CMOS DIGITAL IC DESIGN

Lecture Notes

M.TECH(VLSI & ES)
(I YEAR – I SEM)
(2022-2023)

Prepared by

Dr.M.Arun Kumar, Professor

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution - UGC, Govt. of India)
Department of Electronics and Communication Engineering

Recognized under 2(f) and 12 (B) of UGC ACT 1956
(Affiliated to JNTUH, Hyderabad, Approved by AICTE-Accredited by NBA &NAAC–'A'Grade-ISO9001:2015 Certified) Maisammaguda,Dhulapally(PostVia.Kompally),Secunderabad–500100,Telangana State, India

ELECTIVE-II

CMOS DIGITAL IC DESIGN

Course Objectives:

- To discuss basic CMOS logic gates, implementation of AOI and OAI gates
- Design MOS logic circuits using Transmission gates
- To analyze different delays and power dissipation in number of stages.
- To understand the design of combinational circuits using ratioed, cascade and dynamic logic.
- To design different types of Semiconductor Memories

UNIT –I:

MOS Design:

Pseudo NMOS Logic – Inverter, Inverter threshold voltage, Output high voltage, Output Low voltage, Gain at gate threshold voltage, Transient response, Rise time, Fall time, Pseudo NMOS logic gates, Transistor equivalency, CMOS Inverter logic.

UNIT –II:

Combinational MOS Logic Circuits:

MOS logic circuits with NMOS loads, Primitive CMOS logic gates – NOR & NAND gate, Complex Logic circuits design – Realizing Boolean expressions using NMOS gates and CMOS gates , AOI and OIA gates, CMOS full adder, CMOS transmission gates, Designing with Transmission gates.

UNIT –III:

Sequential MOS Logic Circuits:

Behavior of Bi-stable elements, SR Latch, Clocked latch and flip flop circuits, CMOS D latch and edge triggered Flip-flop.

UNIT –IV:

Dynamic Logic Circuits:

Basic principle, Voltage Bootstrapping, Synchronous dynamic pass transistor circuits, Dynamic CMOST Transmission gate logic, High performance Dynamic CMOS circuits.

UNIT –V:

Semiconductor Memories:

Types, RAM array organization, DRAM – Types, Operation, Leakage currents in DRAM cell and refresh operation, SRAM operation Leakage currents in SRAM cells, Flash Memory- NOR flash and NAND flash.

TEXT BOOKS:

1. Digital Integrated Circuit Design – Ken Martin, Oxford University Press, 2011.
2. CMOS Digital Integrated Circuits Analysis and Design – Sung-Mo Kang, Yusuf Leblebici, TMH, 3 rd Ed., 2011.

REFERENCE BOOKS:

1. Introduction to VLSI Systems: A Logic, Circuit and System Perspective – Ming-BO Lin, CRC Press, 2011
2. Digital Integrated Circuits – A Design Perspective, Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, 2 nd Ed., PHI.

Course Outcomes:

- Able to apply mathematical methods and transistor physics in the analysis of CMOS circuits and design CMOS inverter with different loads for given levels noise margins and propagation delay's.
- Can execute moderately sized digital logic designs with OAI, AOI, and transmission gates.
- Able to design static and dynamic CMOS circuits (both Combinational and sequential) at transistor level and layout level.
- Able to design memory architectures that aids the growth of VLSI designs with reduced access time and reduced power consumption.

UNIT I- MOS DESIGN

Pseudo-NMOS Logic

The original idea behind the pseudo-NMOS inverter is to realize a gate as a common-source amplifier with a current source as a load, as shown in Fig. 4.1. If the gate input voltage is less

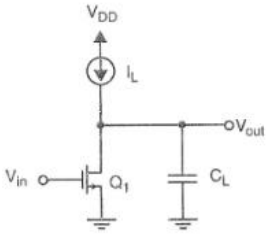


Figure 4.1 A current-source load inverter.

than the threshold voltage[1] of Q_1, then Q_1 will be cut off and I_L will charge the load capacitances to a high voltage, ideally V_{DD}. If, on the other hand, the input signal changes to a high voltage, then initially Q_1 will conduct substantially more current than I_L and the load capacitance will be discharged to a low voltage. When this happens, the voltage across Q_1, that is $V_{DS\text{-}1}$, will be less than the effective gate-source voltage of Q_1, and Q_1 will enter the triode region. This will cause its current to decrease to be equal to I_L. When this occurs, the output voltage will be close to 0 V, assuming the transistor widths have been reasonably chosen.

Normally, to guarantee that the output low voltage, V_{OL}, is close enough to 0 V, it is necessary to make Q_1 sufficiently wide so that its current is many times greater than the value of I_L for the case in which the output is in its transition region around the gate threshold V_{th} while changing from a high to a low voltage. This, however, results in the rise time being considerably larger than the fall time. Thus, most *current-load* gates have poor drive capability for positive-going output transitions, unless they are followed by buffers. Also, they have a power dissipation equal to $I_L V_{DD}$ when the output is low. When the output is high, there is no power dissipation. Thus, approximately half the time they dissipate power. This d.c. power dissipation means that a modern IC containing millions of gates could not be realized using just pseudo-NMOS gates. Nevertheless, these circuits are simple, take up little area, and usually introduce only small loads on preceding gates.

Using a p-Channel Transistor to Realize a Current Source

The basic idea behind pseudo-NMOS logic is to use a p-channel transistor to realize a current-source load as shown in Fig. 4.2. To understand this, consider the equation describing the current of a p-channel MOS transistor in the active or saturation region. For this case, we have

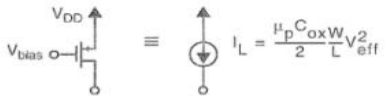


Figure 4.2 Using a p-channel transistor to approximate a current source.

$$I_D = \frac{\mu_p C_{ox} W}{2 L}(V_{SG} + V_{tp})^2 = \frac{\mu_p C_{ox} W}{2 L} V_{eff}^2 \tag{4.1}$$

where $V_{eff} = V_{SG} + V_{tp}$ is the effective gate-source voltage of the transistor. This equation ignores the finite output impedance, r_{ds}, of the transistor. For equation (4.1) to apply, it is necessary that the drain voltage is not greater than the gate voltage by more than $|V_{tp}|$ which is usually about 0.8 to 0.9 V. If we assume that $V_{DD} = 3.3$ V, and that the bias voltage V_{bias} is about half way between ground and V_{DD}, or at 1.65 V, then (4.1) applies as long as the drain voltage is less than 2.45 V, approximately. We now have $V_{SG} = V_{DD} - V_{bias} = 1.65$ V and $V_{eff} = V_{SG} + V_{tp} \approx 0.75$ V in equation (4.1). Therefore, $I_D = I_L$ is independent of the voltage across the transistor as long as this voltage is greater than 0.75 V, or so. Thus, the p-channel transistor can be used to realize an approximate current source as long as the output voltage of the logic gate is less than 2.45 V, or so. This is the case most of the time during an output logic change or transition. When the output voltage becomes greater than approximately 2.45 V, then the load current starts to decrease and a more accurate model would be a current source in parallel with a resistor.

A Pseudo-NMOS Inverter

Figure 4.3 shows a pseudo-NMOS inverter for which the W/L of the drive transistor is one half the W/L of the load device; that is

$$(W/L)_1 = \frac{(W/L)_2}{2} \tag{4.2}$$

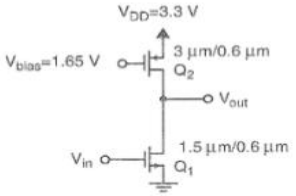


Figure 4.3 A typical pseudo-NMOS inverter in a 0.6-µm technology.

This would be a typical choice for a pseudo-NMOS inverter having a p-channel load with a 1.65-V gate-bias voltage. A simple circuit capable of realizing this bias voltage will be described shortly.

In this section, we will analyze the pseudo-NMOS inverter for its approximate gate threshold voltage, gain at threshold, typical output high and low voltages, and approximate transient responses. To make these analyses tractable, a number of approximations will be made. The errors arising from these approximations vary somewhat, but in all cases are typically less than the errors caused by the inability to accurately predict the processing and transistor parameters, prior to manufacture. Also, by making reasonable approximations, the essential operation of the gate becomes more obvious without miring the reader in complex calculations that are of limited use in designing practical digital integrated circuits. Irrespective of the use of approximations, this section relies heavily on understanding the equations that describe MOS transistors (which were presented in Chapter 3). Thus, not only will this section give greater insight into pseudo-NMOS logic, but it will also serve as the first in-depth example of the application of MOS equations to actual circuits.

Inverter Threshold Voltage (V_{TH})

As was mentioned in Chapter 1, the threshold voltage of the pseudo-NMOS inverter is defined as the input voltage that gives an identical output voltage. Normally, for a properly designed MOS inverter, this voltage would be approximately half the power-supply voltage. To calculate this voltage, it is first necessary to determine the region in which a transistor is operating and which equations apply. That is, is the transistor in the triode region or the saturation region (which we typically call the active region)? First, it is safe to say that the enhancement n-channel drive transistor (Q_1) is definitely in the active or saturation region. This is true because for $V_{in} = V_{out}$, the gate-drain voltage of Q_1 is zero. Any enhancement transistor with $V_{DG} = 0$ is in the active region. It will also be assumed that the p-channel load transistor is also in the active region. This will be the case as long as the gate-threshold voltage happens to be less than 2.45 V, which is almost certainly true for a properly designed inverter. If this were not the case, then the device sizes would have been poorly chosen and the gate would be impractical, in which case it would not be described in this text. Under these assumptions, we may write the following equations. For Q_2 we have

$$I_{D\text{-}2} = \frac{\mu_p C_{ox}}{2}\left(\frac{W}{L}\right)_2\left(\frac{V_{DD}}{2} + V_{tp}\right)^2 \tag{4.3}$$

where it is assumed that $V_{SG\text{-}2} = V_{DD}/2$. For Q_1, we have $V_{GS\text{-}1} = V_{in} = V_{th}$ and

$$I_{D\text{-}1} = \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)_1 (V_{th} - V_{tn})^2 \tag{4.4}$$

Since it is always assumed that MOS gates drive only capacitive loads, we have $I_{D\text{-}1} = I_{D\text{-}2}$ and we may equate (4.3) and (4.4) and solve for V_{th} to obtain

$$V_{th} = V_{tn} + \sqrt{\frac{\mu_p (W/L)_2}{\mu_n (W/L)_1}}\left(\frac{V_{DD}}{2} + V_{tp}\right) \tag{4.5}$$

Example 4.1

For $\mu_n/\mu_p = 4.2$, $V_{tn} = 0.8$ V and $V_{tp} = -0.9$ V, find V_{th} for the inverter of Fig. 4.3.

Solution: For $(W/L)_2/(W/L)_1 = 2$, we have using equation (4.5) $V_{th} = 1.32$ V.

Typical Output High Voltage V_{OH}

The output voltage is high when the input voltage is a "0". Assuming the typical input "0" voltage is less than V_{tn}, then Q_1 will be completely cut off. Under this condition, the source-drain voltage of Q_2 will be very small. For this case, we have $V_{SD2} \ll V_{eff\text{-}2}$ and Q_2 is *hard in the triode region.* A transistor that is hard in the triode region has its current given by

$$I_{D2} \cong \mu_p C_{ox}\left(\frac{W}{L}\right)_2 V_{eff\text{-}2} V_{SD\text{-}2} \tag{4.6}$$

Thus, its current is approximately proportional to the voltage across it. Since $V_{eff\text{-}2} = V_{SG\text{-}2} + V_{tp} = V_{DD}/2 + V_{tp} = 0.75$ V, we can approximate Q_2 by a resistor of size

$$r_{ds\text{-}2} = \frac{1}{\mu_p C_{ox}(W/L)_2 V_{eff\text{-}2}} \tag{4.7}$$

Since Q_1 is off, there will be no current through Q_2, and according to equation (4.6), $V_{SD\text{-}2} = 0$, which implies $V_{out} = V_{OH} = V_{DD}$. Thus, the typical output high voltage is V_{DD}.

It should be emphasized that the approximation of equation (4.7) is valid only for $V_{SD\text{-}2}$ close to 0 V or equivalently for V_{out} close to V_{DD}. It is not valid when the output is in the transition region around V_{th} when changing from a "0" to a "1". Rather, it is valid only when the output voltage has completely finished its change during a transition.

Typical Output Low Voltage V_{OL}

When the input is a typical "1", from the previous section we have $V_{GS\text{-}1} = V_{DD} = 3.3$ V. Thus, Q_1 will be *hard on.* Assuming we have chosen the device sizes correctly, the output will be a low voltage. Thus, it is safe to say that Q_2 will be in the active region and acts like a current source of size

$$I_L = I_{D\text{-}2} \cong \frac{\mu_p C_{ox}}{2}\left(\frac{W}{L}\right)_2 V_{eff\text{-}2}^2 \tag{4.10}$$

Also, since V_{out} is low, we have $V_{DS\text{-}1}$ very small (i.e., much less than its effective gate voltage, which is $V_{GS\text{-}1} - V_{tn}$ or 2.5 V) and Q_1 is hard in the triode region. Thus, Q_1 can be approximated by a resistor of size

$$r_{ds\text{-}1} = \frac{1}{\mu_n C_{ox}(W/L)_1 (V_{DD} - V_{tn})} \tag{4.11}$$

The output low voltage is then simply given by

$$V_{OL} = I_{D\text{-}2} r_{ds\text{-}1} = \frac{1}{2}\frac{\mu_p}{\mu_n}\frac{(V_{DD}/2 + V_{tp})^2}{(V_{DD} - V_{tn})}\frac{(W/L)_2}{(W/L)_1} \tag{4.12}$$

GAIN AT THE GATE THRESHOLD VOLTAGE

One of the commonly used figures of merit for an inverter is the small-signal gain of the inverter when it is at the operating point $V_{in} = V_{out} = V_{th}$. To find this gain, the transistors should be replaced by their small-signal models described in Chapter 3. The small-signal model of an MOS transistor at low frequencies, where the capacitors have been removed, is shown in Fig. 4.4. This model was described in Chapter 3, but for convenience the equations for the various parameters will be repeated here.

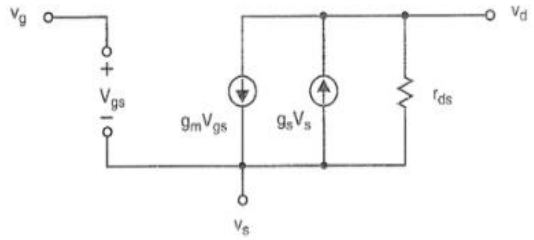


Figure 4.4 The small-signal model of an MOS transistor at low frequencies.

The voltage-controlled current source, $g_m v_{gs}$, is responsible for the transistor operation. In an ideal transistor, this would be the only component in the model. The transconductance g_m is given by

$$g_m = \mu_n C_{ox}\frac{W}{L}(V_{GS} - V_{tn}) = \mu_n C_{ox}\frac{W}{L}V_{eff} = \sqrt{2\mu_n C_{ox}\frac{W}{L}I_D} \tag{4.14}$$

The second voltage-controlled current source, $g_s v_s$, models the body effect. It is responsible for the decrease in current as the source voltage, v_s, increases. This is the reason its direction is opposite that of $g_m v_{gs}$. In many introductory texts, this current source is not included in MOS small-signal models. For the pseudo-NMOS inverter, the sources of both the drive transistor and the p-channel load transistor are connected to ground and V_{DD}, respectively, which are both *small-signal grounds*. Therefore, for both transistors, $v_s = 0$, and the current source modeling the body effect can be ignored. The final parameter of the model is the output resistance of the transistor, r_{ds}. For transistors that do not have short-channel effects, this models the reduction of the channel length due to the increase in the length of the pinch-off region at the drain end of the channel when the drain voltage increases. For this case it is approximately given by

$$r_{ds} \cong \frac{L\alpha\sqrt{V_{DS} + V_t}}{I_D} \tag{4.15}$$

where α is a process-dependent constant on the order of $5\sqrt{V/\mu m}$ (i.e., L in micrometers). For transistors having short-channel effects, the output resistance will be much smaller. For this case we have

$$r_{ds} = \frac{1}{\lambda I_D} \tag{4.16}$$

where λ must now be empirically derived from transistor measurements. This must be done for every different channel length that is to be used.[2]

In deriving the small signal-model of the inverter, V_{DD} is replaced by ground as it is assumed to be a constant voltage. This leads to the small-signal equivalent circuit shown in Fig. 4.5a. Note that $v_{g\text{-}2} = v_{s\text{-}2} = 0$ implies both of the current sources $g_{m\text{-}2}v_{gs\text{-}2}$ and $g_{s\text{-}2}v_{s\text{-}2}$ are zero. Also, since $v_{s\text{-}1} = 0$, $g_{s\text{-}1}v_{s\text{-}1} = 0$, as well. This allows the circuit to be simplified to that shown in Fig. 4.5b. The gain is now readily calculated to be

$$\frac{V_{out}}{V_{in}} = \frac{-g_{m\text{-}1}}{1/r_{ds\text{-}1} + 1/r_{ds\text{-}2}} \tag{4.17}$$

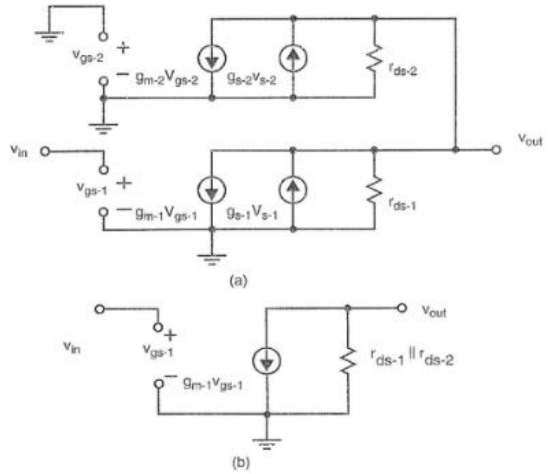


Figure 4.5 (a) The small-signal equivalent circuit of a pseudo-NMOS inverter and (b) a simplified small-signal equivalent circuit of the inverter.

