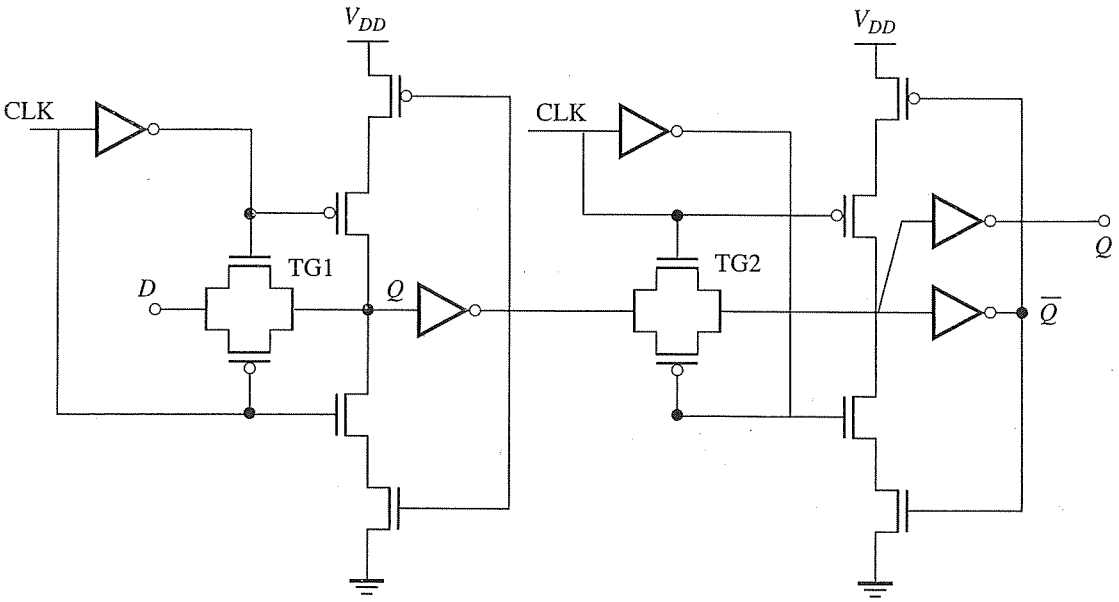


Figure 7.24
Positive edge-triggered D-type flip-flop.

7.5 Domino Logic

We have already seen the numerous difficulties that may arise with dynamic logic: charge sharing, feedthrough, charge leakage, single-event upsets, etc. These problems have limited the use of dynamic logic in industry. Furthermore, the CAD tool support needed to verify these types of designs are not readily available. Nevertheless, it is worthwhile to understand this approach for two reasons. First, static gates require numerous PMOS devices with relatively large sizes. Trying to implement a high fanin gate (i.e., a gate with more than four inputs) is difficult because either the *N*-complex or *P*-complex will require a large series stack. Pseudo-NMOS gates help to reduce the high fanin problem at the expense of static power when the output is low. The dynamic approach allows us to design circuits that are faster than static gates, with lower power than pseudo-NMOS, so they are worth exploring. We limit the treatment here to the most prominent dynamic configurations used in industry.

The structure of a dynamic logic gate can be derived from a static logic gate, as shown in the diagram of Figure 7.25. The conversion from static CMOS to pseudo-NMOS is straightforward, as shown in the first and second diagrams. We know that the second gate suffers from excessive power dissipation when the output is low. We need a way of shutting off the PMOS device when the output is low so as to reduce the power. What if we turned on the PMOS device to precharge the output to a high value, and then shut off the PMOS device during the period when the output would go low? For this purpose, we could introduce a new clock signal, $\Phi_{\text{precharge}}$, to turn the PMOS device *on* and *off* as needed. However, if we tried to charge up the output using the clocked PMOS device while a path to Gnd existed through the NMOS devices, we would have contention between the pull-up and pull-down paths.

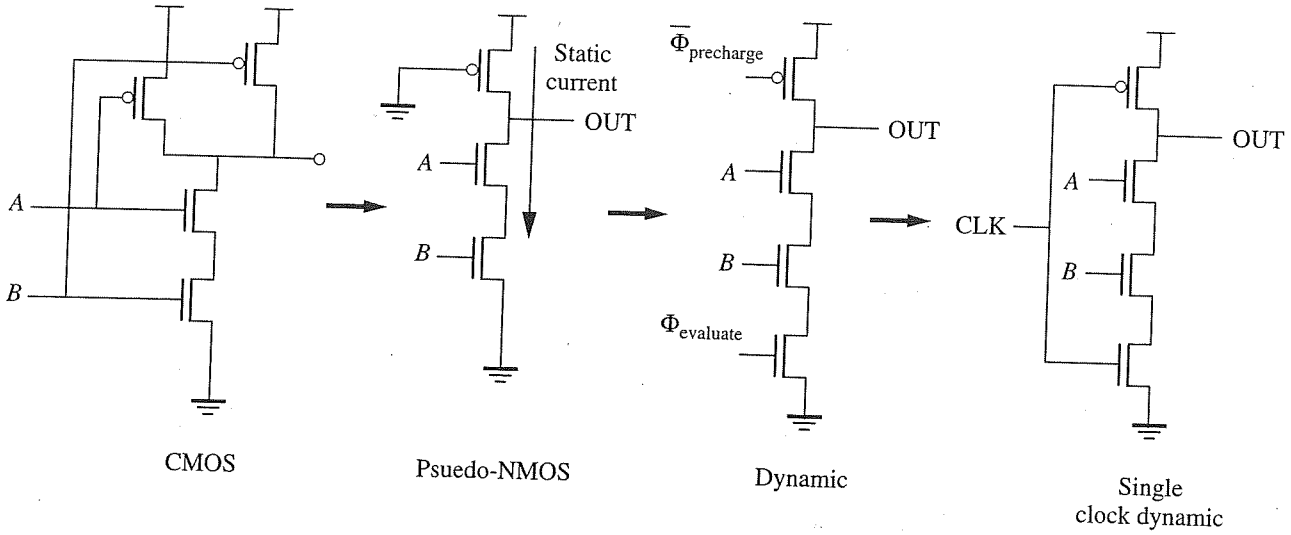


Figure 7.25
Evolution from static gate to dynamic gate.

Therefore, we must add another NMOS transistor, called a *foot* transistor or *evaluation* transistor, in the pull-down path. This transistor is turned on only after the PMOS load is shut off using another clock signal, $\Phi_{evaluate}$.

The configuration described above is depicted in the third circuit of Figure 7.25 where $\Phi_{precharge}$ is used to turn on the *p*-channel device and pull the output high (precharge phase), while $\Phi_{evaluate}$ is used to turn on the *n*-channel device to potentially pull the output node low (evaluation phase). The precharge signal is active low since it is connected to the PMOS device. The evaluate signal is active high since it turns on the NMOS device. Since the precharge and evaluate clocks are in phase, we can simply use one signal called CLK to drive the precharge and foot transistors. This is shown in the fourth circuit for the dynamic 2-input NAND gate.

The operation of the 2-input dynamic NAND gate is as follows. Initially, the precharged signal goes low and the output is precharged high. It is as if we are postulating a "1" at the output of all nodes during this phase. The precharge phase is intended to be a small portion of the clock cycle. Then, the CLK goes high and we enter the evaluation phase when the dynamic gates either remain high or are pulled low. For this NAND gate, if both *A* and *B* are high, then a path exists to Gnd. Otherwise, the output remains high and we preserve the value on the capacitance at the output node. This charge storage at the output node makes it a dynamic gate.

Dynamic Gate Implementations **Example 7.6**

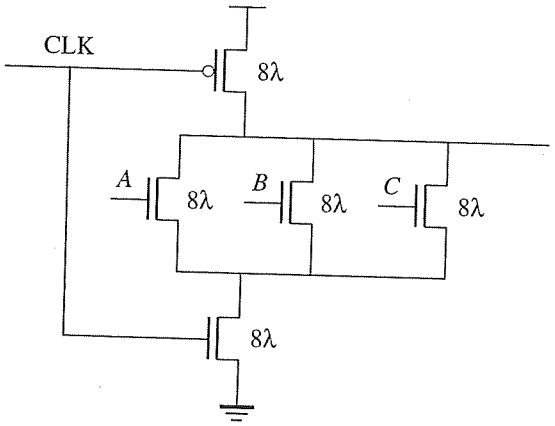
Problem:

Implement a 3-input NOR gate in dynamic logic and explain its operation. Size the transistors to deliver the same delays as a conventional CMOS inverter (PMOS 8λ:2λ, NMOS 4λ:2λ).