This gives a gain at the gate threshold voltage on the order of –15 to –50 depending on processing parameters for a particular technology.

Also,

$$I_{D-1} = \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)(V_{th} - V_{tn})^2 = 63.5\ \mu A \tag{4.19}$$

Thus,

$$r_{ds-1} = \frac{1}{\lambda_n I_{D-1}} = 262\ k\Omega \tag{4.20}$$

Also, since $I_{D-2} = I_{D-1}$, we have

$$r_{ds-2} = \frac{1}{\lambda_p I_{D1}} = 242\ k\Omega \tag{4.21}$$

as well. Finally, using (4.17), we get

$$\frac{v_{out}}{v_{in}} = \frac{-0.244}{1/262 + 1/242} = -31.0 \tag{4.22}$$

The transfer curve, which is a plot of V_{out} as a function of V_{in}, was obtained using SPICE and is shown in Fig. 4.6. The gain at the threshold voltage is –10.9, which is significantly different from that obtained through hand analysis. Estimating circuit gain is an area in which SPICE is especially inaccurate. Luckily, accurate gain estimates are not important when realizing digital circuits.

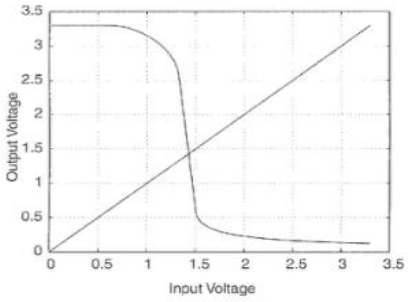


Figure 4.6 The transfer curve of the pseudo-NMOS inverter of Fig. 4.3 obtained using SPICE.

TRANSIENT RESPONSE

The speed with which the output of a gate can change state is limited because the transistors can supply only a finite amount of current to charge or discharge the parasitic capacitances that exist at every node. The parasitic capacitance is composed of three components: the input capacitance of the gates being driven by the output, the capacitance due to the interconnect wiring, and the capacitance due to reverse-biased junctions at the output node of the gate. There is often additional delay terms due to the requirement that internal nodes of a gate must change state before the output can change state. For the pseudo-NMOS inverter, these additional terms are not present, since there is only one node, the output node.

Often, when a gate is driving only a couple of other gates, the junction capacitance at the output node can be the dominant, if not the major, component of the load capacitance. As explained in Chapter 2, this capacitance is highly nonlinear. Its magnitude is also difficult to predict at design time because the area of the junctions is not known before the layout has been done and can only be estimated. For these reasons, *it is very important that the designer guarantees the ICs are functional irrespective of accurate knowledge of gate delays.* Also, the designer should supervise the layout of the circuit to ensure that the junction areas of the critical nodes are minimized.

The exact analytical derivation of the transient response of an NMOS inverter (Taub and Schilling, 1977; Hodges and Jackson, 1988) is tedious and complicated due to the highly nonlinear nature of the transistors and the capacitances (both junction capacitances and gate input capacitances). Given that the load capacitances are never known accurately at the time of design, it is deemed that the exact calculation of the transient delays is not of much use and would never be utilized by a digital designer working under usual time constraints. Rather, approximate methods that give the correct order of magnitude of the response times, and can be used to quickly determine critical nodes, must be developed.

In analyzing the pseudo-NMOS inverter for its approximate delay times, it will be assumed that the load capacitance is ideal and known. In the examples, it will be assumed to be 0.2 pF. Later in Chapter 6, we will discuss how to estimate this capacitance in greater detail.

RISE TIME

First, we will calculate the time from the point at which the input undergoes a step change from a "1" to a "0" to when the output has undergone a 70% change to approximately 2.3 V. As mentioned in Chapter 1, the reason for using the 70% rise time as a figure of merit is that *if a large number of inverters are cascaded, then the sum of the 70% rise and fall times is approximately equal to the total delay through the chain of inverters.* During the 70% rise time, Q_1 will be off and thus will be ignored. Initially, Q_2 is in the active region and has a current of approximately

$$I_{D\text{-}2} = \frac{\mu_p C_{ox}}{2}\left(\frac{W}{L}\right)_2 V_{eff\text{-}2}^2 = \frac{\mu_p C_{ox}}{2}\left(\frac{W}{L}\right)_2\left(\frac{V_{DD}}{2} + V_{tp}\right)^2 \tag{4.23}$$

This remains true until the voltage across Q_2 decreases to $V_{eff\text{-}2} = 0.75$ V or equivalently when $V_{out} = 2.55$ V. Since the output must reach 2.3 V to undergo a 70% change, the transistor Q_2 will remain in the active region for the duration of the 70% rise time. Therefore, $I_{D\text{-}2}$ remains constant at the value given by (4.23). We have

$$t_{+70\%} = \frac{C_L}{I_{D\text{-}2}}\Delta V_{out} = \frac{2C_L 2.3}{\mu_p C_{ox}(W/L)_2[(V_{DD}/2) + V_{tp}]^2} \tag{4.24}$$

where C_L is the total load capacitance in Farads.

FALL TIME

When calculating the fall time, it is assumed that at time 0, the input undergoes a step change to V_{DD}, which turns Q_1 on *hard*. The fall time is defined as the time it takes the output to decrease from 3.3 V through a 70% change to 1.0 V. During most of this time, the current through Q_1 is much larger than the current through Q_2 and thus Q_2 will be ignored. This will make our estimate somewhat optimistic, but helps to simplify the analysis.

Initially Q_1 is in the active region until $V_{DG\text{-}1} < -V_{tn}$ or equivalently $V_{DS\text{-}1} < V_{eff\text{-}1} = V_{GS\text{-}1} - V_{tn}$. This occurs when the output is discharged to 2.5 V (for $V_{tn} = 0.8$ V). Next, Q_1 enters the triode region during most of the fall time. The analytic derivation of the fall time is possible and has been given (Hodges and Jackson, 1988), but is of little use during design. Instead, it is desired to find a formula for an approximately equivalent resistor that would give a similar fall time.

As mentioned, the current through Q_1 is highly nonlinear as the output voltage changes from 3.3 to 1.0 V. Figure 4.7 shows a plot of this current as a function of the output voltage. If we could find a value for a resistor that gives approximately the same current as the actual transistor, the analysis would be greatly simplified. To find a resistor that is approximately equivalent to Q_1, it is necessary to find an I–V curve that is straight, passes through the origin, and results in the same fall time as obtained with Q_1. One approximation that has been

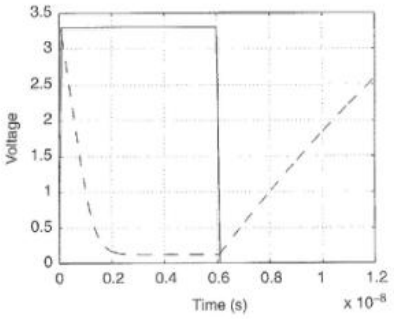


Figure 4.7 The transient response of the inverter of Fig. 4.3 for $C_L = 0.2$ pF.

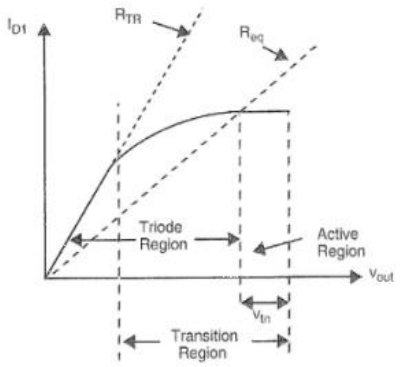


Figure 4.8 The current through Q_1 as a function of the output voltage.

suggested previously is to use a resistor that is equal to the resistance of Q_1 when $V_{DS\text{-}1}$ is small (i.e., Q_1 is hard in the triode region). The equation for this resistance (denoted R_{TR}) is

$$R_{TR} = \frac{1}{\mu_n C_{ox}(W/L)_1(V_{DD} - V_{tn})} \tag{4.25}$$

However, as can easily be seen from Fig. 4.8, this approximation would result in a resistor that always has more current in the transition region than Q_1 has. A better approximation is a resistor whose I–V curve intersects the I–V curve of Q_1 at the edge of the triode region. This approximation is shown in Fig. 4.8 as the straight line labeled R_{eq}. It can be seen that for the

initial interval of the fall time, the current through R_{eq} will be larger than that of Q_1, and for the second part of the transition, the current through R_{eq} will be less than that through Q_1.

The voltage at the intersection point of the I–V curves of R_{eq} and Q_1 is $V_{DD} - V_{tn}$. The current at the intersection point is the current of Q_1 in the active region, that is

$$I_{D\text{-}1} = \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)_1 (V_{DD} - V_{tn})^2 \tag{4.26}$$

The value for R_{eq} is the ratio of the voltage to the current, that is

$$R_{eq} = \frac{V_{DD} - V_{tn}}{\frac{\mu_n C_{ox}}{2}(W/L)_1 (V_{DD} - V_{tn})^2} = \frac{2}{\mu_n C_{ox}(W/L)_1 (V_{DD} - V_{tn})} \tag{4.27}$$

Thus, the approximately equivalent resistance to Q_1 is twice R_{TR}, the triode resistance of Q_1. It turns out that (4.27) results in transient times that are a little over 20% too low. Therefore, some designers modify (4.27) to be somewhat larger according to

$$R_{eq} = \frac{2.5}{\mu_n C_{ox}(W/L)_1 (V_{DD} - V_{tn})} \tag{4.28}$$

Others will simply use (4.27) and then add on 20% additional time after the estimate. Either approach is valid.

4.2 Pseudo-NMOS Logic Gates

In Chapter 1, a number of NMOS logic gates were introduced. In this section, *pseudo-NMOS* logic gates will be revisited, in a somewhat more formal manner, where extensive use will be made of the equivalent-resistor approximation. This section will also deal with how to choose the device sizes for NMOS logic gates.

An example of NMOS logic is the straightforward realization of the *exclusive-or* function. The *exclusive-or* is defined by

$$\begin{aligned} y = x_1 \oplus x_2 &= x_1\overline{x_2} + \overline{x_1}x_2 \\ &= \overline{x_1 x_2 + \overline{x_1}\,\overline{x_2}} \\ &= \overline{x_1 x_2 + \overline{x_1 + x_2}} \end{aligned}$$

A logic circuit that realizes this function is shown in Fig. 4.9a. A pseudo-NMOS realization of this logic circuit is shown in Fig. 4.9b. The *nor* gate *a* is realized by transistors Q_1, Q_2, and Q_3. The second NMOS gate, composed of transistors Q_4 to Q_7, is a compound gate; it realizes both the *nor* gate *c* and the *and* gate *b* by a series connection of drive transistors. Notice, how the series connected transistors (Q_5 and Q_6) are chosen to be wider than the parallel transistor Q_4.

The general form of a typical NMOS logic gate is shown in Fig. 4.10. It consists of a single p-channel load transistor, between V_{DD} and V_{out}, that has its gate connected to a bias voltage.

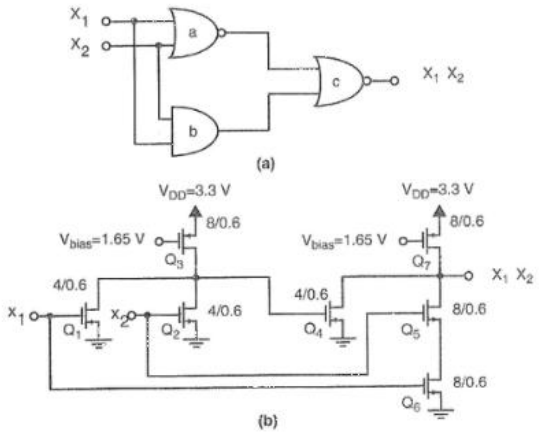


Figure 4.9 (a) A logic circuit realization of the *exclusive-or* function. (b) An NMOS realization of the *exclusive-or* function.

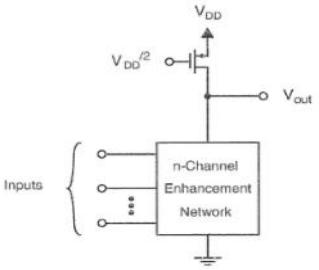


Figure 4.10 A generalized pseudo-NMOS logic gate.

In addition, there is a network of n-channel-enhancement *drive transistors* between V_{out} and ground. Each of the drive transistors in this network has its gate connected to an input. Depending on the logic values of the inputs, the network of drive transistors will either provide an infinite impedance between V_{out} and ground, in which case the output will be V_{DD}, or will provide a low enough impedance so that V_{out} will be pulled low to a "0".

In the event the network of drive transistors is low impedance, such impedance must not be larger than that of a single enhancement transistor, with its gate connected to V_{DD}, and having its W/L at least one-half the size of the load transistor. This constrains the minimum W/Ls of the drive transistors.

The maximum W/Ls of the drive transistors are constrained by the desire for speed. Making them larger than necessary makes the logic slower than it need be for two reasons; first, it increases the parasitic capacitances of the internal nodes of the gate; second, it increases the output load capacitance of preceding gates. Taking the W/Ls too large does not cause the gate to function incorrectly.

Before describing how the W/Ls of the n-channel network should be chosen, it is necessary to state some important theorems regarding approximate transistor equivalency that will be used often in this text.

4.3 Transistor Equivalency

1. *Two transistors are equivalent if the ratio of their W/Ls is equal independent of the absolute values of* W *or* L. This theorem is depicted pictorially in Fig. 4.11. This statement is obvious since the large signal I–V equations are only functions of W/L. In reality, there are second-order effects that make this statement only approximate. A transistor with a smaller W tends to have slightly less current than that predicted by (4.1) due to the cross section of the channel, as viewed along the channel being rounded rather than square. Also, a transistor with a shorter length tends to inaccuracies due to short-channel effects. Because of these second-order effects, transistors having very large Ws (50 to 1000 μm) are usually realized by connect-

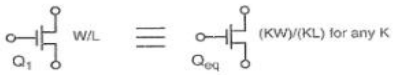


Figure 4.11 Scaling both W and L of a transistor has no effect on a first-order approximation.

Figure 4.12 Two transistors in parallel (Q_1 and Q_2) having equal Ls, are equivalent to a single transistor having a width equal to the sum of the widths of Q_1 and Q_2.

ing a number of smaller transistors (with Ws on the order of 10 to 25 μm) in parallel. As we will be doing only approximate analyses by hand, these second-order effects will be ignored.

2. *Two transistors, with the same length, connected in parallel, are equivalent to a single transistor having a width equal to the sum of the widths of the two transistors.* This theorem is depicted graphically in Fig. 4.12. The proof is simple and straightforward. Assume there are two transistors, Q_1 and Q_2, having identical lengths L and widths W_1 and W_2, respectively. Furthermore, assume their gates, sources, and drains are all connected together, and the transistors are in the active region. Then the total current, I_T, is given by

$$I_T = I_{D\text{-}1} + I_{D\text{-}2} = \frac{\mu C_{ox}(W_1 + W_2)}{2L}(V_{GS} - V_t)^2 \tag{4.30}$$

But this is the I–V equation of a single transistor having a width $W_1 + W_2$. A similar proof can be given for transistors in the triode region.