Solution:

The dynamic 3-input NOR gate requires three parallel NMOS transistors, plus two transistors connected to the CLK input: one PMOS pull-up and one NMOS foot transistor. When CLK is low, the precharge transistor pulls the output to V_{DD}. When the clock goes high, the evaluate transistor turns on. If *A* or *B* or *C* is high, the output will be discharged to Gnd. If all three inputs are low, the output will remain high.

The pull-up requires a width of 8λ to deliver the same rise delay as the CMOS inverter. This is somewhat irrelevant since all gates are precharged simultaneously. The pull-down tree should also be sized with 8λ transistors to produce the equivalent 4λ device to deliver the same fall delay as the CMOS inverter. The final sizes are shown below:



In reality, the CLK signal arrives first so *A*, *B*, and *C* can be considered as late arriving signals. In that case, the transistors can be reduced to half the size to deliver the same delay as the inverter.

Most logic functions can be implemented as dynamic circuits using the structure shown in Figure 7.26. The desired function is implemented in the NMOS complex. Transistors M_P and M_N are the precharge and evaluate transistors, respectively. The foot transistor is the only extra device relative to pseudo-NMOS gates. Note that all dynamic gates require a clock signal for proper operation. Furthermore, the clock must be routed to all dynamic gates, as opposed to static gates which do not require a clock. This places an additional burden on the physical layout tools.

There is a notable complication when one dynamic gate feeds another, as shown in Figure 7.27a. Since all output nodes are precharged high, all inputs to subsequent gates are high, implying that there will be an active path to Gnd from each output node as soon as the foot transistor is turned on. Once an output node has been discharged, it cannot go high until the next precharge phase. Therefore, a direct connection is not suitable.

A simple solution exists for this problem: add a static inverter to the output of all dynamic gates. Then define a "stage" as the dynamic gate plus the inverter. The

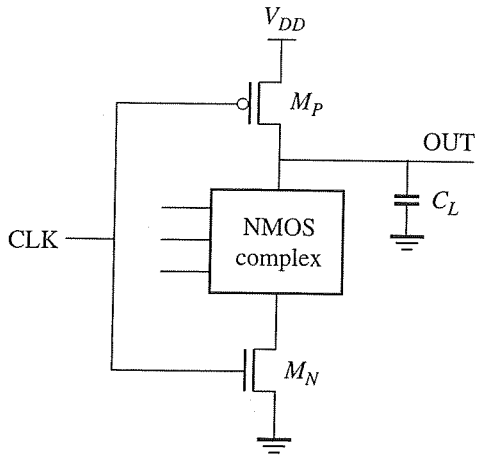
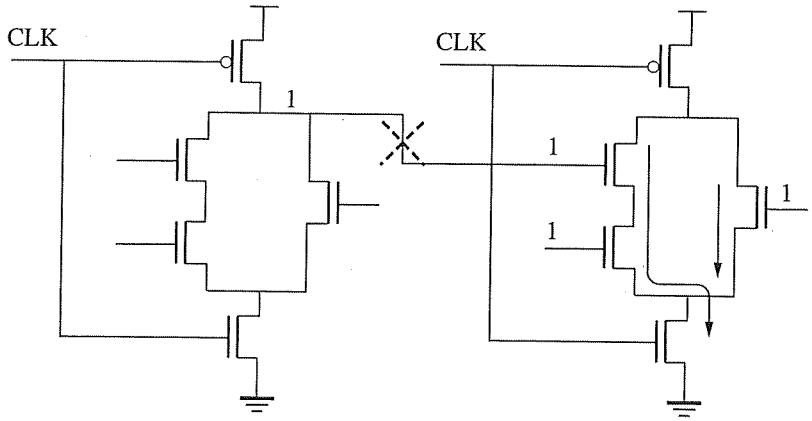
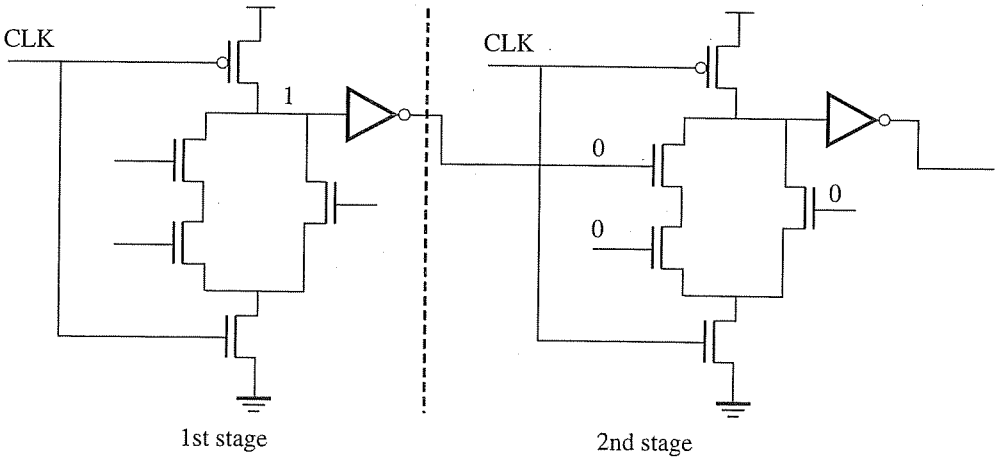


Figure 7.26
General structure of a dynamic gate.



(a) Direct connection not possible



(b) Insert inverter between dynamic gates

Figure 7.27
Connecting dynamic gates.

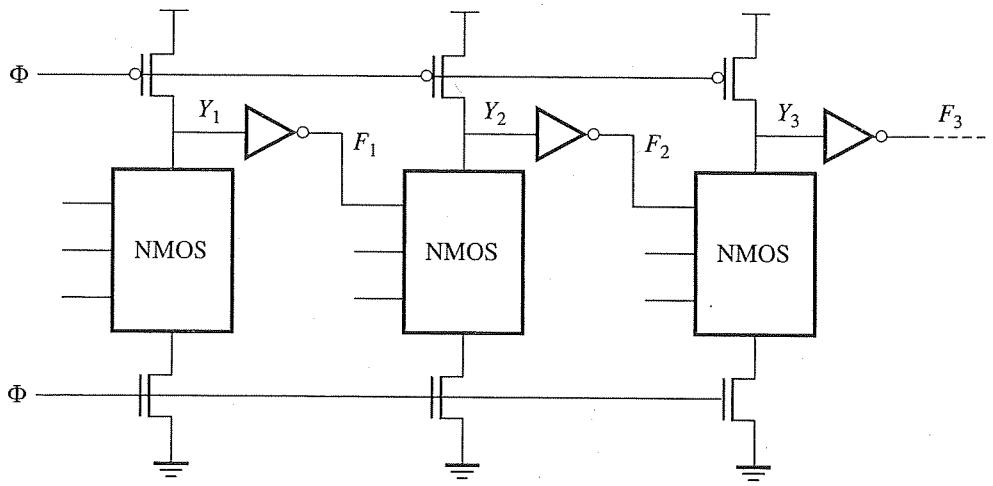


Figure 7.28
Domino cascaded gates.

output of each stage is now low during precharge. Therefore, all NMOS transistors are off during precharge and can only be turned on during the evaluate phase as shown in Figure 7.27b. The combination of a dynamic stage and an inverter is referred to as a *domino stage,* and circuits using this form of dynamic logic are referred to as *domino logic* circuits.

A cascade of three domino logic stages is shown in Figure 7.28. All three stages are precharged high when Φ is low. The clock, Φ, need not be low for very long since all stages are precharged simultaneously. In fact, the clock should only be low until all inverter outputs are low. When Φ goes high, we enter the evaluation phase where the internal nodes (labeled Y_1, Y_2, Y_3) will fall like dominos in order from left to right, assuming there is a path to the Gnd through their respective *n*-complexes. This is why the name *domino* is used to describe logic circuits implemented in this fashion.

Meanwhile, the true outputs, F_1, F_2, and F_3, would rise to V_{DD}, depending on the logic in the *n*-complexes. The clock must remain high long enough for logic to propagate through the entire chain. Therefore, an asymmetric clock with a relatively high duty cycle (percentage of time that the clock is high) is used for domino logic. The propagation delay is the sum of each dynamic block and inverter. However, we care most about the falling edge of the dynamic block and the rising edge of the inverter output. When we optimize the speed of this type of logic, we try to speed up these transitions because they are the only ones that control the cycle time.