If the lengths of the transistors are not equal, then one of the transistors can have both its W and L scaled (according to Theorem (1)) to make the lengths equal and Theorem (2) can then be applied to simplify the network.

3. *Two transistors having equal widths, connected in series, and having their gates connected together, are equivalent to a single transistor with a length equal to the sum of the individual lengths.* This theorem is shown pictorially in Fig. 4.13. The proof of this theorem, although straightforward, does require a little algebra. Q_1 will certainly be operating in the triode region, since its drain-source voltage is small, and has a current given by

$$I_{D\text{-}1} = \mu C_{ox}\frac{W}{L_1}\left[(V_{GS\text{-}1} - V_t)V_{DS\text{-}1} - \frac{V_{DS\text{-}1}^2}{2}\right] \tag{4.31}$$

Figure 4.13 Two transistors (Q_1 and Q_2), having equal Ws and connected in series, with their gates connected together are equivalent to a single transistor having a length equal to the sum of the lengths of Q_1 and Q_2.

Assuming Q_2 is in the active region, its gate-source voltage is given by

$$V_{GS\text{-}2} = V_{GS\text{-}1} - V_{DS\text{-}1} \tag{4.32}$$

Then, the drain current of Q_2 is given by

$$\begin{aligned} I_{D\text{-}2} &= \frac{\mu C_{ox}}{2}\frac{W}{L_2}(V_{GS\text{-}1} - V_{DS\text{-}1} - V_t)^2 \\ &= \frac{\mu C_{ox}}{2}\frac{W}{L_2}[(V_{GS\text{-}1} - V_t) - V_{DS\text{-}1}]^2 \end{aligned} \tag{4.33}$$

Expanding (4.33), we have

$$I_{D\text{-}2} = \frac{\mu C_{ox}}{2}\frac{W}{L_2}[(V_{GS\text{-}1} - V_t)^2 - 2(V_{GS\text{-}1} - V_t)V_{DS\text{-}1} + V_{DS\text{-}1}^2] \tag{4.34}$$

Since $I_{D\text{-}1} = I_{D\text{-}2}$, we can equate (4.31) and (4.34) and rearrange to get

$$\left(\frac{W}{L_1} + \frac{W}{L_2}\right)(V_{GS\text{-}1} - V_t)V_{DS\text{-}1} - \left(\frac{W}{L_1} + \frac{W}{L_2}\right)\frac{V_{DS\text{-}1}^2}{2} = \frac{1}{2}\frac{W}{L_2}(V_{GS\text{-}1} - V_t)^2 \tag{4.35}$$

$$\Rightarrow (V_{GS\text{-}1} - V_t)V_{DS\text{-}1} - \frac{V_{DS\text{-}1}^2}{2} = \frac{1}{2}\frac{L_1}{L_1 + L_2}(V_{GS\text{-}1} - V_t)^2 \tag{4.36}$$

Substituting (4.36) into (4.31) we get

$$I_{D1} = \frac{\mu C_{ox}}{2}\frac{W}{L_1 + L_2}(V_{GS\text{-}1} - V_t)^2 \tag{4.37}$$

But this is the I–V equation of a single transistor having length $L_1 + L_2$ and the proof is completed. If Q_2 had been in the triode region, a similar proof is possible. This is left as an exercise for the reader (Problem 4.7).

As before, if the widths of Q_1 and Q_2 are not equal, then one of the transistors can be scaled and the circuit can then be simplified.

It is now possible to describe a procedure for simplifying the network of drive transistors of an NMOS gate to either an open circuit, or a single equivalent transistor:

1. All transistors that have "0" inputs at their gates are off and can be eliminated from the circuit.
2. Any transistors in series with transistors that are off can be eliminated from the circuit.
3. All the remaining transistors have equal gate voltages. Theorems 1 and 3 can now be repeatedly applied to replace parallel and series connections by equivalent transistors until eventually a single transistor (or an open circuit) results. This procedure is best illustrated by an example.

Realizing Complex Pseudo-NMOS Gates

As has been seen, quite complex logic functions can be realized by single NMOS gates. In general, any combination of noninverting *and* and *or* gates feeding into a single inverting *nand* or *nor* gate can be realized by a single pseudo-NMOS gate. The procedure for doing this (synthesis) is the reverse of the procedure for finding the logic function of an NMOS gate (analysis). The final inversion is realized because the n-channel drive network pulls the output low. Therefore, starting at the output, realize the inversion by the combination of the n-channel drive network along with the load transistor. The final *nand* or *nor* gate should now be considered an *and* or *or* gate, respectively. Now, working from the output to the input, recursively realize the gate by subnetworks. An n-input *and* gate is realized by n series subnetworks; an n-input *or* gate is realized by n parallel subnetworks. Each of these logic functions in turn is decomposed. If the input to an *and* or *or* gate does not come from another logic gate, but is an input to the overall circuit from the *outside world*, then the recursion is terminated by an enhancement drive transistor. As before, this procedure is most readily understood with the use of some simple examples.

Realize the logic circuit of Fig. 4.17 by an NMOS gate.

Solution: The step-by-step procedure is illustrated in Fig. 4.18.

1. The inversion of *or* gate *a* is realized by the p-channel transistor load in combination with the complete n-channel driver network. Since gate *a* is now considered a three input or gate, it translates to three parallel subnetworks between the output and ground. Since one of the inputs of gate *a* is from the outside world (i.e., x_8), one of the subnetworks is simply a transistor (Q_8). This is illustrated in Fig. 4.18a.

2. Next, subnetwork 1 is decomposed. It corresponds to gates *b* and c in Fig. 4.17. *And* gate *b* corresponds to two series subnetworks between the output and ground, one of which is a transistor Q_6 having input x_6. The other subnetwork, corresponding to *or* gate *c* translates to two parallel transistors Q_1 and Q_2 having inputs x_1 and x_2. This step of the realization is illustrated in Fig. 4.18b.

3. Finally, a similar procedure is used for realizing subnetwork 2. The complete circuit is shown in Fig. 4.18c.

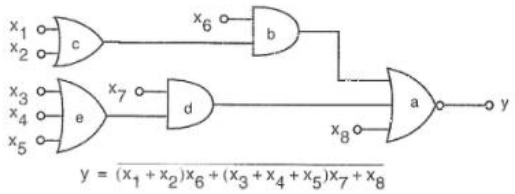


Figure 4.17 A logic circuit used in Example 4.10.

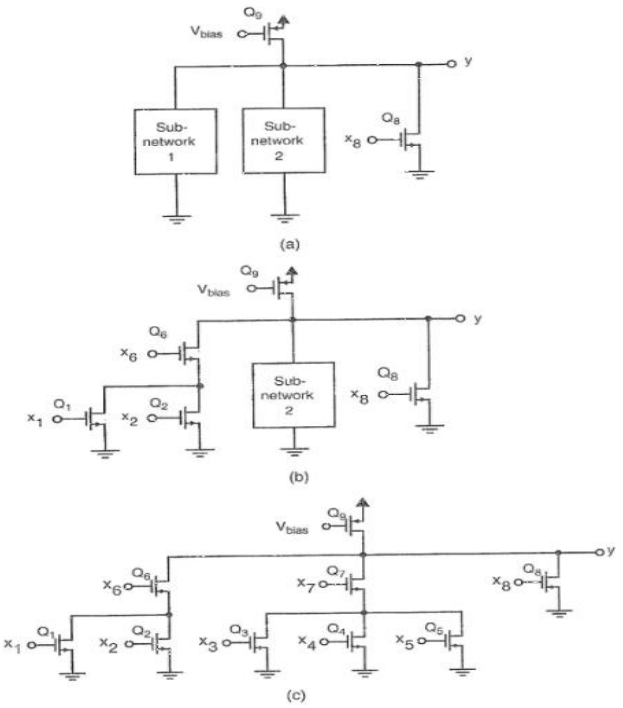


Figure 4.18 The various stages of realizing the logic function of Fig. 4.17.

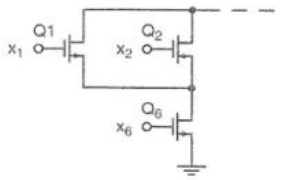


Figure 4.19 An alternative (and inferior) realization of subnetwork 1 of Fig. 4.18a.

It should be noted that the realization of subnetwork 1 in the above example is not unique. It would have been possible to have Q_6 connected to ground and Q_1 and Q_2 connected to the output, as shown in Fig. 4.19. As will be shown later in this chapter, this would result in a slower gate, as it would increase the capacitance at the output node. The following rule is used for minimum delay: *when realizing and functions by a series connection of subnetworks, place the more complicated subnetworks closer to ground.* This keeps the parasitic capacitances introduced by the transistor junctions as close to ground as possible, which in turn means they are being discharged by the smallest equivalent resistance possible. This principle is particularly important for dynamic gates, which will be described in a later chapter. As a final note, never realize NMOS gates that can have more than four drive transistors in series as this would seriously degrade the speed of the gate.

Choosing Transistor Sizes

As mentioned previously, the basic rule in choosing transistor sizes is that for any possible input combination that results in a finite impedance between the output and ground, the equivalent transistor to the n-channel driver network should have a W/L at least one-half that of the load, assuming the load is a p-channel transistor with its gate connected to $V_{DD}/2$. Given this constraint, the W/Ls of the driver network should be as small as possible.

The relative size of the p-channel load transistor (i.e., its W/L) is chosen as a compromise between speed and size versus power dissipation. The larger the W/L of the load transistor, the faster the gate will be, particularly when driving many other gates or a bus. Unfortunately, this increases the power dissipation and the area of the driver network. A typical W/L might be around 5 $\mu m/L_{min}$ or 10 $\mu m/L_{min}$ where L_{min} is the minimum channel length of the process.

Once the size of the load transistor has been chosen, then a simple procedure can be used to choose the W/Ls of the drive transistor. Although it is not an optimum procedure for maximizing speed, the difference in speed between the resulting gate and the optimum gate is usually small. The procedure is as follows:

1. Let $(W/L)_{eq}$ be equal to one-half the W/L of the p-channel load transistor.
2. For each transistor Q_i, determine the maximum number of drive transistors it will be in series with, for all possible inputs. Denote this number n_i. This can almost always be determined by a very cursory examination.
3. Take $(W/L)_i = n_i (W/L)_{eq}$.

Power Dissipation

A pseudo-NMOS logic gate having a "1" output has no d.c. power dissipation. A pseudo-NMOS logic gate having a "0" output has a d.c. power dissipation equal to the current of the p-channel load transistor multiplied by the power supply voltage. Thus, the power dissipation of a gate with a "0" output is given by

$$P_d = \frac{\mu_p C_{ox}}{2}\left(\frac{W}{L}\right)_p V_{eff\text{-}p}^2 V_{DD} \tag{4.51}$$

where $(W/L)_p$ is the size of the p-channel load transistor. Assuming that a pseudo-NMOS logic gate will have a "1" output for half the time and a "0" output for half the time, the average power dissipation will be

$$P_d = \frac{\mu_p C_{ox}}{4}\left(\frac{W}{L}\right)_p V_{eff\text{-}p}^2 V_{DD} \tag{4.52}$$

In addition to this d.c. power, there is a.c. power dissipated whenever a capacitor is charged or discharged. Also, most ICs require buffers for driving output pins and internal buses. These buffers often dissipate an order of magnitude more power than a typical gate. Altogether, this implies that pseudo-NMOS ICs with tens of thousands of gates (which is not a large IC by modern standards) dissipate a very large amount of power, typically too much for an IC package to dissipate. For this reason, pseudo-NMOS logic is seldom used for a complete circuit; rather, it might be used only in select locations in which the fan-out is small and speed is critical. *However, the design procedures used for the n-channel drive network are identical to the design procedures for the n-type drive networks of other logic families such as traditional CMOS logic, dynamic CMOS logic, and GaAs logic.*

Other Pseudo-NMOS Circuits

The bias voltage for pseudo-NMOS circuits can be easily obtained using a reference circuit such as that shown in Fig. 4.21. The ratio of $(W/L)_1$ to $(W/L)_2$ should be the same as the ratio of $(W/L)_{eq}$ to $(W/L)_p$; $(W/L)_{eq} = (1/2)(W/L)_p$ is a reasonable choice as shown. The transistor Q_3 has been included to realize capacitive loading, which helps minimize noise being injected onto V_{bias}. Its exact size is dependent on how many gates the reference circuit is connected to and can be determined using SPICE-level simulation.

Alternative pseudo-NMOS circuits can be realized that do not require a reference (or bias) circuit by simply connecting the gate of the p-channel load to ground as is shown in Fig. 4.22. Notice that the relative size of the n-channel drive transistor relative to the p-channel load transistor is now different. To obtain reasonable gate threshold voltages and output-low voltages, the typical choice is to now take $(W/L)_{eq} = 2(W/L)_p$. The transfer curve of the inverter of Fig. 4.22, obtained using SPICE, is shown in Fig. 4.23. It is seen that the gate threshold voltage is 1.58 V and the output low voltage is 0.17 V; both are reasonable values. Also note that the gain at the gate threshold voltage is –6.5, which is less than the pseudo-NMOS gate when the p-channel load is biased using a bias voltage. The reason for the lower gain is that when the gate output voltage is at the threshold voltage, the p-channel load transistor is in the triode region and has a smaller drain-source impedance (i.e., r_{ds}) than when it is in the active or saturation region. The analysis for the gain at the gate threshold voltage is left as an exercise for the interested reader (see Problem 4.12).

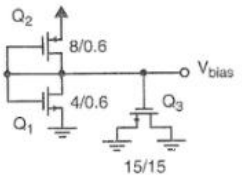


Figure 4.21 A circuit that can be used to generate the bias voltage for pseudo-NMOS gates.

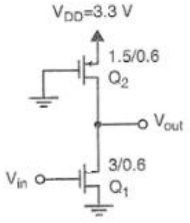


Figure 4.22 An alternative typical pseudo-NMOS inverter in a 0.6 μm technology.

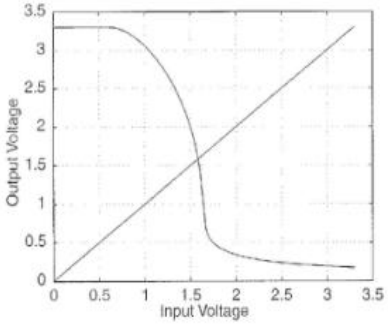


Figure 4.23 The transfer function of the pseudo-NMOS inverter of Fig. 4.22 obtained using SPICE.

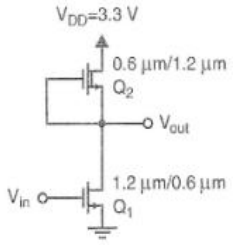


Figure 4.24 A typical NMOS depletion-load inverter in a 0.6 µm technology.

NMOS Logic with Depletion-Load Transistors

The pseudo-NMOS gates we have just described were actually based on an earlier type of logic where n-channel depletion transistors were used as the loads. This logic family was called NMOS logic and was prevalent in the early days of 8-bit microprocessors; now it is seldom used. NMOS logic is very similar to pseudo-NMOS logic; although historically it came second, pseudo-NMOS logic was described first because it is currently used sometimes for critical circuits in an IC, whereas depletion-load NMOS logic is almost never used now. Also, the analysis of pseudo-NMOS logic for the gain at the gate threshold voltage is simpler than for traditional NMOS logic where a second-order effect called the body effect becomes dominant. This section should be considered optional for a first-level course; it has been included because of its historical importance and as an example of using depletion transistors.

An example NMOS depletion-load inverter is shown in Fig. 4.24. The load is realized by a depletion transistor having its gate connected to its source. Since for a depletion transistor, we have $V_{td} < 0$, where V_{td} is the threshold voltage of the transistor, a channel is present for $V_{GS} = 0$. The depletion transistor will be in the active region for $V_{DS} > V_{eff\text{-}d} = -V_{td}$, which might be

on the order of 2 V or a little larger. Assuming this to be the case, we have the drain current of the depletion load transistor given by

$$I_{D\text{-}2} = \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)_2 V_{td}^2 \tag{4.54}$$

Thus, the depletion transistor with its gate connected to its source approximates a d.c. current source ignoring second-order effects; as such it is an extremely efficient realization. In reality, it approximates only a moderate-quality current source. A major source of error is the fact that as the source-to-substrate voltage changes, the depletion-region width between the channel and the substrate changes, which in turn causes the threshold voltage to change according to the equation

$$V_{td} = V_{td\text{-}0} + \gamma\left(\sqrt{V_{SB} + |2\phi_F|} - \sqrt{|2\phi_F|}\right) \tag{4.55}$$

This effect is called the *body effect*. When the inverter output voltage is around the gate threshold voltage, this causes the output impedance of the inverter to be finite, perhaps on the order of only eight times greater than $1/g_{m\text{-}2}$ where $g_{m\text{-}2}$ is the transconductance of the depletion transistor. In addition, the finite output impedance of the depletion and driver transistors also reduce the gain of the inverter somewhat. Together, these two effects cause the gain of the inverter of Fig. 4.24 at its threshold voltage to be given by

$$\frac{V_{out}}{V_{in}} = A_{inv} = \frac{-g_{m\text{-}1}}{g_{s\text{-}2} + g_{ds\text{-}1} + g_{ds\text{-}2}} \tag{4.56}$$

when the output voltage is around the gate threshold voltage. Note that $g_{s\text{-}2}$ is the body-effect transconductance parameter and that $g_{ds\text{-}1}$ and $g_{ds\text{-}2}$ are the drain-source admittances of Q_1 and Q_2, respectively. The derivation (4.56) is left as an exercise for the interested reader (see Problem 4.13). Thus, the gain of the inverter at the gate-threshold voltage might be between –6 and –9. However, this somewhat low gain has little impact on digital circuits. A typical transfer curve of an NMOS inverter obtained using SPICE is shown in Fig. 4.25. A value of $V_{td} = -2$ V was used to obtain this curve. Also, the W/L of the drive transistor was taken to be four times the W/L of the drive transistor as is shown in Fig. 4.25. This ratio of sizes was a typical ratio often chosen by NMOS circuit designers. The gate-threshold voltage is seen to be 1.44 V and the gain at the threshold voltage is –8.3. The output low voltage is 0.14 V. This transfer curve is acceptable.

NMOS logic dominated integrated circuit design during the late 1970s. Indeed, it was NMOS technology that was a key factor in the development of 8- and 16-bit microprocessors that enabled the home-computer revolution. In the early 1980s, CMOS logic, which will be described next, began to become more popular because its power dissipation was much less, despite the fact that the circuits took more area, the yield was less, and the processing was more complicated. In the late 1990s, NMOS logic has practically disappeared due to its power dissipation. However, as was mentioned previously, similar logic families with similar design methodologies are still quite popular.

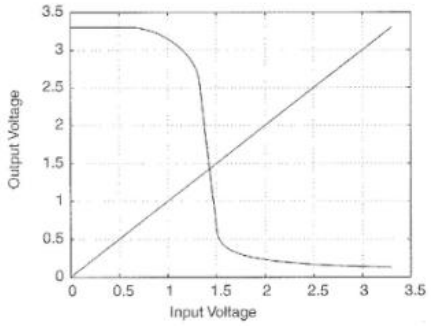


Figure 4.25 The transfer curve of the NMOS inverter of Fig. 4.24.

CMOS Logic

CMOS Logic, first developed for commercial use in the early 1970s, was originally thought to be too complicated, expensive, and slow to compete with NMOS technology. It was also originally prone to a failure mechanism called *latch-up* (which was described in Chapter 3). However, with improvements in technology, the increasing complexity of NMOS processing, and the increasing importance of power dissipation as ICs got larger, CMOS technology has almost completely supplanted NMOS technology at present. Not only does the availability of p-channel transistors allow for much lower power dissipation, but it also gives a much better drive capability for positive-going signals. As ICs become faster, this becomes more critical, especially for output buffers that need to drive large capacitive loads.

Normally, a CMOS IC has only n-channel and p-channel *enhancement* transistors; it is possible to realize *depletion* transistors as well, but typically this is not done because the few times they would be useful does not justify the extra mask required.

As was done for pseudo-NMOS logic, the CMOS inverter will first be revisited in greater detail, and then CMOS logic will be studied more fully and in greater detail than was done in the introduction in Chapter 1.

CMOS Inverter

A typical CMOS inverter is shown in Fig. 4.26. Note that the size of the p-channel transistor is wider than that of the n-channel transistor. This width difference is not needed for functionally correct operation. Rather, it somewhat compensates for the difference in the mobilities of n-channel and p-channel transistors. The effective mobility of n-channel transistors is between two and four times that of p-channel transistors. Making the p-channel device wider by a ratio equal to the inverse of the corresponding mobility ratio gives a gate threshold voltage close to $V_{DD}/2$ and more nearly equal rise and fall times than would otherwise be the case.

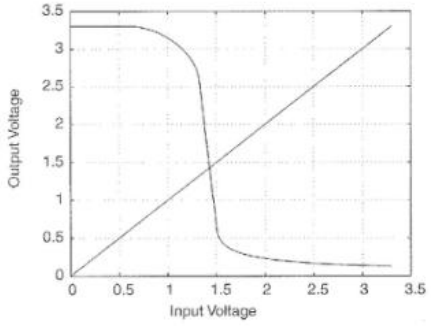


Figure 4.25 The transfer curve of the NMOS inverter of Fig. 4.24.

CMOS Logic

CMOS Logic, first developed for commercial use in the early 1970s, was originally thought to be too complicated, expensive, and slow to compete with NMOS technology. It was also originally prone to a failure mechanism called *latch-up* (which was described in Chapter 3). However, with improvements in technology, the increasing complexity of NMOS processing, and the increasing importance of power dissipation as ICs got larger, CMOS technology has almost completely supplanted NMOS technology at present. Not only does the availability of p-channel transistors allow for much lower power dissipation, but it also gives a much better drive capability for positive-going signals. As ICs become faster, this becomes more critical, especially for output buffers that need to drive large capacitive loads.

Normally, a CMOS IC has only n-channel and p-channel *enhancement* transistors; it is possible to realize *depletion* transistors as well, but typically this is not done because the few times they would be useful does not justify the extra mask required.

As was done for pseudo-NMOS logic, the CMOS inverter will first be revisited in greater detail, and then CMOS logic will be studied more fully and in greater detail than was done in the introduction in Chapter 1.

CMOS Inverter

A typical CMOS inverter is shown in Fig. 4.26. Note that the size of the p-channel transistor is wider than that of the n-channel transistor. This width difference is not needed for functionally correct operation. Rather, it somewhat compensates for the difference in the mobilities of n-channel and p-channel transistors. The effective mobility of n-channel transistors is between two and four times that of p-channel transistors. Making the p-channel device wider by a ratio equal to the inverse of the corresponding mobility ratio gives a gate threshold voltage close to $V_{DD}/2$ and more nearly equal rise and fall times than would otherwise be the case.

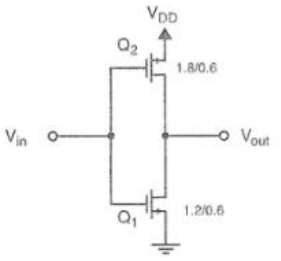


Figure 4.26 A typical CMOS inverter.

Alternatively, taking the p-channel width more closely equal to the width of the n-channel saves on area and helps minimize loads on preceding gates. Irrespective of the relative sizes chosen, traditional CMOS gates will function correctly. For this reason, *traditional CMOS logic is called a ratioless logic family.* Typical choices for the width of p-channel transistors might be between 1 and 2.5 times the width of the n-channel devices. Unless large capacitive loads are being driven, a width 1.5 times that of the n-channel transistor is a reasonable choice most of the time; the exception is when a large number of p-channel transistors are in series, in which case they should be taken correspondingly wider.

Threshold Voltage

When analyzing the CMOS inverter for its threshold voltage (V_{th}), it is known a priori that both the p-channel and n-channel transistors are in their active or saturation regions. This is true because since V_{in} and V_{out} are equal (when $V_{in} = V_{th}$), the drain-gate voltages of both the p- and the n-channel transistors are zero. Since they are both enhancement transistors, this implies they must be in the active region.[3] Thus for the n-channel transistor Q_1, we have

$$I_{D\text{-}1} = \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)_1 (V_{th} - V_{tn})^2 \tag{4.57}$$

and for the p-channel transistor Q_2, we also have[4]

$$I_{D\text{-}2} = \frac{\mu_p C_{ox}}{2}\left(\frac{W}{L}\right)_2 (V_{DD} - V_{th} + V_{tp})^2 \tag{4.58}$$

where we recall that V_{tp} is negative. Setting (4.57) equal to (4.58), and solving for V_{th}, gives

$$V_{th} = \frac{V_{tn} + (V_{DD} + V_{tp})\sqrt{\mu_p(W/L)_2/\mu_n(W/L)_1}}{1 + \sqrt{\mu_p(W/L)_2/\mu_n(W/L)_1}} \tag{4.59}$$

INVERTER GAIN AT $V_{IN} = V_{TH}$

In analyzing the CMOS inverter for its gain at the threshold voltage, the procedure is the same as for the pseudo-NMOS inverter. The small-signal equivalent circuit for the inverter is used, which is then analyzed for its gain. The small-signal equivalent circuit for the CMOS inverter is shown in Fig. 4.27.

Notice that as in all small-signal circuits, the d.c. power supplies have been set to zero. There are no current sources to model the body effect, as the sources of both transistors are connected to small-signal grounds. Note also that the small-signal circuit for the p-channel transistor is identical to the small-signal circuit for the n-channel transistor.[5] Thus, there are two small-signal models for transistors in parallel. This allows Fig. 4.27 to be redrawn as shown in Fig. 4.28.

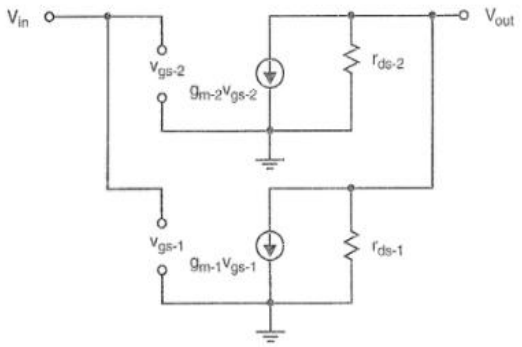


Figure 4.27 The equivalent small-signal model of a CMOS inverter.

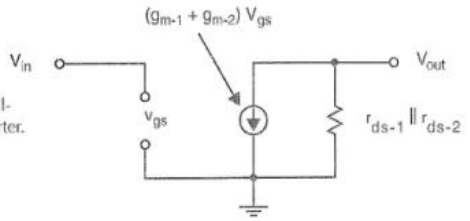


Figure 4.28 A simplified small-signal model of a CMOS inverter.

The gain is now immediately found to be

$$\frac{v_{out}}{v_{in}} = -(g_{m-1} + g_{m-2})(r_{ds-1} \| r_{ds-2}) \tag{4.61}$$

Transient Response

The calculation of the transient response for both the rise and the fall times is almost identical to the calculation of the fall time of the NMOS inverter. For example, with respect to the fall time of the CMOS inverter, the p-channel transistor is off and can be ignored, and the n-channel transistor can be approximated by an equivalent resistor of size

$$R_{eq\text{-}1} = \frac{2.5}{\mu_n C_{ox}(W/L)_1(V_{DD} - V_{tn})} \tag{4.66}$$

Similarly, during the rise time the p-channel transistor can be approximated by

$$R_{eq\text{-}2} = \frac{2.5}{\mu_p C_{ox}(W/L)_2(V_{DD} + V_{tp})} \tag{4.67}$$

Using these approximations, the rise or fall times can now be approximated using the solution to a first-order RC circuit:

$$t_f \cong R_{eq\text{-}1} C_L \ln\left[\frac{V_{out}(\infty) - V_{out}(t_1)}{V_{out}(\infty) - V_{out}(t_2)}\right] \tag{4.68}$$

For a –70% voltage change, this yields

$$t_{-70\%} \cong 1.2 R_{eq\text{-}1} C_L \tag{4.69}$$

Similarly, for the rise time we have

$$t_{+70\%} \cong 1.2 R_{eq\text{-}2} C_L \tag{4.70}$$

The Effect of Transistor Sizes on the Transient Responses

In NMOS logic, the correct choice of the sizes of the transistors was necessary for the gates to work. This is not necessary for typical CMOS logic to operate functionally correctly. However, as was mentioned, the ratio of the W/Ls of the p-channel transistors to the W/Ls of the n-channel transistors does affect the gate threshold voltages and, more importantly, the transient response.

There is no exact optimum ratio for the sizes of p-channel transistors to n-channel transistors that can be specified independent of the application. However, there are two situations in which

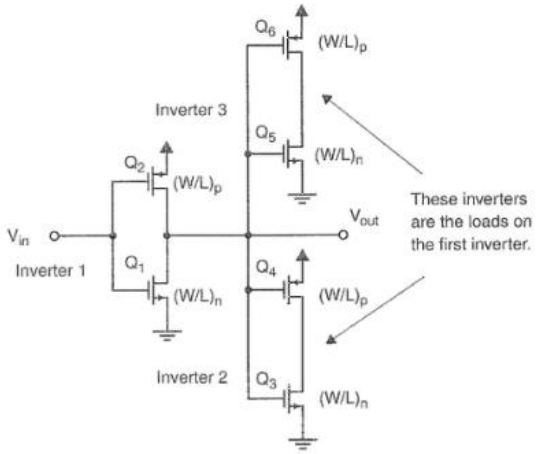


Figure 4.29 A CMOS inverter driving two identical inverters.

something can be said about the relative W/Ls. The first case is when a single gate is driving a number of identical gates.

Consider the situation shown in Fig. 4.29 in which a CMOS inverter is loaded by the input capacitance of two identical inverters. This example is to be analyzed for the transient response times of the first inverter. It is assumed that all n-channel transistors have identical W/Ls. Also, all p-channel transistors are assumed to have the same W/Ls, but these may be different from those of the n-channel transistors. This situation is representative of many typical situations encountered in CMOS design where the fan-out is small.

When considering the transient response of the first inverter, it is first necessary to approximate the capacitive load. There are two components to this load. The first is the junction capacitance of the drains of Q_1 and Q_2. These capacitances are roughly proportional to the widths of Q_1 and Q_2, assuming Q_1 and Q_2 are not too large[6]. In addition, there are the capacitances due to driving the inverters 2 and 3. The input capacitances of these inverters consist principally of the gate capacitances of two n-channel transistors and two p-channel transistors. The exact

determination of these capacitances is a difficult and highly nonlinear problem, but fortunately conservative estimates are easily arrived at using (3.61) of Chapter 3, which is repeated here. When a transistor is *hard* in the triode region, its gate-channel capacitance is approximately given by

$$C_{gs} = WLC_{ox} \tag{4.73}$$

Using equation (4.73), we find the capacitive loading of the two inverters is approximately given by

$$C_L = 2C_{ox}L(W_n + W_p) \tag{4.74}$$

assuming the n- and p-channel transistors have equal lengths, which are also chosen equal to the minimum length. Thus, these input capacitances are also proportional to the widths of the transistors.

Assuming all transistors have equal minimum lengths, then scaling the widths of all the transistors equally, both n-channel and p-channel, has little effect on the transient delays. This is because as the widths go up, the capacitive loads go up proportionally, but the equivalent resistances go down inverse proportionally and the rise and fall time constants are roughly unchanged. This is not exactly true for very small widths in which the junction side-wall capacitances become more important causing larger delays. It also ignores wiring capacitance and external capacitances. When these become important, wider transistors result in smaller delays. However, for the simple case being considered, the independence of the delay on the scaling of all transistor widths is a reasonable approximation. Thus, it is necessary to optimize the delay only as a function of the ratio of the W/L of the p-channel transistors to the W/L of the n-channel transistors.

If a number of gates are in series, with no memory or local feedback (i.e., acyclic logic), then the average of the rise and the fall times determines the overall gate delay. This might not be the case for flip-flops or pipelined logic, but will be the case considered now. Also, only the load capacitance due to the gates being driven will be considered. If the junction capacitance was also considered, the conclusions would change very little, but the analysis would be more complicated. This extra complication is not felt to be merited by any substantial increase in insight.