In fact, we should design a domino stage with a stronger pull-down in the dynamic gate and a stronger pull-up in the static inverter. For the dynamic gate, we should increase the sizes of the NMOS devices in the *n*-complex. Since we are only interested in the rising output of the inverter, we could increase the size of its PMOS device. This type of inverter is a *skewed* inverter since its switching threshold is skewed relative to the conventional inverter. As a side-effect of the sizing operations, the V_S is higher for the static inverter. Furthermore, there is no pull-up fighting the pull-down in the first gate so its V_S is actually lower than expected (roughly equal

to the V_{TN} of the NMOS device). Therefore, a domino stage will actually switch earlier than a regular gate.

There is also a power savings in domino logic. Only those gates that discharge to Gnd will dissipate power. There will be no power dissipation due to crowbar current since a direct path between V_{DD} and Gnd is not allowed in this type of logic. In addition, glitches can be effectively removed since all inputs switch from low to high.

One disadvantage of domino logic is that it can only be used to create noninverting functions. This may not seem to be a problem at first glance, but one quickly realizes that *an inverter cannot be implemented* in domino logic. This is not a serious problem since we can push the inversions around the circuit as required until they appear at a primary input or output. We can also generate the true and complement values if necessary using two precharged gates. In the worst case, we will require twice as many gates but the overall speed will still be faster.

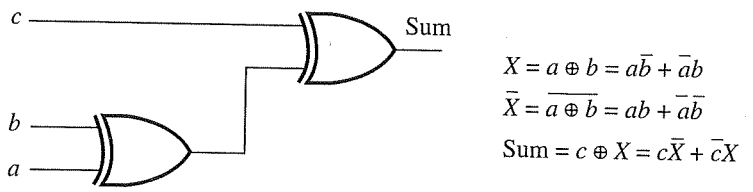
Adder Function in Domino Logic **Example 7.7**

Problem:

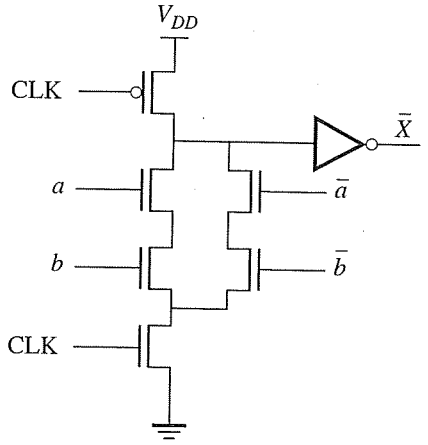
Implement the function $sum = a \oplus b \oplus c$ in domino logic. Assume that the literals $a, \bar{a}, b, \bar{b}$ are available as stable inputs to the gates.

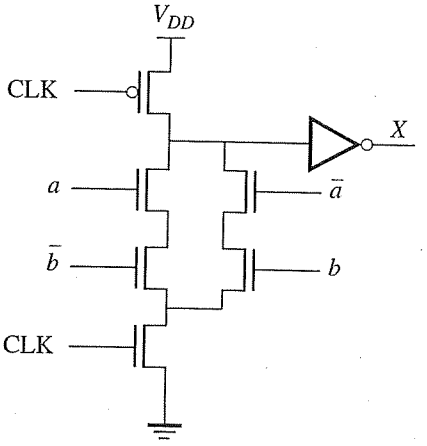
Solution:

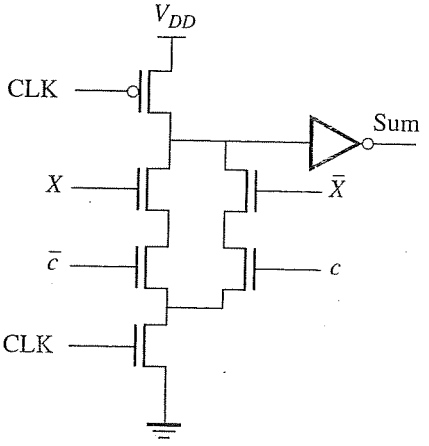
We need to create the XOR and XNOR of a and b using domino gates and then apply them to a third XOR gate. Note that we need the true and complement of X to implement the function.

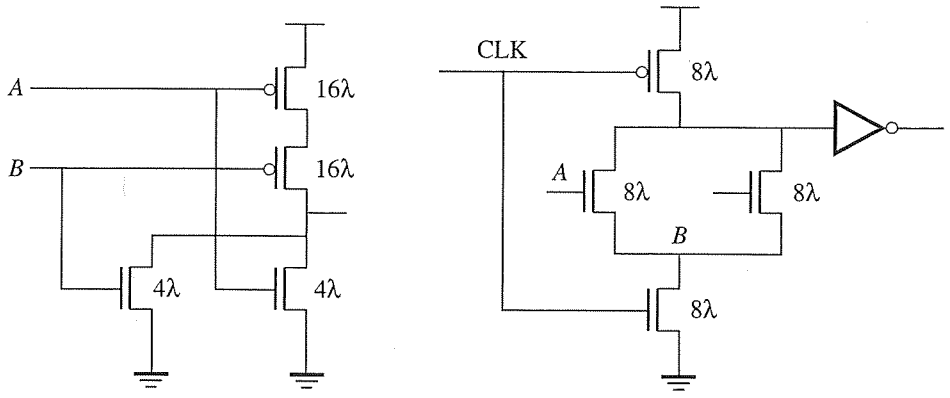


$$X = a \oplus b = a\bar{b} + \bar{a}b$$
$$\bar{X} = \overline{a \oplus b} = ab + \bar{a}\bar{b}$$
$$\text{Sum} = c \oplus X = c\bar{X} + \bar{c}X$$









(a) Static NOR gate (b) Dynamic NOR gate

Figure 7.29
Comparison of static and dynamic logical effort.

7.5.1 Logical Effort for Domino Gates

One question to pose at this stage is whether or not domino logic is actually faster than the static CMOS gates, and by how much. One way to evaluate its timing performance is to use logical effort. For comparison, consider a static NOR gate and a dynamic NOR gate as shown in Figure 7.29. The pull-down strengths of both gates have been made equal and we assume that they are driving the same output load to conduct a fair comparison. The length of all devices is 2λ.

For the static gate, we already determined its logical effort to be 5/3. This can be derived by comparing the input capacitance of the static NOR to a corresponding inverter. The inverter size of 8λ for the pull-up and 4λ for the pull-down produces a ratio of

$$\mathrm{LE_{NOR}} = \frac{(\text{NOR input cap.})}{(\text{Inverter input cap.})} = \frac{16\lambda + 4\lambda}{8\lambda + 4\lambda} = 5/3$$

For the domino circuit, the input capacitance is simply 8λ since it is only connected to NMOS devices. Then,

$$\mathrm{LE_{dyn\_NOR}} = \frac{(\text{Dynamic input cap.})}{(\text{Inverter input cap.})} = \frac{8\lambda}{12\lambda} = 2/3$$

Since the extra inverter in domino logic has an LE = 1, the average LE for each of the two gates is the geometric mean of the product of their logical efforts (i.e., $\mathrm{LE_{avg}} = \sqrt{(2/3)\,(1)} \approx 0.8$). Therefore, the domino stage is better in terms of its overall drive capability and input capacitive loading.

The results above may seem counterintuitive at first since there are two stages in the domino circuit and only one in the static NOR gate. However, the fact that only one NMOS device is driven in the domino case gives it a decided advantage in terms of input capacitance. This advantage translates into a faster overall circuit.

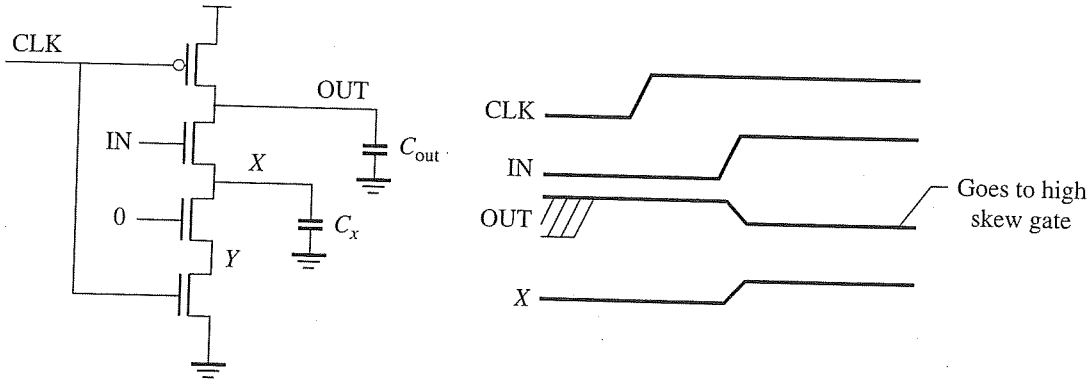


Figure 7.30
Charge-sharing example in domino logic.