The average of the rise and the fall times of the first inverter of Fig. 4.29 is

$$t_{AV} = 1.2C_L\frac{R_{eq\text{-}1} + R_{eq\text{-}2}}{2} \tag{4.75}$$

where C_L is given by (4.74) and $R_{eq\text{-}1}$ and $R_{eq\text{-}2}$ are given by (4.66) and (4.67), respectively. Therefore, we have

$$t_{AV} = 1.2 \times 2C_{ox}L(W_n + W_p)\frac{1}{2}\left[\frac{2.5}{\mu_n C_{ox}\frac{W_n}{L}(V_{DD} - V_{tn})} + \frac{2.5}{\mu_p C_{ox}\frac{W_p}{L}(V_{DD} - V_{tp})}\right] \tag{4.76}$$

Assuming $V_{DD} - V_{tn} = V_{DD} + V_{tp}$ and rearranging, we have

$$t_{AV} = \frac{3L^2}{(V_{DD} - V_{tn})\mu_n}\left(1 + \frac{W_p}{W_n}\right)\left(1 + \frac{\mu_n W_n}{\mu_p W_p}\right) \tag{4.77}$$

To find the optimum W_p/W_n to minimize the average of the rise and fall times, we can differentiate (4.77) with respect to W_p/W_n, set the result equal to zero, and solve for W_p/W_n. Continuing, after some algebra, we have

$$\frac{\partial t_{AV}}{\partial(W_p/W_n)} = \frac{3L^2}{(V_{DD} - V_{tn})\mu_n}\left[1 - \frac{\mu_n}{\mu_p}\left(\frac{W_n}{W_p}\right)^2\right] \tag{4.78}$$

Setting (4.78) equal to zero gives the optimum (W_p / W_n) as

$$\left(\frac{W_p}{W_n}\right)_{opt} = \sqrt{\frac{\mu_n}{\mu_p}} \tag{4.79}$$

For example, if $\mu_n/\mu_p = 2.5$ the optimum ratio for W_p/W_n is 1.58. This was the reason for the earlier statement in Section 4 that a ratio of W_p/W_n equal to 1.5 is usually a reasonable choice. For this choice we have

$$t_{AV} = \frac{3L^2}{(V_{DD} - V_{tn})\mu_n}\left[1 + \sqrt{\frac{\mu_n}{\mu_p}}\right]^2 \tag{4.80}$$

It is interesting to see how the average rise and fall time increases for the special cases $W_p/W_n = 1$ and $W_p/W_n = \mu_n/\mu_p$. For $W_p/W_n = 1$ and using (4.77), we have

$$t_{AV} = \frac{6L^2}{(V_{DD} - V_{tn})\mu_n}\left(1 + \frac{\mu_n}{\mu_p}\right) \tag{4.81}$$

Also, for $W_p / W_n = \mu_n/\mu_p$, we have

$$t_{AV} = \frac{6L^2}{(V_{DD} - V_{tn})\mu_n}\left(1 + \frac{\mu_n}{\mu_p}\right) \tag{4.82}$$

which is identical to (4.81). For example, if $\mu_n/\mu_p = 2.5$, a typical value, using (4.80), (4.81), and (4.82), we see the increase in the average of the rise and fall times is 5%. This very small increase indicates that though an optimum exists, taking $W_p/W_n = 1$ would cause very little penalty and would save on space. This choice is often used except in more critical designs.

It should be emphasized that the preceding analysis ignored any wiring capacitance or external capacitances that do not scale with the transistor widths. When these load capacitances dominate, taking the p-channel wider definitely improves the rise time and therefore the average of the rise and fall times. In this extreme, designers often take $W_p/W_n = 2$ or 3. As a compromise, for when the load of the gate is not known, a reasonable choice again might be $W_p/W_n = 1.5$.

The constant term in equation (4.80), that is the term

$$T_{proc} = \frac{L^2}{(V_{DD} - V_{tn})\mu_n} \tag{4.83}$$

is independent of geometry and dependent only on the technology. It is a good figure of merit when comparing two different technologies. Also, it makes it obvious how important short-channel lengths are for high-speed operation.

Power Dissipation

As has been mentioned many times previously," the major reason for the popularity of CMOS logic is that traditional gates have no d.c. power dissipation when their outputs are not changing. However, whenever the output of a CMOS gate changes, there is power dissipated in the gate transistors. The primary reason for this power dissipation is the movement of charge required to charge or discharge external load capacitances and internal parasitic capacitances. In addition, for input signals with finite rise and fall times, it is possible that d.c. paths exist between the power supply and ground temporarily while the output is changing. This is especially true for large inverters used as buffers.

The power dissipated by an inverter charging or discharging a load capacitor is particularly easy to calculate. Consider the inverter of Fig. 4.30 that has a square-wave input of frequency f_{clk} and a load capacitance C_L. Each time the output changes from a "0" to a "1", the load capacitor is charged from 0 V to V_{DD} by the n-channel transistor. The energy dissipated in the n-channel transistor during this time is equal to

$$E_n = \frac{C_L V_{DD}^2}{2} \tag{4.84}$$

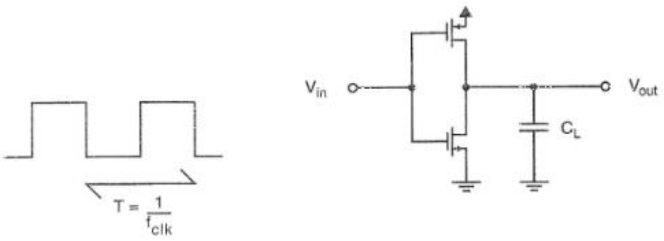


Figure 4.30 A CMOS inverter with a square-wave input (at a frequency f_{clk}).

Similarly, the energy dissipated in the n-channel transistor when the inverter output changes from a "1" to a "0" is

$$E_p = \frac{C_L V_{DD}^2}{2} \tag{4.85}$$

Since the output changes from a "0" to a "1" and back to a "0" each period (T) of the input, the total energy dissipated each period is equal to

$$E_T = C_L V_{DD}^2 \tag{4.86}$$

The average power dissipated during a period is the total energy dissipated divided by T. Thus, the average power dissipation due to the dynamic charging and discharging of the load capacitance is

$$P_{dyn\text{-}avg} = \frac{C_L V_{DD}^2}{T} = C_L V_{DD}^2 f_{clk} \tag{4.87}$$

Thus, the average power dissipation is proportional to the clock frequency. At higher clock frequencies, more power is proportionally dissipated. Also, the power dissipated is very sensitive to the power-supply voltage. Decreasing the power-supply voltage from 5 to 3.3 V decreases the power dissipation by approximately one-half.

A CMOS gate continuously changing at maximum speed dissipates a similar amount of power as an NMOS gate running at a similar speed. This would seem to contradict the stated superiority of CMOS logic with respect to power dissipation. *Luckily, most modern chips have tens of thousands or even millions of gates that do not change in any given clock cycle.* For example, in a computer chip, most of the memory and registers are unchanging in any given clock cycle. The gates that do not change do not dissipate power and the overall dissipation is

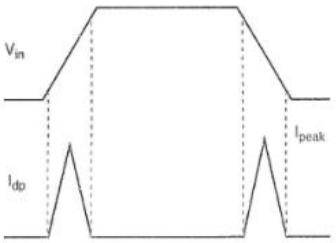


Figure 4.31 The input voltage waveform and the direct-path current of a CMOS inverter when the input voltage has finite rise and fall times.

much less. However, for the particular case of a CMOS gate changing continuously at full speed, the designer might consider logic circuits that have d.c. power dissipation, if smaller size or greater speed results. The overall difference of the power dissipation of the total chip will be small for this case.

There is additional power dissipated over and above that given by (4.87). This additional component is due to that fact that during transitions there is some d.c. current going through both the p-channel and n-channel transistors at the same time. This current is often called *direct-path current*. This additional power is usually less than 20% of that due to charging and discharging parasitic capacitances (Veendrick, 1984), but can be substantial, particularly if the input changes slowly.

This power dissipation can be estimated assuming the area under a plot of the direct-path current of an inverter can be approximated by a triangular wave (Rabaey, 1996). For example, consider the plots of the input voltage and direct-path current of a CMOS inverter shown in Fig. 4.31. Once the inverter threshold voltage, V_{th}, has been found, then the peak direct-path current is found from

$$I_{peak} = \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)_1 (V_{th} - V_{tn})^2 \tag{4.88}$$

where it is assumed the n-channel drive transistor is Q_1. The energy dissipated each period is then given by

$$E_{dp} = V_{DD}\left(\frac{I_{peak}t_r}{2} + \frac{I_{peak}t_f}{2}\right) = V_{DD}I_{peak}\left(\frac{t_r + t_f}{2}\right) \tag{4.89}$$

The average power dissipated is the energy dissipated per period divided by the time of the period. Thus, we have

$$P_{dp\text{-}avg} = \frac{1}{T}V_{DD}I_{peak}\left(\frac{t_r + t_f}{2}\right) = V_{DD}I_{peak}\left(\frac{t_r + t_f}{2}\right)f_{clk} \tag{4.90}$$

CMOS Gate Design

There are many different design methodologies for realizing logic functions using CMOS transistors. In Chapter 1, we were introduced to traditional CMOS gates. These will be dealt with in greater detail in this section. Different types of CMOS logic families are now gaining in popularity. These newer families tend to minimize the number of p-channel transistors required and often use dynamic techniques such as precharging outputs to a high voltage before evaluation. In addition, they often operate on fully differential signals. That is, every signal is propagated along with its complement. These advanced techniques will be covered in Chapter 9.

Traditional Logic Design

The traditional approach to CMOS logic design is to realize an n-channel driver circuit the same as one would for pseudo-NMOS logic, although the transistor sizing might be done differently. The load is then realized as a *complementary* p-channel network.

As an example of a moderately complicated CMOS logic circuit, consider the realization of a full adder. A full adder is one of the most often used building blocks in arithmetic circuits. These circuits are often in the critical path of microcomputers and digital signal-processing circuits; thus, a lot of effort has been spent on optimizing them. A full adder has three inputs (A, B, and C) and two outputs (Sum and Carry). The functions realized by a full adder are

$$\text{Sum} = A \oplus B \oplus C = ABC + A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C \tag{4.94}$$

and

$$\text{Carry} = AB + AC + BC = AB + C(A + B) \tag{4.95}$$

In words, the *sum* function realizes the three-input *exclusive-or* function; its output is a "1" if an odd number of inputs are "1"s. The *carry* is a "1" if two or more inputs are "1"s. A logic diagram that realizes both functions is shown in Fig. 4.32. Since there are only four inverting gates in the diagram, the function can be realized by four CMOS gates; two of these gates are simple inverters.

The first complex gate can realize the same function as logic gates *a, b, c,* and *d*. The n-channel drive network, which is arrived at using the same procedures as described previously for pseudo-NMOS logic gates, is shown in Fig. 4.33. The complementary p-channel network,

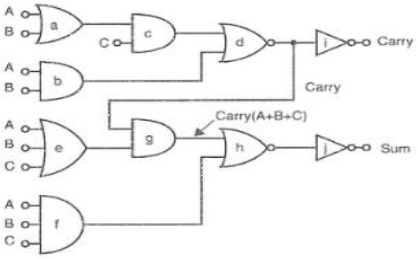


Figure 4.32 A logic diagram that realizes the full adder.

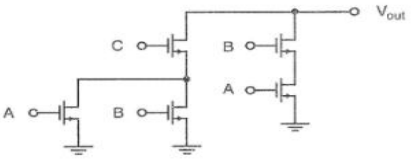


Figure 4.33 The n-channel drive network for realizing the gates *a, b, c,* and *d* of Fig. 4.32.

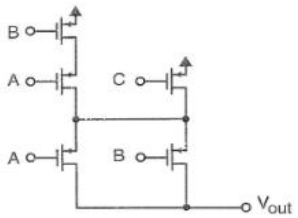


Figure 4.34 The complementary p-channel network to the n-channel network of Fig. 4.33.

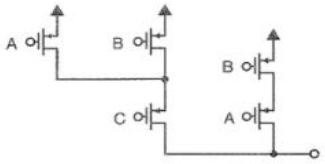


Figure 4.35 An equivalent p-channel network to the p-channel network of Fig. 4.34.

derived using the techniques of Section 1.2, is shown in Fig. 4.34. Figure 4.34 could be simplified somewhat by noting that the p-channel network of Fig. 4.35 realizes the same function and only has two transistors in series worst-case. This and similar simplifications are often possible; unfortunately, a formal procedure to identify and implement these simplifications has not been found.

In a similar manner, the CMOS realizations of logic gates *e, f, g,* and *h* can be found and simplified. The realization of inverters *i* and *j* are obvious. The complete CMOS realization of a full-adder cell is shown in Fig. 4.36.

There is no *exact* optimum procedure for choosing the W/Ls of transistors used to realize traditional CMOS logic gates. The gates will always function correctly independent of the sizes chosen. However, there are a few guidelines that are often followed. These guidelines mimic the procedures used for choosing the W/Ls of transistors used in NMOS gates. Basically, the transistors are taken wider if it is possible they might be in series with a larger number of transistors. However, in traditional CMOS circuits, the Ws are seldom taken larger than 10 to 20 μm to conserve space. Also, the W/Ls of p-channel networks are sometimes taken larger than the W/Ls of n-channel networks to give more equal rise and fall times, given the unequal mobilities of electrons and holes.

When the designer has the choice, *nand* gates are preferable in traditional CMOS design compared to *nor* gates. This is because *nand* gates have the n-channel transistors in series and

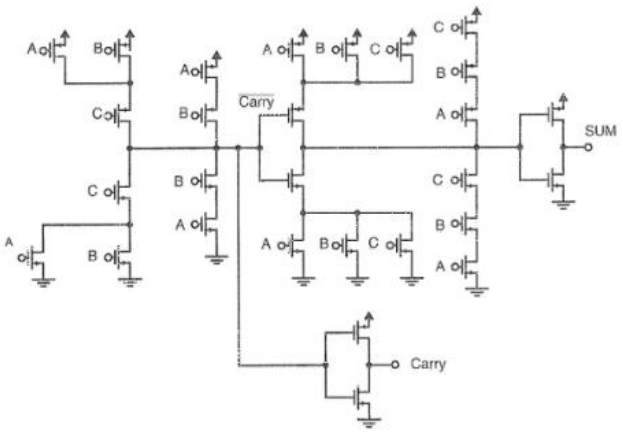


Figure 4.36 A CMOS realization of the full-adder function.

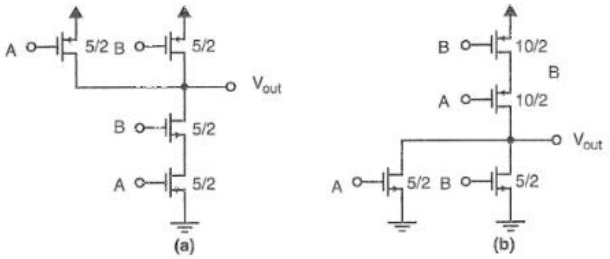


Figure 4.37 CMOS *nand* (a) and *nor* (b) gates along with typical W/Ls.

the p-channel transistors in parallel, where the opposite is true for *nor* gates. When designing for almost-equal rise and fall times, *nand* gates have more equally sized transistors; in *nor* gates the W/Ls of the p-channel transistors get quite large. For example, Fig. 4.37a and b shows typical realizations of two input *nand* and *nor* gates with reasonable sizes. Notice the smaller W/Ls of the *nand* gate.

Despite the more reasonable W/Ls, the worst case rise time of the *nand* gate is the same as that of the *nor* gate when driving large capacitive loads.

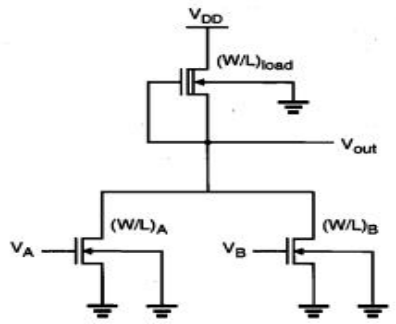
Given the preference for the *and* function when formal logic design techniques are used, functions that result in *and-or* functions are usually preferable to those that result in *or-and* functions, as the former is traditionally realized by two levels of *nand* gates, whereas the latter is normally realized by two levels of *nor* gates. This means grouping the "1"s is often preferable to grouping the "0"s when using Karnaugh map realization techniques for logic minimization (Mano,1984).

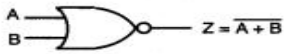
This generalization only holds when simple *nand* or *nor* gates are used. When complex gates are used, one should check all possibilities.

MOS Logic Circuits with Depletion nMOS Loads

Two-Input NOR Gate

The first circuit to be examined in this section is the two-input NOR gate. The circuit diagram, the logic symbol, and the corresponding truth table of the gate are given in Fig. 7.2. The Boolean OR operation is performed by the parallel connection of the two enhancement-type nMOS driver transistors. If the input voltage V_A *or* the input voltage V_B is equal to the logic-high level, the corresponding driver transistor turns on and provides a conducting path between the output node and the ground. Hence, the output voltage becomes low. In this case, the circuit operates like a depletion-load inverter with respect to its static behavior. A similar result is achieved when both V_A and V_B are high, in which case two parallel conducting paths are created between the output node and the ground. If, on the other hand, both V_A and V_B are low, both driver transistors remain cut-off. The output node voltage is pulled to a logic-high level by the depletion-type nMOS load transistor.





V_A	V_B	V_{out}
low	low	high
low	high	low
high	low	low
high	high	lower

Figure 7.2. A two-input depletion-load NOR gate, its logic symbol, and the corresponding truth table. Note that the substrates of all transistors are connected to ground.