Furthermore, the pull-down transistors in the domino case do not fight with the pull-up device as in the case of the static NOR gate (albeit, only for a brief period during switching). This makes the switching threshold V_S smaller for domino inputs than for static gates.

7.5.2 Limitations of Domino Logic

While the previous sections highlighted the merits of dynamic logic, there are notable disadvantages that have prevented its widespread use in industry. The main problem is the potential for logic upset due to charge loss on a capacitor, and that is not acceptable to most designers. Charge may be lost via charge sharing, noise injection due to capacitive coupling, charge leakage, or α-particle hits. Once lost, it cannot be recovered and the circuit ceases to function correctly. We address each of these issues below.

Charge sharing was described earlier in this chapter. We now apply it to the domino stage shown in Figure 7.30. Initially, CLK is low so the circuit is in the precharge phase. The *OUT* signal is unknown until the precharge is complete. We assume that signal X is low initially. When CLK goes high, we enter the evaluation phase. At some point, *IN* goes high and turns on one of the transistors connected to the output, while the other transistor in the stack remains off. As a result, we have charge sharing between the output node and the internal node X. The degree of charge sharing depends on the two capacitances C_{out} and C_x:

$$V^* = \frac{C_{out}}{C_x + C_{out}} V_{DD}$$

If they are equal, then the output value will be reduced to half its original value (i.e., 1/2 V_{DD}). Since this value is being fed to an inverter, the implications of this degree of charge sharing is that the inverter may flip its value and produce an erroneous result. Furthermore, the fact that the inverter is skewed for timing reasons means that a smaller drop in the output voltage will cause the inverter to switch. There is

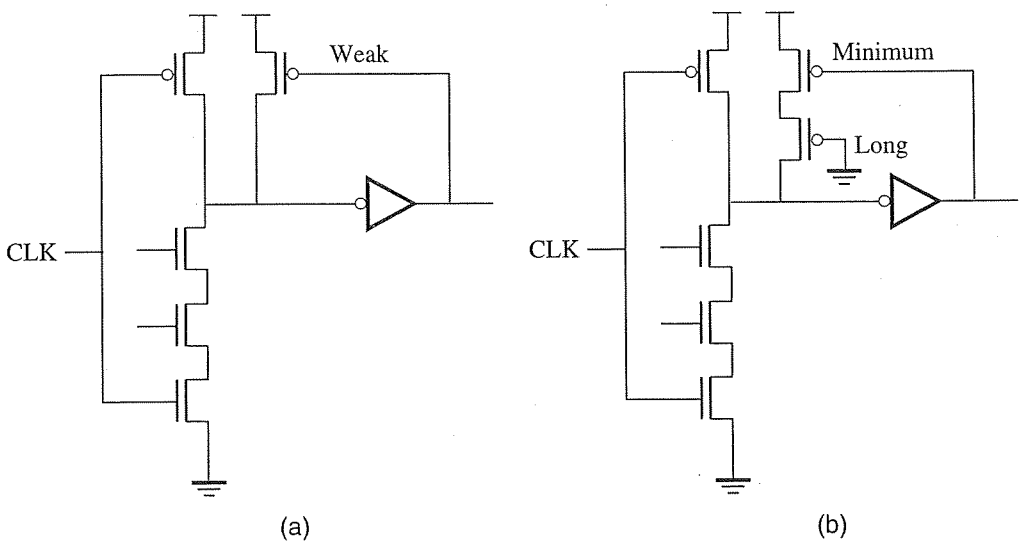


Figure 7.31
Minimizing the effects of charge sharing using keepers.

obviously a design tradeoff between timing and noise tolerance when sizing the inverter.

There are a number of ways to minimize charge sharing in these types of circuits. One way is to increase the output capacitance so that the final output will not vary significantly from its initial value. The input capacitance of the static inverter helps in this regard. Another approach is to precharge the internal node X to V_{DD} during the precharge phase with an additional transistor. This will require additional power and area. It will also increase the delay since there will be additional charge to remove from the internal nodes.

A third approach is to introduce "keepers" or "baby-sitters" to hold the output value high in the presence of charge sharing. In Figure 7.31a, a weak PMOS device is placed in a feedback path from the output to the internal node. After precharge, the PMOS device is on and it continues to pull up the node even after the precharge is complete. If there is any charge sharing, the output is pulled back up to V_{DD} by this device. On the other hand, if there is a path to Gnd from the output of the dynamic stage, the NMOS path must overpower the PMOS device so that the output actually switches. For this reason, we need to make the PMOS device very weak. That is, the length L of the device is made large which makes the W/L small.

The problem with this requirement is that it places a larger drive requirement on the output inverter. To circumvent this issue, two transistors can be used in the feedback path: one transistor is driven by the inverter output but it is a minimum size device so it does not place a significant load on the inverter. The second one is always *on* but it is a very long (and hence weak) device. The effective pull-up strength is controlled by the long device, but the capacitance seen by the inverter output is due to the small device. There is additional power dissipation, some ratioing that is required and some additional delay. Clearly, as you begin to make the

dynamic circuit more tolerant to noise, it begins to take on the characteristics of a static device.

A second issue in dynamic circuits is leakage. Both subthreshold current and reverse-bias diode leakage act to remove stored charge from any *high-impedance* nodes. However, every output node is precharged on every cycle. Perhaps a more serious issue is that of α-particle hits on dynamic nodes. These are cosmic particles that may enter the substrate and discharge capacitive nodes. Depending on the size of the capacitance, and hence the total charge on a dynamic node, it could reduce the value to a point where the stored information is lost. One way to overcome this issue is to increase the size of the capacitances, but this increases the area and power of the circuits. The keeper circuit is also useful in mitigating the effects of leakage and α-particles.

A third area of concern is noise injection from the clock or capacitive coupling to neighboring nodes. Clock feedthrough was described earlier in this chapter and this effect can be observed at the dynamic output node through the C_{GD} capacitance of the PMOS precharge device. The example below illustrates that this effect does not pose a problem. The issue of capacitive coupling to other nodes is described in more detail in the chapter on interconnect. The design approach to reducing this problem is to minimize the coupling between any dynamic node and its neighbor using either spacing or shielding techniques that are described in Chapter 10.

Charge Sharing and Feedthrough in Domino Logic **Example 7.8**

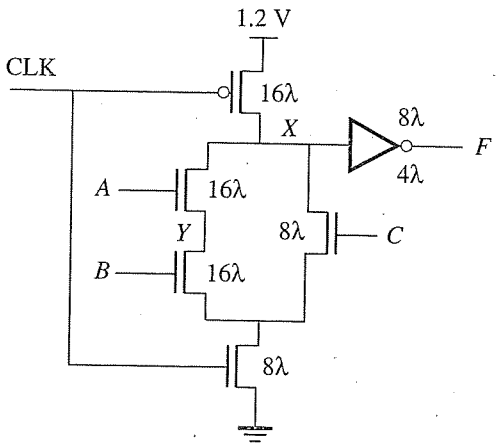
Problem:

For the domino function below, assume 0.13 μm technology parameters and answer the following questions:

(a) What function does the gate perform at the output *F*?

(b) How much clock feedthrough do we observe at the internal node *X*? Is this a potential problem?

(c) What is the worst-case charge sharing voltage that we observe at node *X*?



Solution:

(a) The function at the output is $F = AB + C$ (a noninverting function).

(b) Clock feedthrough is due to the clock going from low to high. The PMOS device is in the linear region after precharge. The feedthrough capacitance is

$$C_f = C_{GD} = \frac{C_g W_p}{2} = \frac{(2\ \text{fF}/\mu\text{m})(16)(0.05\ \mu\text{m})}{2} = 0.8\ \text{fF}$$

The grounded capacitance at node X, assuming no source/drain sharing, is

$$\begin{aligned} C_{\text{gnd}} &= C_{eff}(16\lambda + 16\lambda + 8\lambda) + C_g(8\lambda + 4\lambda) \\ &= (1\ \text{fF}/\mu\text{m})(40)(0.05\ \mu\text{m}) + (2\ \text{fF}/\mu\text{m})(12)(0.05\ \mu\text{m}) \\ &= 5\ \text{fF} \end{aligned}$$

The feedthrough voltage is

$$\Delta V_x = \frac{C_f \Delta V_{\text{clk}}}{C_f + C_{\text{gnd}}} = \frac{0.8\ \text{fF}(1.2\ \text{V})}{0.8\ \text{fF} + 5\ \text{fF}} = 0.17\ \text{V}$$

Therefore the output voltage due to feedthrough is $1.2 + 0.17\ \text{V} = 1.37\ \text{V}$. Since the value is above V_{DD}, it is not a problem.