The DC analysis of the circuit can be simplified significantly by considering the structural similarities between this circuit and the simple nMOS depletion-load inverter. In the following, the calculation of output low and output high voltages will be examined.

Calculation of V_{OH}

When both input voltages V_A and V_B are lower than the corresponding driver threshold voltage, the driver transistors are turned off and conduct no drain current. Consequently, the load device, which operates in the linear region, also has zero drain current. In particular, its linear region current equation becomes

$$I_{D,load} = \frac{k_{n,load}}{2} \cdot \left[2\left|V_{T,load}(V_{OH})\right| \cdot (V_{DD} - V_{OH}) - (V_{DD} - V_{OH})^2 \right] = 0 \qquad (7.1)$$

The solution of this equation gives $V_{OH} = V_{DD}$.

Calculation of V_{OL}

To calculate the output low voltage V_{OL}, we must consider three different cases, i.e., three different input voltage combinations, which produce a conducting path from the output node to the ground. These cases are

(i) $V_A = V_{OH}$ $V_B = V_{OL}$
(ii) $V_A = V_{OL}$ $V_B = V_{OH}$
(iii) $V_A = V_{OH}$ $V_B = V_{OH}$

For the first two cases, (i) and (ii), the NOR circuit reduces to a simple nMOS depletion-load inverter. Assuming that the threshold voltages of the two enhancement-type driver transistors are identical ($V_{T0,A} = V_{T0,B} = V_{T0}$), the driver-to-load ratio of the corresponding inverter can be found as follows. In case (i), where the driver transistor A is on, the ratio is

$$k_R = \frac{k_{driver,A}}{k_{load}} = \frac{k'_{n,driver}\left(\frac{W}{L}\right)_A}{k'_{n,load}\left(\frac{W}{L}\right)_{load}} \tag{7.2}$$

In case (ii), where the driver transistor B is on, the ratio is

$$k_R = \frac{k_{driver,B}}{k_{load}} = \frac{k'_{n,driver}\left(\frac{W}{L}\right)_B}{k'_{n,load}\left(\frac{W}{L}\right)_{load}} \tag{7.3}$$

The output low voltage level V_{OL} in both cases is found by using (5.54), as follows:

$$V_{OL} = V_{OH} - V_{T0} - \sqrt{(V_{OH} - V_{T0})^2 - \left(\frac{k_{load}}{k_{driver}}\right) \cdot \left|V_{T,load}(V_{OL})\right|^2} \tag{7.4}$$

Note that if the (W/L) ratios of both drivers are identical, i.e., $(W/L)_A = (W/L)_B$, the output low voltage (V_{OL}) values calculated for case (i) and case (ii) will be identical.

In case (iii), where both driver transistors are turned on, the saturated load current is the sum of the two linear-mode driver currents.

$$I_{D,load} = I_{D,driverA} + I_{D,driverB} \tag{7.5}$$

$$\begin{aligned} \frac{k_{load}}{2}\left|V_{T,load}(V_{OL})\right|^2 = \; & \frac{k_{driver,A}}{2}\left[2(V_A - V_{T0})V_{OL} - V_{OL}^2\right] \\ + \; & \frac{k_{driver,B}}{2}\left[2(V_B - V_{T0})V_{OL} - V_{OL}^2\right] \end{aligned} \tag{7.6}$$

Since the gate voltages of both driver transistors are equal ($V_A = V_B = V_{OH}$), we can devise an *equivalent* driver-to-load ratio for the NOR structure:

For the first two cases, (i) and (ii), the NOR circuit reduces to a simple nMOS depletion-load inverter. Assuming that the threshold voltages of the two enhancement-type driver transistors are identical ($V_{T0,A} = V_{T0,B} = V_{T0}$), the driver-to-load ratio of the corresponding inverter can be found as follows. In case (i), where the driver transistor A is on, the ratio is

$$k_R = \frac{k_{driver,A}}{k_{load}} = \frac{k'_{n,driver}\left(\frac{W}{L}\right)_A}{k'_{n,load}\left(\frac{W}{L}\right)_{load}} \tag{7.2}$$

In case (ii), where the driver transistor B is on, the ratio is

$$k_R = \frac{k_{driver,B}}{k_{load}} = \frac{k'_{n,driver}\left(\frac{W}{L}\right)_B}{k'_{n,load}\left(\frac{W}{L}\right)_{load}} \tag{7.3}$$

The output low voltage level V_{OL} in both cases is found by using (5.54), as follows:

$$V_{OL} = V_{OH} - V_{T0} - \sqrt{(V_{OH} - V_{T0})^2 - \left(\frac{k_{load}}{k_{driver}}\right) \cdot \left|V_{T,load}(V_{OL})\right|^2} \tag{7.4}$$

Note that if the (W/L) ratios of both drivers are identical, i.e., $(W/L)_A = (W/L)_B$, the output low voltage (V_{OL}) values calculated for case (i) and case (ii) will be identical.

In case (iii), where both driver transistors are turned on, the saturated load current is the sum of the two linear-mode driver currents.

$$I_{D,load} = I_{D,driverA} + I_{D,driverB} \tag{7.5}$$

$$\begin{aligned} \frac{k_{load}}{2}\left|V_{T,load}(V_{OL})\right|^2 = \; & \frac{k_{driver,A}}{2}\left[2(V_A - V_{T0})V_{OL} - V_{OL}^2\right] \\ + \; & \frac{k_{driver,B}}{2}\left[2(V_B - V_{T0})V_{OL} - V_{OL}^2\right] \end{aligned} \tag{7.6}$$

Since the gate voltages of both driver transistors are equal ($V_A = V_B = V_{OH}$), we can devise an *equivalent* driver-to-load ratio for the NOR structure:

Generalized NOR Structure with Multiple Inputs

At this point, we can expand our analysis to generalized n-input NOR gates, which consist of n parallel driver transistors, as shown in Fig. 7.3. Note that the combined current I_D in this circuit is supplied by the driver transistors which are turned on, i.e., transistors which have gate voltages higher than the threshold voltage V_{T0}.

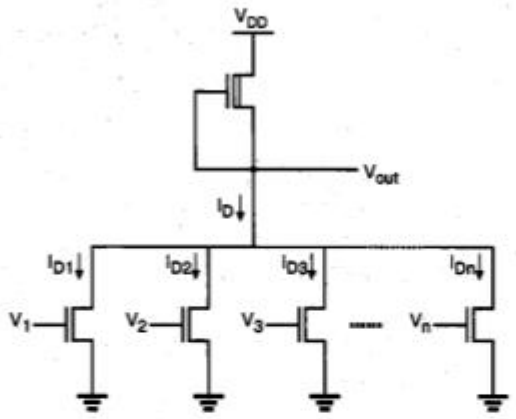


Figure 7.3. Generalized n-input NOR gate.

The combined pull-down current can then be expressed as follows:

$$I_D = \sum_{k(on)} I_{D,k} = \begin{cases} \displaystyle\sum_{k(on)} \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)_k \left[2\left(V_{GS,k} - V_{T0}\right)V_{out} - V_{out}^2\right] & \textit{linear} \\ \displaystyle\sum_{k(on)} \frac{\mu_n C_{ox}}{2}\left(\frac{W}{L}\right)_k \left(V_{GS,k} - V_{T0}\right)^2 & \textit{saturation} \end{cases} \tag{7.10}$$

Assuming that the input voltages of all driver transistors are identical,

$$V_{GS,k} = V_{GS} \qquad \textit{for} \qquad k = 1, 2, \ldots, n \tag{7.11}$$

the pull-down current expression can be rewritten as

$$I_D = \begin{cases} \dfrac{\mu_n C_{ox}}{2}\left(\displaystyle\sum_{k(on)}\left(\frac{W}{L}\right)_k\right)\left[2\left(V_{GS} - V_{T0}\right)V_{out} - V_{out}^2\right] & \textit{linear} \\ \dfrac{\mu_n C_{ox}}{2}\left(\displaystyle\sum_{k(on)}\left(\frac{W}{L}\right)_k\right)\left(V_{GS} - V_{T0}\right)^2 & \textit{saturation} \end{cases} \tag{7.12}$$

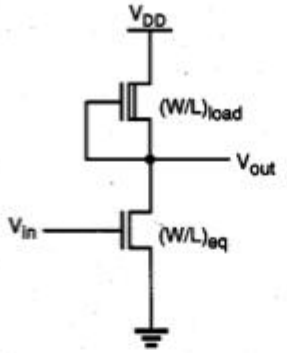


Figure 7.4. Equivalent inverter circuit corresponding to the *n*-input NOR gate.

Thus, the multiple-input NOR gate can also be reduced to an equivalent inverter, shown in Fig. 7.4, for static analysis. The (*W*/*L*) ratio of the driver transistor here is

$$\left(\frac{W}{L}\right)_{equivalent} = \sum_{k(on)} \left(\frac{W}{L}\right)_k \tag{7.13}$$

Note that the source terminals of all enhancement-type nMOS driver transistors in the NOR gate are connected to ground. Thus, the drivers do not experience any substrate-bias effect. The depletion-type nMOS load transistor, however, is subject to substrate-bias effect, since its source is connected to the output node, and its source-to-substrate voltage is $V_{SB} = V_{out}$.

Transient Analysis of NOR Gate

Figure 7.5 shows the two-input NOR (NOR2) gate with all of its relevant parasitic device capacitances. As in the inverter case, we can combine the capacitances seen in Fig. 7.5 into one lumped capacitance, connected between the output node and the ground. The value of this combined load capacitance, C_{load}, can be found as

$$C_{load} = C_{gd,A} + C_{gd,B} + C_{gd,load} + C_{db,A} + C_{db,B} + C_{sb,load} + C_{wire} \tag{7.14}$$

Note that the output load capacitance given in (7.14) is valid for simultaneous as well as for single-input switching, i.e., the load capacitance C_{load} will be present at the output node even if only one input is active and all other inputs are low. This fact must be taken into account in calculations using the inverter equivalent of the NOR gate. The load capacitance at the output node of the *equivalent* inverter corresponding to a NOR gate is always *larger* than the total lumped load capacitance of an actual inverter with the same dimensions. Hence, while the static (DC) behaviors of the NOR gate and the inverter are

essentially equivalent in this case, the actual transient response of the NOR gate will be slower than that of the inverter.

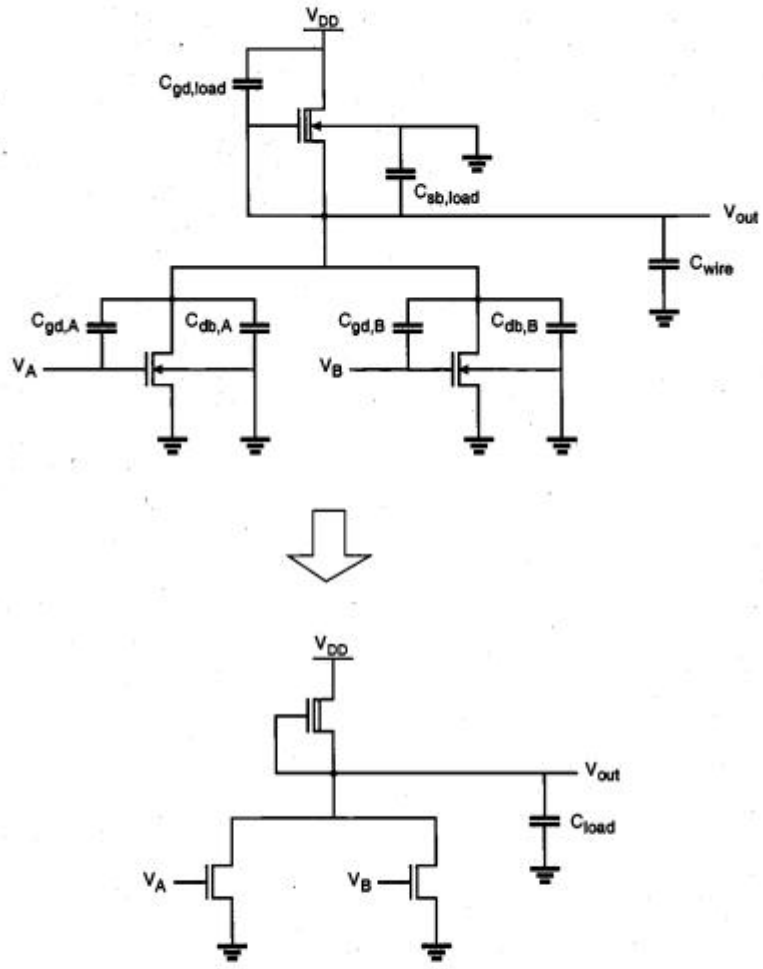


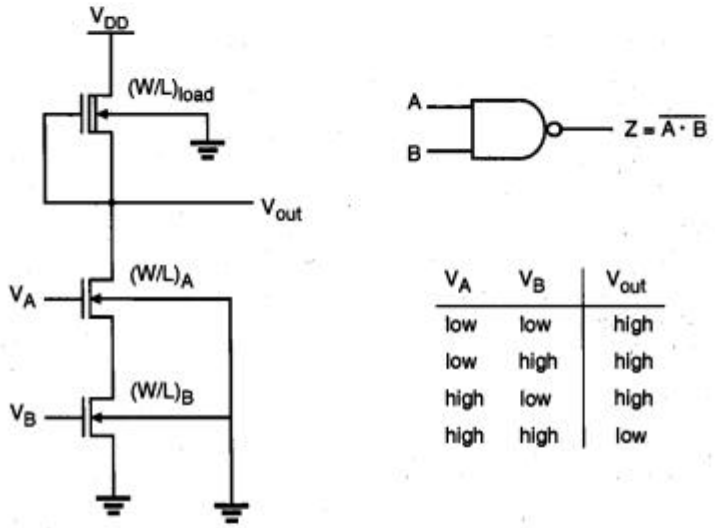
Figure 7.5. Parasitic device capacitances in the NOR2 gate and the lumped equivalent load capacitance. The gate-to-source capacitances of the driver transistors are included in the load of the previous stages driving the inputs A and B.

Two-Input NAND Gate

Next, we will examine the two-input NAND (NAND2) gate. The circuit diagram, the logic symbol, and the corresponding truth table of the gate are given in Fig. 7.6. The Boolean AND operation is performed by the series connection of the two enhancement-type nMOS driver transistors. There is a conducting path between the output node and the

ground only if the input voltage V_A *and* the input voltage V_B are equal to logic-high, i.e., only if both of the series-connected drivers are turned on. In this case, the output voltage will be low, which is the complemented result of the AND operation. Otherwise, either one or both of the driver transistors will be off, and the output voltage will be pulled to a logic-high level by the depletion-type nMOS load transistor.

Figure 7.6 shows that all transistors except the one closest to the ground are subject to substrate-bias effect, since their source voltages are larger than zero. We have to consider this fact in detailed calculations. For all of the three input combinations which produce a logic-high output voltage, the corresponding V_{OH} value can easily be found as $V_{OH} = V_{DD}$. The calculation of the logic-low voltage V_{OL}, on the other hand, requires a closer investigation.



V_A	V_B	V_{out}
low	low	high
low	high	high
high	low	high
high	high	low

Figure 7.6. A two-input depletion-load NAND gate, its logic symbol, and the corresponding truth table. Notice the substrate-bias effect for all nMOS transistors except one.

Consider the NAND2 gate with both of its inputs equal to V_{OH}, as shown in Fig. 7.7. It can easily be seen that the drain currents of all transistors in the circuit are equal to each other.

$$I_{D,load} = I_{D,driverA} = I_{D,driverB} \tag{7.15}$$

$$\begin{aligned} \frac{k_{load}}{2}\left|V_{T,load}(V_{OL})\right|^2 &= \frac{k_{driver,A}}{2}\left[2\left(V_{GS,A} - V_{T,A}\right)V_{DS,A} - V_{DS,A}^2\right] \\ &= \frac{k_{driver,B}}{2}\left[2\left(V_{GS,B} - V_{T,B}\right)V_{DS,B} - V_{DS,B}^2\right] \end{aligned} \tag{7.16}$$

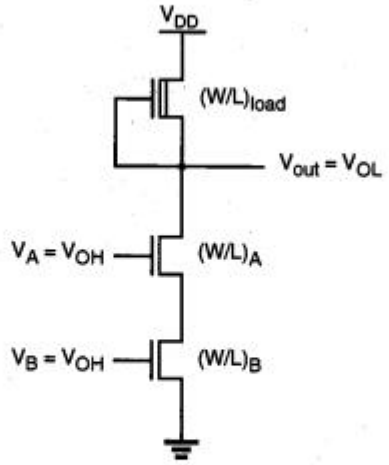


Figure 7.7. The NAND2 gate with both of its inputs at logic-high level.