(c) The worst-case charge sharing assumes that the output is sitting at 1.2 V, $A = 1, B = C = 0$. We have already determined that $C_X = 5$ fF. The capacitance at the intermediate node between transistors A and B is

$$C_Y = (1\ \text{fF}/\mu\text{m})(16\lambda) = 16(0.05) = 0.8\ \text{fF}$$

Therefore, the worst-case charge sharing is given by

$$V^* = \frac{C_X(1.2\ \text{V})}{C_X + C_Y} = \frac{5(1.2\ \text{V})}{0.8 + 5} = 1.0\ \text{V}$$

This level is not possible at node Y. Set $V_Y = V_{DD} - V_{TN} = 0.73$ V. The rest of the charge remains at node X. The remaining charge is (5 fF)(1.2 V) − 0.8 fF (0.73 V) = 5.42 fC. This translates to a voltage of 1.08 V at node X.

7.5.3 Dual-Rail (Differential) Domino Logic

A popular form of domino logic is called *dual-rail* or *differential* domino logic. It is also known as differential cascode voltage switch (DCVS) logic. Since standard domino can only implement noninverting logic functions, the dual-rail approach has been adopted by many microprocessor designers to overcome this limitation. The basic structure is shown in Figure 7.32. Rather than having only one output node, a complementary pair of logic outputs are generated. There are two precharge transistors that force the outputs low when CLK is low. When CLK goes high, either OUT or $\overline{OUT}$ will go high, depending on the logic blocks, $\overline{F}$ and F, respectively. Of course, both outputs cannot go high during the evaluate phase.

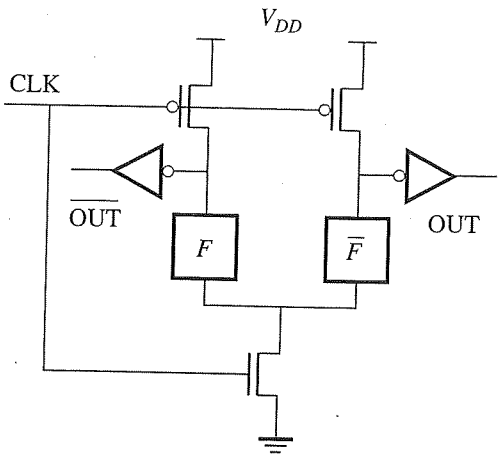


Figure 7.32
Structure of dual-rail domino logic.

An example of a dual-rail AND/NAND gate is shown in Figure 7.33. One side of the gate implements the AND function while the opposite side implements the NAND function.

This configuration has a number of problems similar to the standard domino gate. The internal nodes are susceptible to charge sharing. Therefore, a pair of *keeper* devices are needed to maintain high levels if the clock is high for a long period of time. The keeper devices are shown in Figure 7.34. It is clear that the number of devices needed to implement the gate has increased significantly from the modest levels that we expected when first considering dynamic logic. Furthermore, the power dissipation of these gates is much higher than the conventional CMOS gate or the standard domino gate since one side will always be precharged high and then

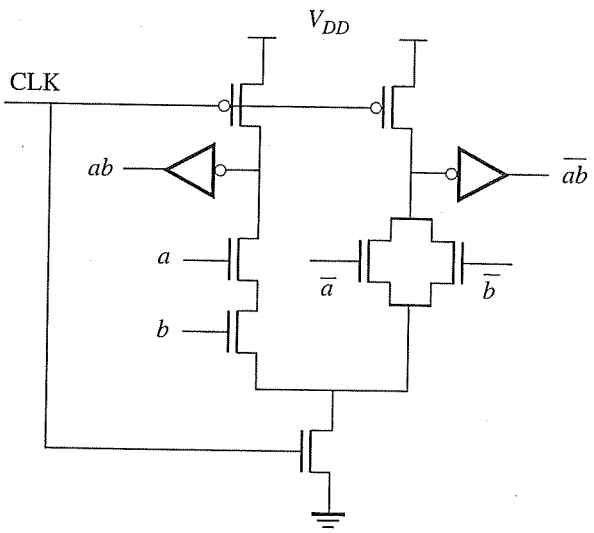


Figure 7.33
Dual-rail domino AND/NAND function.

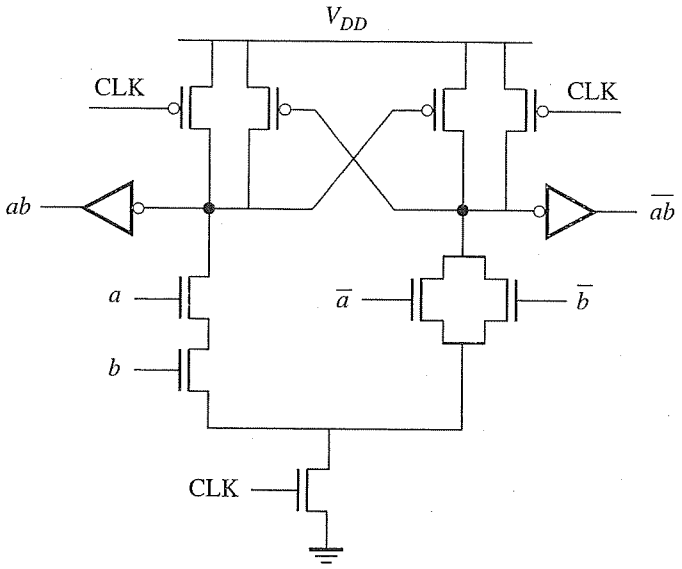


Figure 7.34
Dual-rail domino with keeper devices.

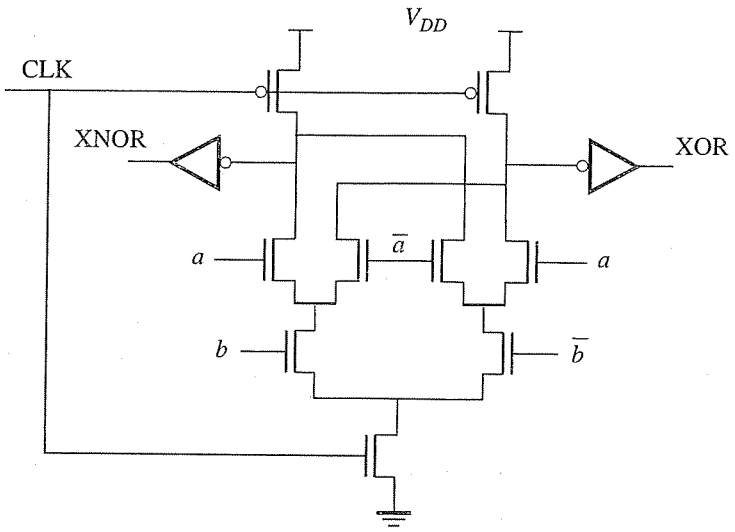
go low. This is a significant price to pay for the extra output. However, we should note that not every logic gate requires a complement function. Only those functions that explicitly require a complementary term would require the use of dual-rail gates. The rest can be implemented as standard domino gates.

Example 7.9 **Dual-Rail XOR/XNOR Domino Gate**

Problem:

Design a dual-rail XOR/XNOR domino gate and share as many transistors as possible between the true and complement logic blocks.

Solution:



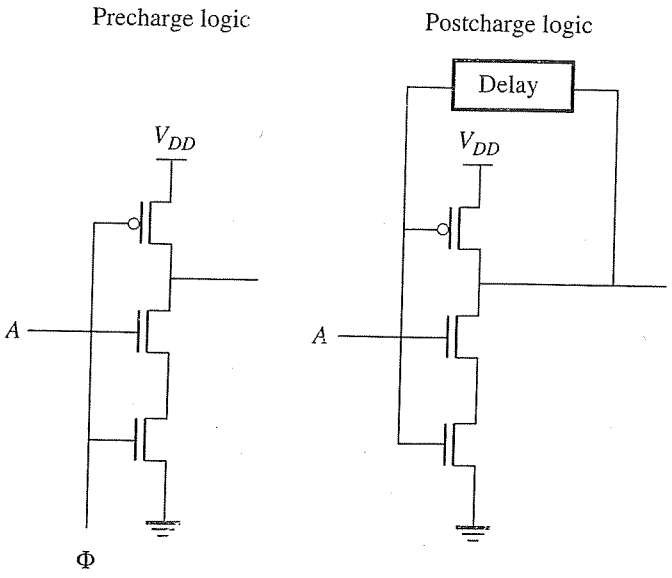


Figure 7.35
Precharge and postcharge circuits.

7.5.4 Self-Resetting Circuits

The routing of the clock to domino gates presents a problem to CAD tools and introduces issues of delay and skew into the circuit design process. There are situations that permit the use of circuits that can automatically precharge themselves (i.e., reset themselves) after a prescribed delay. These are called postcharge or self-resetting logic circuits. They find applications where a small percentage of gates switch in each cycle, such as memory decoders (described in Chapter 10). Here we will examine the basic operation of these circuits.

Figure 7.35 shows the standard precharge configuration and the postcharge configuration. In the domino case, the clock is used to operate the circuit. In the self-resetting case, the output is fed back to the precharge control input and, after a specified time delay, the pull-up is reactivated. The delay line is implemented as a series of inverters. The signals that propagate through these circuits are pulses. The width of the pulses must be controlled carefully or else there may be contention between NMOS and PMOS devices, or even worse, oscillations may occur. For example, if A is high for more than twice the delay around the loop, the circuit will oscillate. Therefore, special care must be taken to ensure correct timing.

7.6 Summary

Capacitive feedthrough and bootstrapping equation:

$$\Delta V_2 = \frac{C_f \Delta V_1}{C_f + C_{\text{gnd}}}$$

Charge-sharing equation:

$$V^* = \frac{(C_1V_1 + C_2V_2)}{C_1 + C_2}$$

Large-signal resistance for CMOS transmission gate:

$$R_{on} \approx \frac{V_{DS}}{I_{DS}}$$

$$R_{TG} = R_N // R_P \approx R_{eqn}\left(\frac{L}{W}\right)$$

REFERENCES

1. I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits,* Morgan Kaufman Publishers, San Francisco, CA, 1999.
2. J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Designer Perspective,* Second Edition, Prentice-Hall, Upper Saddle River, NJ, 2003.
3. S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits, Analysis and Design,* Third Edition, McGraw-Hill, New York, NY, 2003.
4. H. Veendrick, *Deep-Submicron CMOS ICs,* Second Edition, Kluwer Academic Publishers, Boston, MA, 2000.
5. J. P. Uyemura, *CMOS Logic Circuit Design,* Kluwer Academic Publishers, Boston, MA, 1999.

PROBLEMS

P7.1. Compute all the voltages at all the nodes for the pass transistor circuit of Figure P7.1 including the body effect. Use 0.13 μm technology parameters. Assume that all nodes are initially at 0 V.

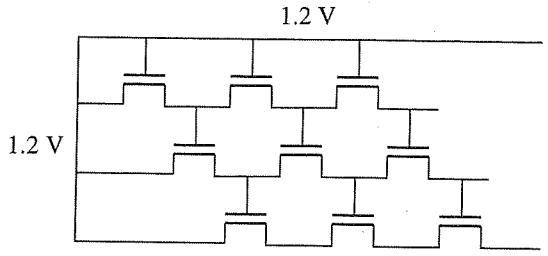


Figure P7.1

P7.2. Provide a truth table that describes the functionality of each circuit in Figure P7.2. (Note: In some cases, the output of the circuit may be high-*Z*.) The symbol A' refers to the complement of A.

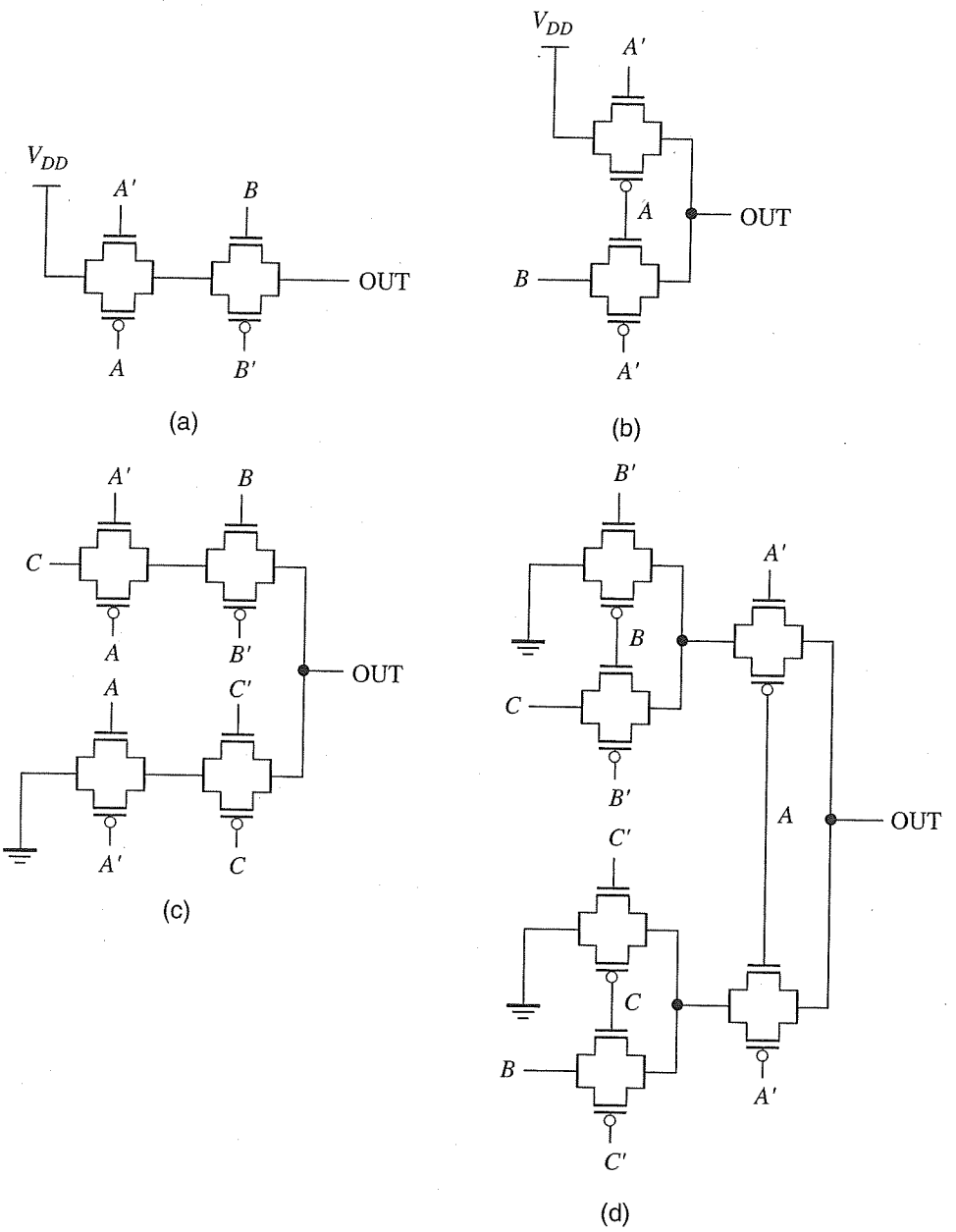


Figure P7.2

P7.3. (a) For the dynamic D-latch shown in Figure P7.3a, estimate the output voltages for Q and $\overline{Q}$ given that the input is at V_{DD} when CLK goes high (V_{DD}). Assume that $W = 200$ nm and $V_{DD} = 1.8$ V. Then, recompute the values when the CLK goes low. Include the effects of clock feedthrough.

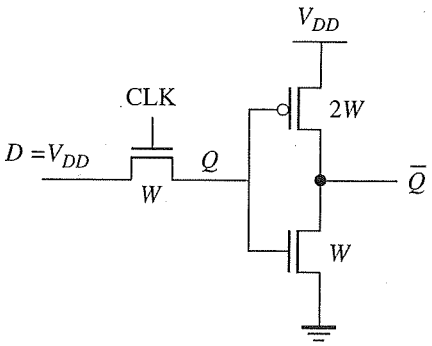


Figure P7.3a

(b) Repeat part (a) for Figure P7.3b. Then, compute the worst-case delay through the D-latch when the clock input goes high. Include all capacitance. Assume $W = 200$ nm.