The gate-to-source voltages of both driver transistors can be assumed to be approximately equal to V_{OH}. Also, we may neglect, for simplicity, the substrate-bias effect for driver transistor A, and assume $V_{T,A} = V_{T,B} = V_{T0}$, since the source-to-substrate voltage of driver A is relatively low. The drain-to-source voltages of both driver transistors can then be solved from (7.16) as

$$V_{DS,A} = V_{OH} - V_{T0} - \sqrt{(V_{OH} - V_{T0})^2 - \left(\frac{k_{load}}{k_{driver,A}}\right) \cdot \left|V_{T,load}(V_{OL})\right|^2} \tag{7.17}$$

$$V_{DS,B} = V_{OH} - V_{T0} - \sqrt{(V_{OH} - V_{T0})^2 - \left(\frac{k_{load}}{k_{driver,B}}\right) \cdot \left|V_{T,load}(V_{OL})\right|^2} \tag{7.18}$$

Let the two driver transistors be identical, i.e., $k_{driver,A} = k_{driver,B} = k_{driver}$. Noting that the output voltage V_{OL} is equal to the sum of the drain-to-source voltages of both drivers, we obtain

$$V_{OL} \approx 2\left(V_{OH} - V_{T0} - \sqrt{(V_{OH} - V_{T0})^2 - \left(\frac{k_{load}}{k_{driver}}\right) \cdot \left|V_{T,load}(V_{OL})\right|^2} \right) \tag{7.19}$$

The following analysis gives a better and more accurate view of the operation of two series-connected driver transistors. Consider the two identical enhancement-type nMOS transistors with their gate terminals connected. At this point, the only simplifying

assumption will be $V_{T,A} = V_{T,B} = V_{T0}$. When both driver transistors are in the linear region, the drain currents can be written as

$$I_{D,A} = \frac{k_{driver}}{2}\left[2\left(V_{GS,A} - V_{T0}\right)V_{DS,A} - V_{DS,A}^2\right] \tag{7.20}$$

$$I_{D,B} = \frac{k_{driver}}{2}\left[2\left(V_{GS,B} - V_{T0}\right)V_{DS,B} - V_{DS,B}^2\right] \tag{7.21}$$

Since $I_{D,A} = I_{D,B}$, this current can also be expressed as

$$I_D = I_{D,A} = I_{D,B} = \frac{I_{D,A} + I_{D,B}}{2} \tag{7.22}$$

Using $V_{GS,A} = V_{GS,B} - V_{DS,B}$, (7.22) yields

$$I_D = \frac{k_{driver}}{4}\left[2\left(V_{GS,B} - V_{T0}\right)\left(V_{DS,A} + V_{DS,B}\right) - \left(V_{DS,A} + V_{DS,B}\right)^2\right] \tag{7.23}$$

Now let $V_{GS} = V_{GS,B}$ and $V_{DS} = V_{DS,A} + V_{DS,B}$. The drain-current expression can then be written as follows.

$$I_D = \frac{k_{driver}}{4}\left[2\left(V_{GS} - V_{T0}\right)V_{DS} - V_{DS}^2\right] \tag{7.24}$$

Thus, two nMOS transistors connected in series and with the same gate voltage behave like *one* nMOS transistor with $k_{eq} = 0.5\, k_{driver}$.

Generalized NAND Structure with Multiple Inputs

At this point, we expand our analysis to generalized n-input NAND gates, which consist of n series-connected driver transistors, as shown in Fig. 7.8. Neglecting the substrate-bias effect, and assuming that the threshold voltages of all transistors are equal to V_{T0}, the driver current I_D in the linear region can be derived as in Eq. (7.25) whereas I_D in saturation is taken as its extension.

$$I_D = \frac{\mu_n C_{ox}}{2}\left(\frac{1}{\sum_{k(on)}\frac{1}{\left(\frac{W}{L}\right)_k}}\right)\cdot\begin{cases}\left[2\left(V_{in} - V_{T0}\right)V_{out} - V_{out}^2\right] & \textit{linear}\\ \left(V_{in} - V_{T0}\right)^2 & \textit{saturation}\end{cases} \tag{7.25}$$

Hence, the (W/L) ratio of the equivalent driver transistor is

$$\left(\frac{W}{L}\right)_{equivalent} = \frac{1}{\sum_{k(on)} \frac{1}{\left(\frac{W}{L}\right)_k}} \tag{7.26}$$

If the series-connected transistors are identical, i.e., $(W/L)_1 = (W/L)_2 = \ldots = (W/L)$, the width-to-length ratio of the equivalent transistor becomes

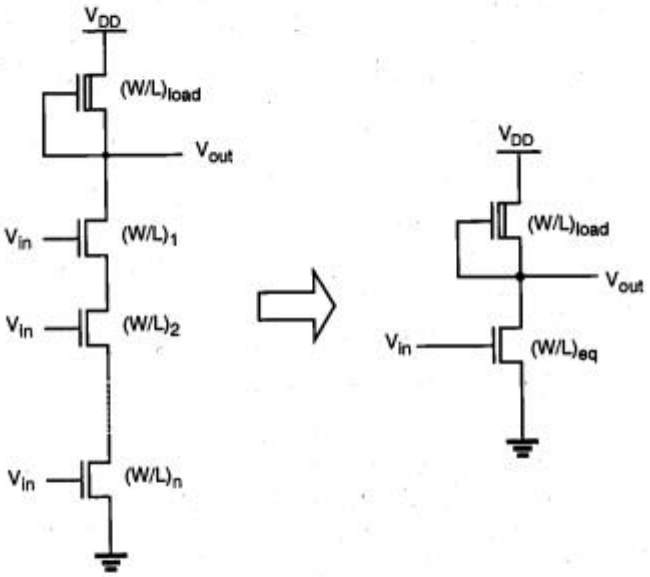


Figure 7.8. The generalized NAND structure and its inverter equivalent.

$$\left(\frac{W}{L}\right)_{equivalent} = \frac{1}{n}\left(\frac{W}{L}\right) \tag{7.27}$$

The NAND design strategy which emerges from this analysis is summarized as follows, for an n-input NAND. First, we determine the (W/L) ratios for an *equivalent inverter* that satisfies the required V_{OL} value. This gives us the driver transistor ratio $(W/L)_{driver}$ and the load transistor ratio $(W/L)_{load}$. Then, we set the (W/L) ratios of all NAND driver transistors as $(W/L)_1 = (W/L)_2 = \ldots = n\,(W/L)_{driver}$. This guarantees that the series structure consisting of n driver transistors has an equivalent (W/L) ratio of $(W/L)_{driver}$ when all inputs are logic-high.

For a two-input NAND gate, this means that each driver transistor must have a (W/L) ratio twice that of the equivalent inverter driver. If the area occupied by the depletion-type load transistor is negligible, the resulting NAND2 structure will occupy approximately four times the area occupied by the equivalent inverter which has the same static characteristics.

Transient Analysis of NAND Gate

Figure 7.9 shows a NAND2 gate with all parasitic device capacitances. As in the inverter case, we can combine the capacitances seen in Fig. 7.9 into one lumped capacitance, connected between the output node and the ground. The value of the lumped capacitance C_{load}, however, depends on the input voltage conditions.

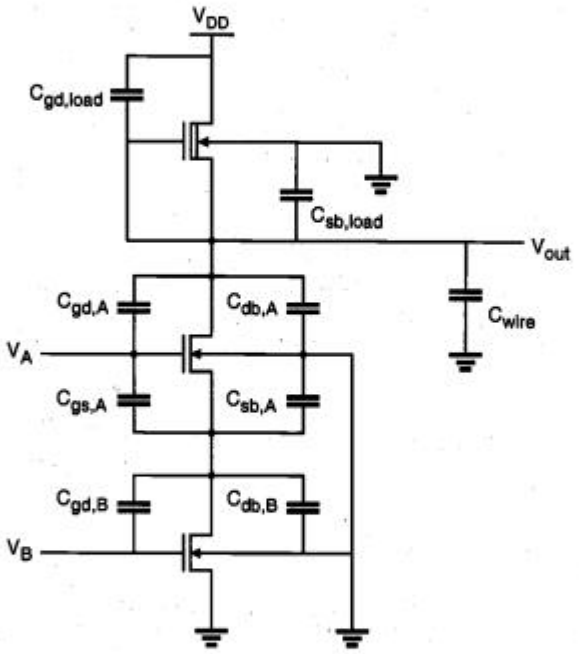


Figure 7.9. Parasitic device capacitances in the NAND2 gate.

Assume, for example, that the input V_A is equal to V_{OH} and the other input V_B is switching from V_{OH} to V_{OL}. In this case, both the output voltage V_{out} and the internal node voltage V_x will rise, resulting in

$$\begin{aligned} C_{load} = \; & C_{gd,load} + C_{gd,A} + C_{gd,B} + C_{gs,A} \\ & + C_{db,A} + C_{db,B} + C_{sb,A} + C_{sb,load} + C_{wire} \end{aligned} \tag{7.28}$$

Note that this value is quite conservative and fully reflects the internal node capacitances into the lumped output capacitance C_{load}. In reality, only a fraction of the internal node capacitance is reflected into C_{load}.

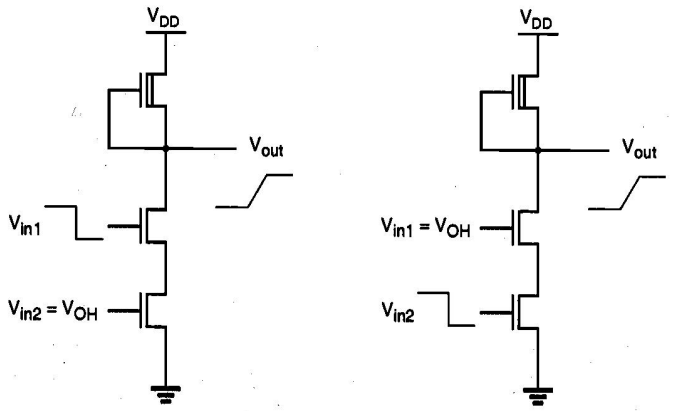
Now consider another case where V_B is equal to V_{OH} and V_A switches from V_{OH} to V_{OL}. In this case, the output voltage V_{out} will rise, but the internal node voltage V_x will remain low because the bottom driver transistor is on. Thus, the lumped output capacitance is

$$C_{load} = C_{gd,load} + C_{gd,A} + C_{db,A} + C_{sb,load} + C_{wire} \tag{7.29}$$

It should be noted that the load capacitance in this case is smaller than the load capacitance found in the previous case. Thus, it is expected that the high-to-low switching delay from signal B connected to the bottom transistor is larger than the high-to-low switching delay from signal A connected to the top transistor.

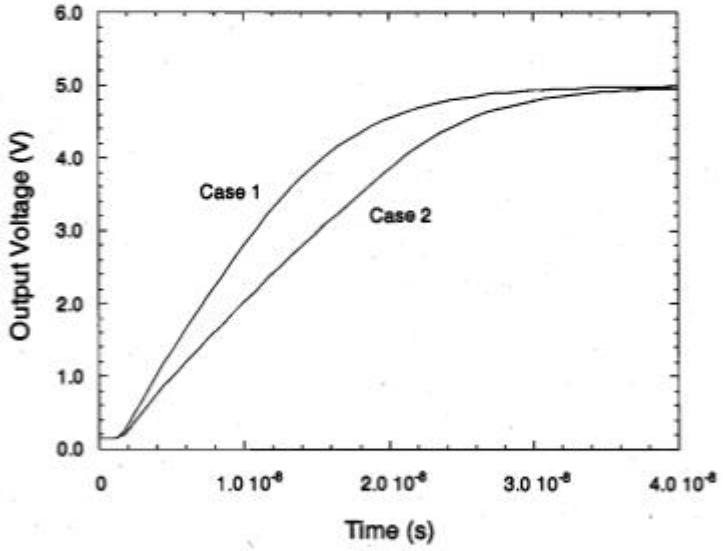
Example 7.1.

A depletion-load nMOS NAND2 gate is simulated with SPICE for the two different input switching events described above. The SPICE input file of the circuit is listed in the following. Note that the total capacitance between the intermediate node X and the ground is assumed to be half of the total capacitance appearing between the output node and the ground.



Case 1 Case 2

```
NAND2 circuit delay analysis
m1 3 1 0 0 mn w=5u l=1u
m2 4 2 3 0 mn w=5u l=1u
m3 5 4 4 0 mnd w=1u l=3u
c1 4 0 0.1p
cp 3 0 0.05p
vdd 5 0 dc 5.0
* case 1 (upper input switching from high to low)
vin1 2 0 dc pulse (5.0 0.0 1ns 1ns 2ns 40ns 50ns)
vin2 1 0 dc 5.0
* case 2 (lower input switching from high to low)
* vin1 2 0 dc 5.0
* vin2 1 0 dc pulse (5.0 0.0 1ns 1ns 2ns 40ns 50ns)
.model mn nmos (vto=1.0 kp=25u gamma=0.4)
.model mnd nmos (vto=-3.0 kp=25u gamma=0.4)
.tran 0.1ns 40ns
.print tran v(1) v(2) v(4)
.end
```



The simulated transient response of the NAND2 gate for both cases is plotted against time above. The time delay difference between the two cases is clearly visible. In fact, the propagation delay time in Case 2 is about 30% larger than that in Case 1, which proves that the input switching order has a significant influence on speed.

CMOS NOR2 (Two-Input NOR) Gate

The design and analysis of CMOS combinational logic circuits can be based on the basic principles developed for the nMOS depletion-load logic circuits in the previous section. Figure 7.10 shows the circuit diagram of a two-input CMOS NOR gate. Note that the circuit consists of a parallel-connected n-net and a series-connected complementary p-net. The input voltages V_A and V_B are applied to the gates of one nMOS and one pMOS transistor.

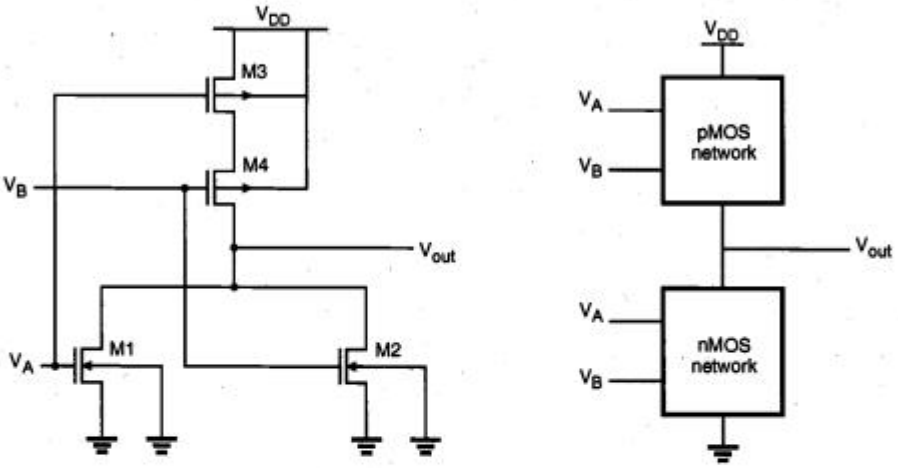


Figure 7.10. A CMOS NOR2 gate and its complementary operation: Either the nMOS network is on and the pMOS network is off, or the pMOS network is on and the nMOS network is off.

The complementary nature of the operation can be summarized as follows: When either one or both inputs are high, i.e., when the n-net creates a conducting path between the output node and the ground, the p-net is cut-off. On the other hand, if both input voltages are low, i.e., the n-net is cut-off, then the p-net creates a conducting path between the output node and the supply voltage V_{DD}. Thus, the dual or complementary circuit structure allows that, for any given input combination, the output is connected either to V_{DD} or to ground via a low-resistance path. A DC current path between the V_{DD} and ground is not established for any of the input combinations. This results in the fully complementary operation mode already examined for the simple CMOS inverter circuit.

The output voltage of the CMOS NOR2 gate will attain a logic-low voltage of $V_{OL} = 0$ and a logic-high voltage of $V_{OH} = V_{DD}$. For circuit design purposes, the switching threshold voltage V_{th} of the CMOS gate emerges as an important design criterion. We start our analysis of the switching threshold by assuming that both input voltages switch simultaneously, i.e., $V_A = V_B$. Furthermore, it is assumed that the device sizes in each block are identical, $(W/L)_{n,A} = (W/L)_{n,B}$ and $(W/L)_{p,A} = (W/L)_{p,B}$, and the substrate-bias effect for the pMOS transistors is neglected for simplicity.