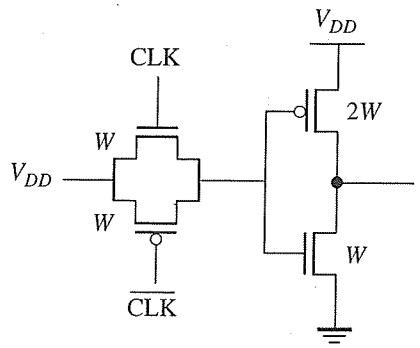


Figure P7.3b

P7.4. Using transmission gates, design a circuit whose output is

(a) Out $= A + BC$

(b) Out $= AB + BC + \overline{C}$

(c) Out $= \overline{(A + B + C)} + \overline{A}B$

(d) Out $= \overline{\left(\overline{(A + B + \overline{C})} + \overline{A}\,\overline{B}\right)}$

Then remove the transistors and switches that are not necessary.

P7.5. What is the output of each of the circuits in Figure P7.5?

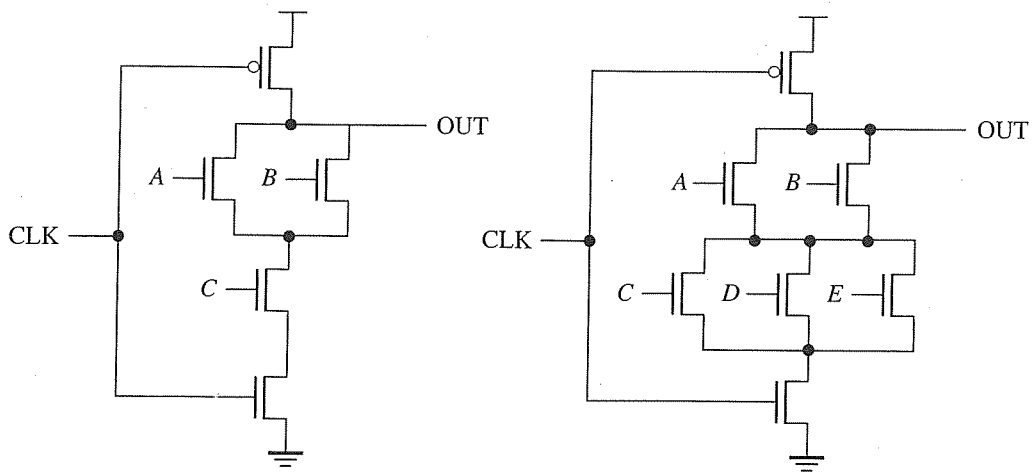


Figure P7.5 (a) (b)

P7.6. Design a domino circuit whose output is

(a) Out = $\overline{A} + \overline{B}C$

(b) Out = $A\overline{B} + BC + \overline{C}$

(c) Out = $\overline{(\overline{A} + \overline{B} + C)} + A\overline{B}$

(d) Out = $\overline{\left(\overline{(A + \overline{B} + C)} + \overline{A}B\right)}$

P7.7. What is the LE (logic effort) of each input signal of the circuits of Figure P7.7? (All transistors are minimum size, $L = 2\lambda$ and $W = 2\lambda$, unless specified.)

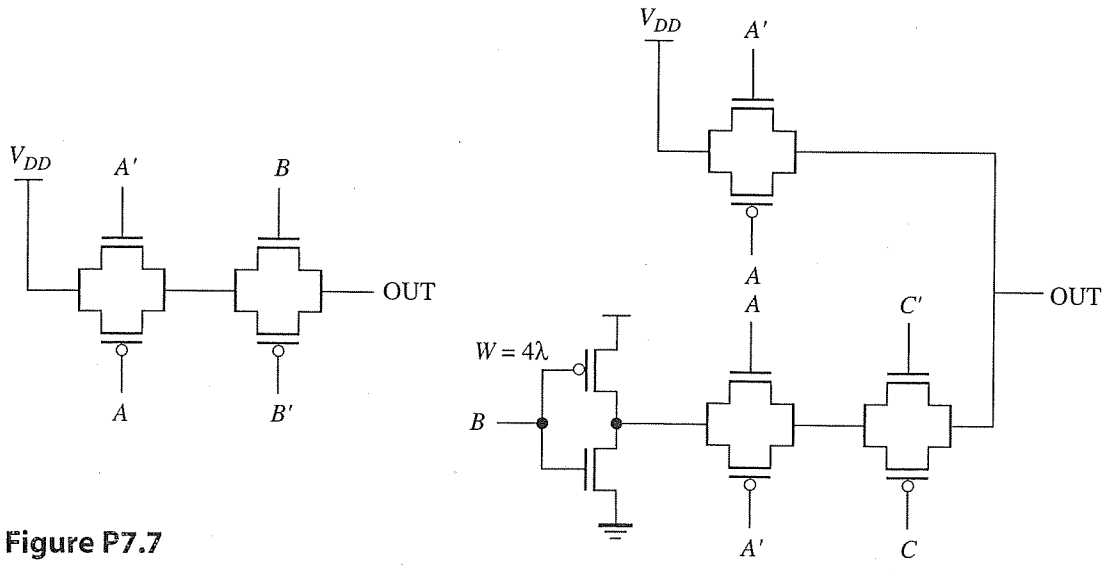


Figure P7.7

P7.8. For the circuit shown in Figure P7.8, assume that the transmission gates are all 4λ:2λ and that the inverters driving the transmission gates have PMOS transistors that are 8λ:2λ, and NMOS transistors that are 4λ:2λ, where $\lambda = 0.1\ \mu$m. The output inverter is to drive a 50 fF load. The output inverter is f times larger than the input inverters.

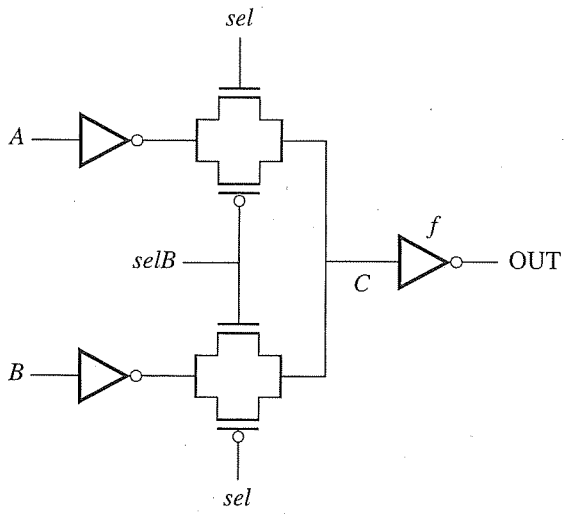


Figure P7.8

(a) Write the expression for the output function in terms of *A*, *B*, *sel*, and *selB*.

(b) Draw an equivalent *RC* circuit model for the path from *A* to *C* assuming that the *sel* signal is high. Write down the individual contributions for each resistance and capacitance and place the total values at the appropriate nodes.

(c) What is the expression for the delay from *A* to *C*? Use Elmore delay here.

(d) Write down the expression for the delay through the output inverter knowing that it is f times larger than the input inverters. Use an *RC* delay here.

(e) Determine the optimal size of the output inverter to minimize the total delay from *A* to OUT. That is, write the complete expression for the delay and then derive the best value of f that would produce the smallest delay.

P7.9. What is the logical effort of the circuits of Figure P7.9 ($W_P = 8\lambda$, $W_N = 12\lambda$)? Consider the fall transition in each case.

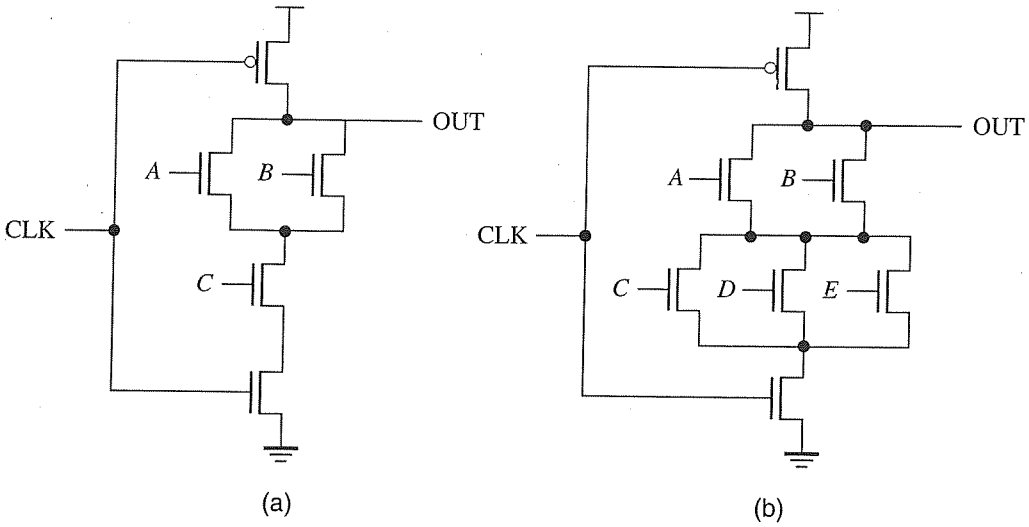


Figure P7.9

P7.10. In the circuit of Figure P7.10, determine the capacitance of each node and calculate the minimum and maximum delay. (All transistors are minimum size, $L = 2\lambda$ and $W = 2\lambda$, unless otherwise specified.)

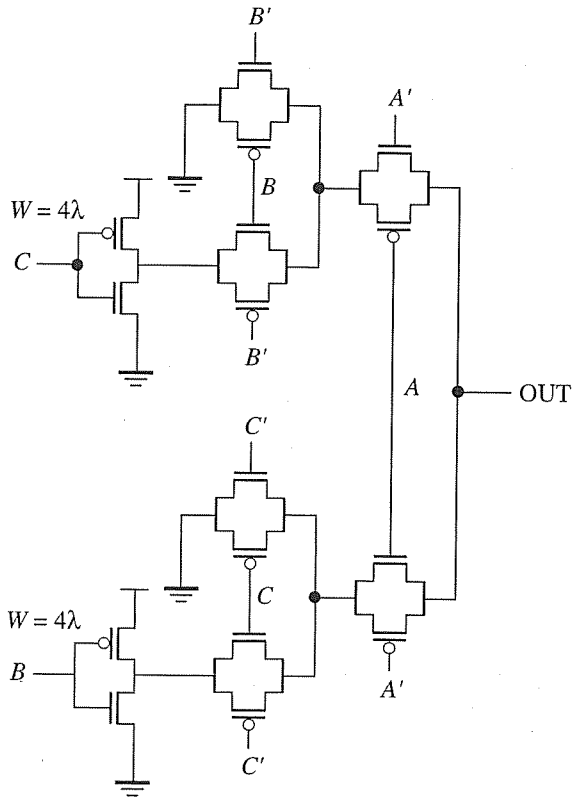


Figure P7.10

P7.11. The precharged circuit of Figure P7.11 is a 2-input AND gate. Draw the waveforms at nodes *F*, *X*, and *Y* as a function of time. You can assume short rise and fall times (i.e., you do not need to calculate them), but the final value in each phase must be correct. Only the capacitances shown here need to be considered. All devices are minimum size. Both *F* and *Y* start out unknown. Use 0.13 μm parameters.

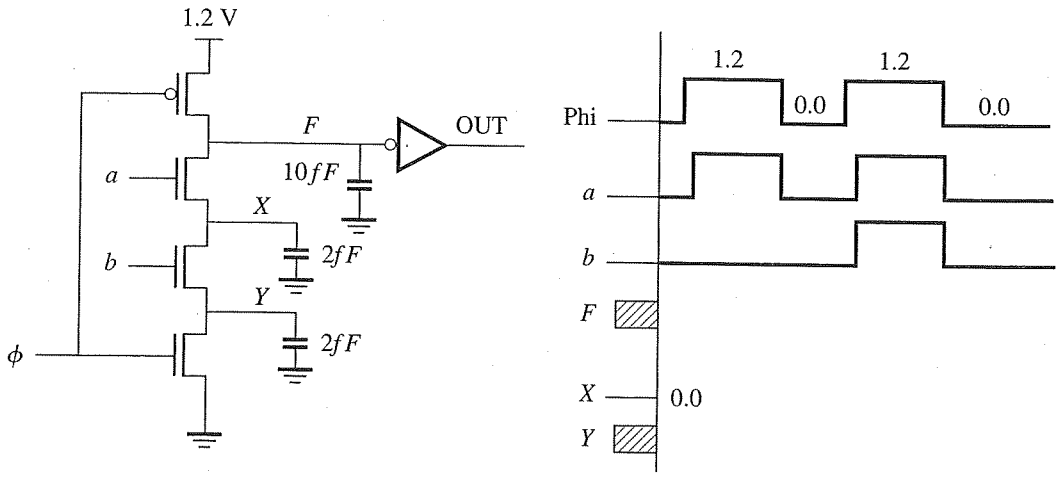


Figure P7.11

P7.12. What is the effect of temperature on static and dynamic logic circuits in terms of speed, power, and leakage current?

P7.13. Shown in Figure P7.13 is a dynamic logic gate in which the precharge device has a *W/L* of 5, the *n*-channel devices are all 3, and the inverter is a skewed inverter with a pull-up of 4 and a pull-down of 1 to reduce the delay.

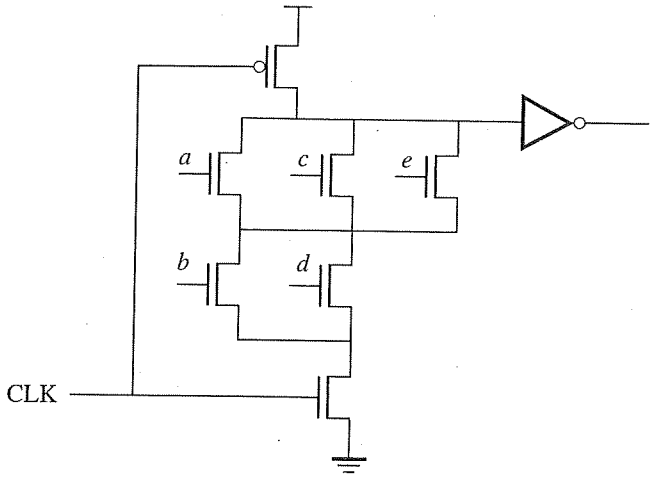


Figure P7.13

(a) What input settings give you the worst-case charge sharing?

(b) What is the amount of charge sharing in the worst-case?

(c) Does this circuit fail in the worst-case charge sharing situation? Explain.

P7.14. The two CMOS circuits in Figure P7.14 are intended to be simple latches. Do these gates work as latches? Are there any problems with using them as latches? Is one better than the other? Explain.

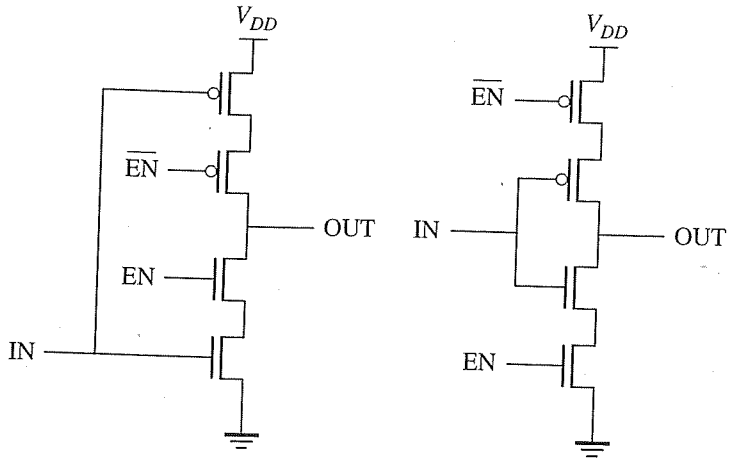


Figure P7.14

P7.15. The circuit shown in Figure P7.15 is a bootstrapped inverter. It uses a linear enhancement load, M_2, with an additional transistor, M_3, to provide gate drive to M_2. A bootstrapping capacitor, C_b, is used to pull node X up when the output is being pulled up. Answer the following questions in terms of the quantities V_{DD}, V_T, V_{OL}, C_X, C_{out}, and C_b.

(a) When the input is V_{DD}, what is the output voltage and the voltage at node X?

(b) When the input switches to a low voltage, what is the required value of C_b that allows the output to reach V_{DD}?

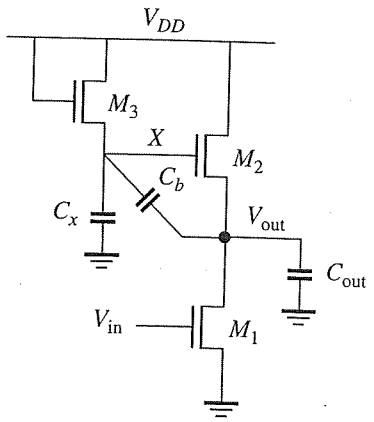


Figure P7.15