By definition, the output voltage is equal to the input voltage at the switching threshold.

$$V_A = V_B = V_{out} = V_{th} \tag{7.30}$$

It is obvious that the two parallel nMOS transistors are saturated at this point, because $V_{GS} = V_{DS}$. The combined drain current of the two nMOS transistors is

$$I_D = k_n \left(V_{th} - V_{T,n}\right)^2 \tag{7.31}$$

Thus, we obtain the first equation for the switching threshold V_{th}.

$$V_{th} = V_{T,n} + \sqrt{\frac{I_D}{k_n}} \tag{7.32}$$

Examination of the p-net in Fig. 7.10 shows that the pMOS transistor M3 operates in the linear region, while the other pMOS transistor, M4, is in saturation for $V_{in} = V_{out}$. Thus,

$$I_{D3} = \frac{k_p}{2}\left[2\left(V_{DD} - V_{th} - \left|V_{T,p}\right|\right)V_{SD3} - V_{SD3}^2\right] \tag{7.33}$$

$$I_{D4} = \frac{k_p}{2}\left(V_{DD} - V_{th} - \left|V_{T,p}\right| - V_{SD3}\right)^2 \tag{7.34}$$

The drain currents of both pMOS transistors are identical, i.e., $I_{D3} = I_{D4} = I_D$. Thus,

$$V_{DD} - V_{th} - \left|V_{T,p}\right| = 2\sqrt{\frac{I_D}{k_p}} \tag{7.35}$$

This yields the second equation of the switching threshold voltage V_{th}. Combining (7.32) and (7.35), we obtain

$$V_{th}(\text{NOR2}) = \frac{V_{T,n} + \frac{1}{2}\sqrt{\frac{k_p}{k_n}}\left(V_{DD} - \left|V_{T,p}\right|\right)}{1 + \frac{1}{2}\sqrt{\frac{k_p}{k_n}}} \tag{7.36}$$

Now compare this expression with the switching threshold voltage of the CMOS inverter, which was derived in Chapter 5.

$$V_{th}(\text{INR}) = \frac{V_{T,n} + \sqrt{\frac{k_p}{k_n}}\left(V_{DD} - |V_{T,p}|\right)}{1 + \sqrt{\frac{k_p}{k_n}}} \tag{7.37}$$

If $k_n = k_p$ and $V_{T,n} = |V_{T,p}|$, the switching threshold of the CMOS inverter is equal to $V_{DD}/2$. Using the same parameters, the switching threshold of the NOR2 gate is

$$V_{th}(\text{NOR2}) = \frac{V_{DD} + V_{T,n}}{3} \tag{7.38}$$

which is not equal to $V_{DD}/2$. For example, when $V_{DD} = 5$ V and $V_{T,n} = |V_{T,p}| = 1$ V, the switching threshold voltages of the NOR2 gate and the inverter are

$$V_{th}(\text{NOR2}) = 2 \text{ V}$$

$$V_{th}(\text{INR}) = 2.5 \text{ V}$$

The switching threshold voltage of the NOR2 gate can also be obtained by using the equivalent-inverter approach. When both inputs are identical, the parallel-connected nMOS transistors can be represented by a single nMOS transistor with $2k_n$. Similarly, the series-connected pMOS transistors are represented by a single pMOS transistor with $k_p/2$. The resulting equivalent CMOS inverter is shown in Fig. 7.11.

Using the inverter switching threshold expression (7.37) for the equivalent inverter circuit, we obtain

$$V_{th}(\text{NOR2}) = \frac{V_{T,n} + \sqrt{\frac{k_p}{4k_n}}\left(V_{DD} - |V_{T,p}|\right)}{1 + \sqrt{\frac{k_p}{4k_n}}} \tag{7.39}$$

which is identical to (7.36).

From (7.36), we can easily derive simple design guidelines for the NOR2 gate. For example, in order to achieve a switching threshold voltage of $V_{DD}/2$ for simultaneous switching, we have to set $V_{T,n} = |V_{T,p}|$ and $k_p = 4\,k_n$.

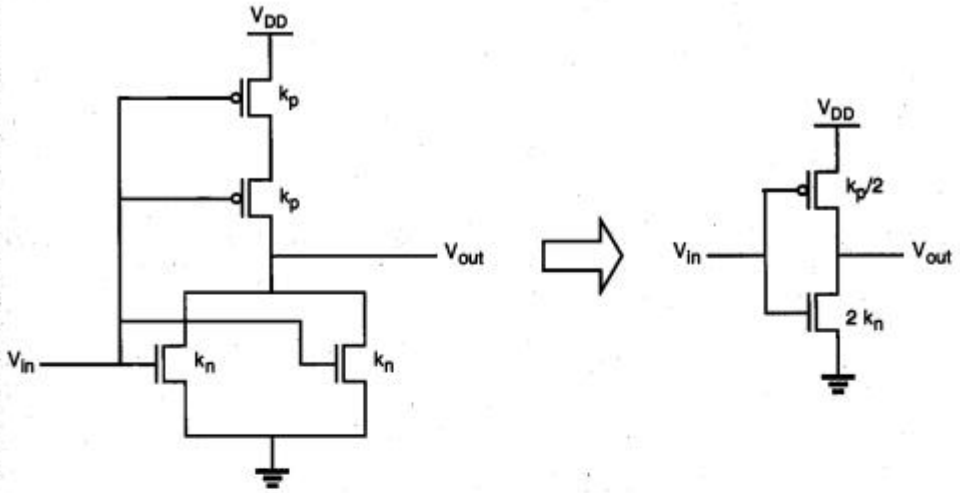


Figure 7.11. A CMOS NOR2 gate and its inverter equivalent.

Figure 7.12 shows the CMOS NOR2 gate with the parasitic device capacitances, the inverter equivalent, and the corresponding lumped output load capacitance. In the worst case, the total lumped load capacitance is assumed to be equal to the sum of all internal parasitic device capacitances seen in Fig. 7.12.

CMOS NAND2 (Two-Input NAND) Gate

Figure 7.13 shows a two-input CMOS NAND (NAND2) gate. The operating principle of this circuit is the exact dual of the CMOS NOR2 operation examined earlier. The n-net consisting of two series-connected nMOS transistors creates a conducting path between the output node and the ground only if both input voltages are logic-high, i.e., are equal to V_{OH}. In this case, both of the parallel-connected pMOS transistors in the p-net will be off. For all other input combinations, either one or both of the pMOS transistors will be turned on, while the n-net is cut-off, thus creating a current path between the output node and the power supply voltage.

Using an analysis similar to the one developed for the NOR2 gate, we can easily calculate the switching threshold for the CMOS NAND2 gate. Again, we will assume that the device sizes in each block are identical, with $(W/L)_{n,A} = (W/L)_{n,B}$ and $(W/L)_{p,A} = (W/L)_{p,B}$. The switching threshold for this gate is then found as

$$V_{th}(\text{NAND2}) = \frac{V_{T,n} + 2\sqrt{\frac{k_p}{k_n}}\left(V_{DD} - |V_{T,p}|\right)}{1 + 2\sqrt{\frac{k_p}{k_n}}} \tag{7.40}$$

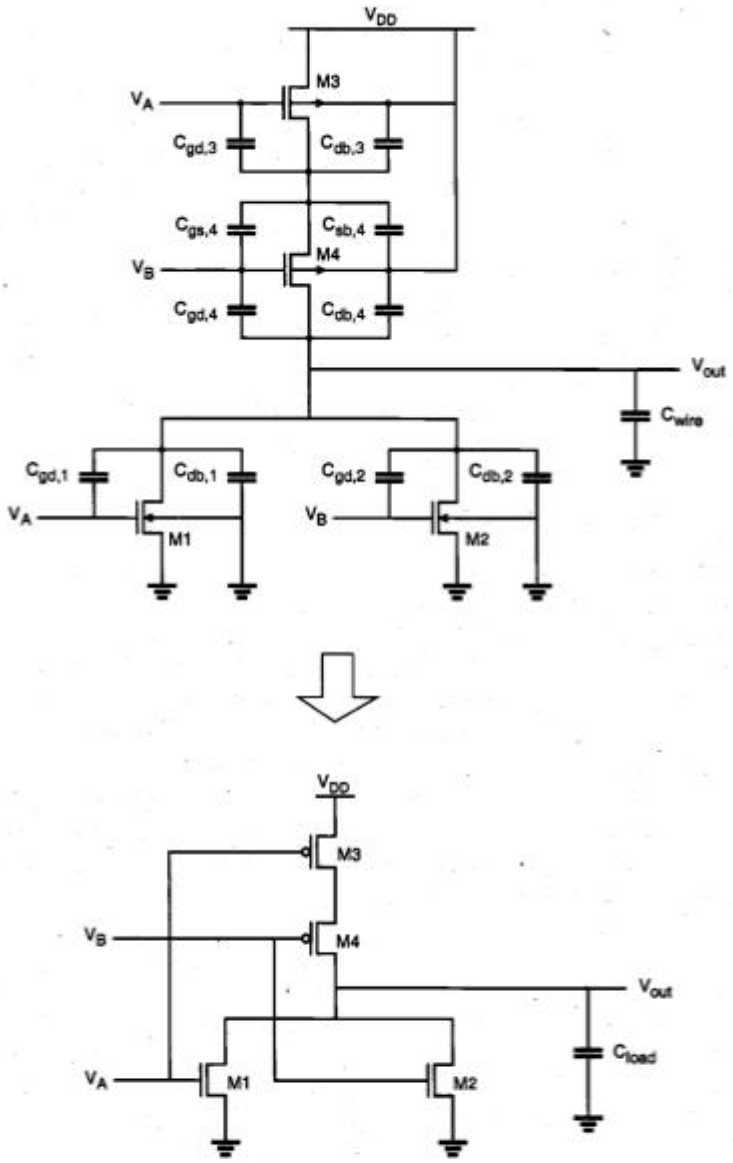


Figure 7.12. Parasitic device capacitances of the CMOS NOR2 circuit and the simplified equivalent with the lumped output load capacitance.

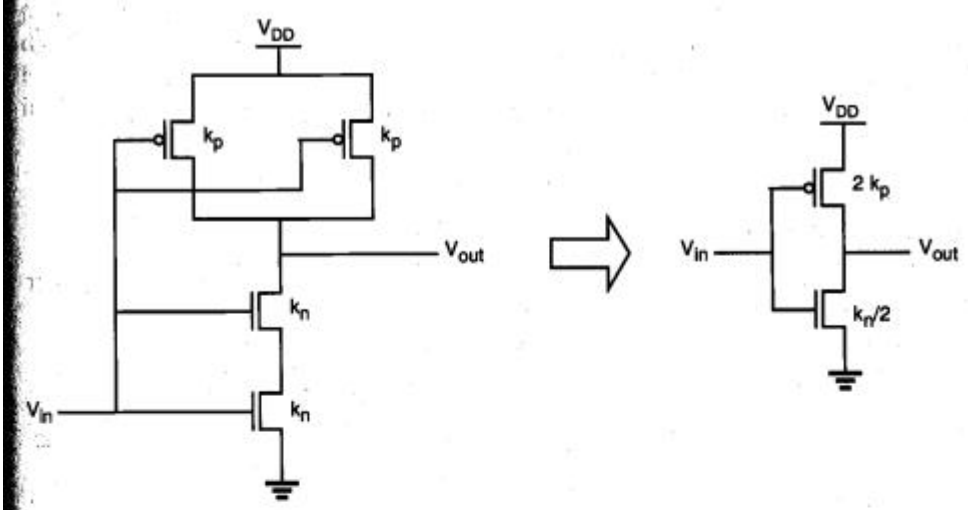


Figure 7.13. A CMOS NAND2 gate and its inverter equivalent.

As we can see from (7.40), a switching threshold voltage of $V_{DD}/2$ (for simultaneous switching) is achieved by setting $V_{T,n} = |V_{T,p}|$ and $k_n = 4\ k_p$ in the NAND2.

At this point, we can state the following observation about the area requirements of CMOS combinational logic gates. In comparison with equivalent nMOS depletion-load logic, the total number of transistors in CMOS gates is about twice the number of transistors in nMOS gates ($2n$ vs. $(n+1)$ for n inputs). The silicon area occupied by the CMOS gate, however, is not necessarily twice the area occupied by the nMOS depletion-load gate, since a significant portion of the silicon area must be reserved for signal routing and contacts in both cases. Thus, the area disadvantage of CMOS logic may actually be smaller than the simple transistor count suggests.

Layout of Simple CMOS Logic Gates

In the following, we will examine simplified layout examples for CMOS NOR2 and NAND2 gates. Figure 7.14 shows a sample layout of a CMOS NOR2 gate, using single-layer metal and single-layer polysilicon.

In this example, the p-type diffusion area for pMOS transistors and the n-type diffusion area for nMOS transistors are aligned in parallel to allow simple routing of the gate signals via two parallel polysilicon lines running vertically. Figure 7.15 shows the layout of a CMOS NAND2 gate, using the same basic layout principles as in the NOR2 layout example.

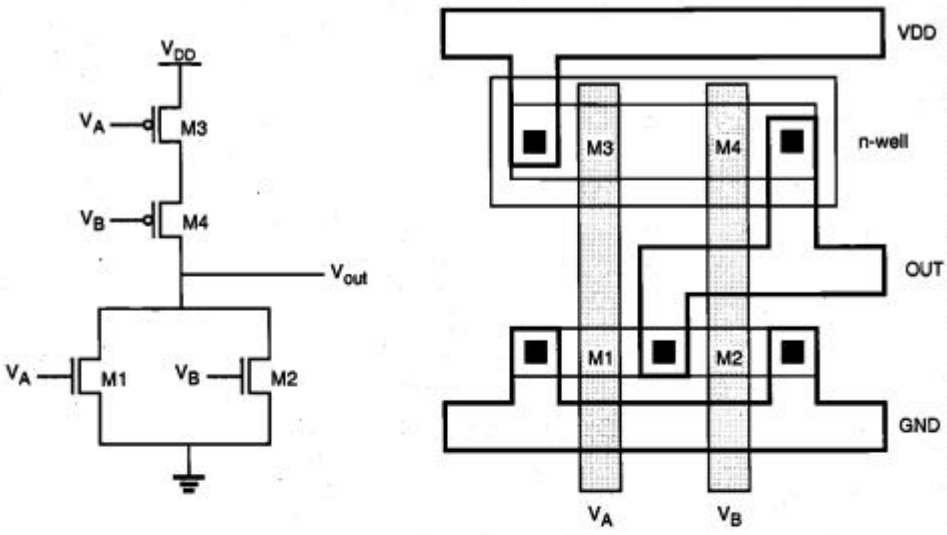


Figure 7.14. Sample layout of the CMOS NOR2 gate.

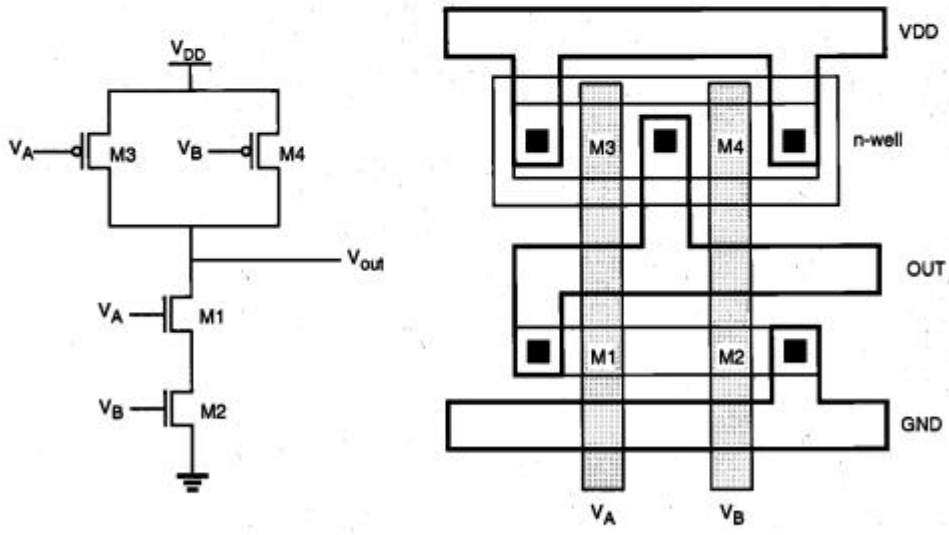


Figure 7.15. Sample layout of the CMOS NAND2 gate.

Finally, Fig. 7.16 shows a simplified (stick diagram) view of the CMOS NOR2 gate layout given in Fig. 7.14. Here, the diffusion areas are depicted by rectangles, the metal

connections and contacts are represented by solid lines and circles, respectively, and the polysilicon columns are represented by cross-hatched strips. The stick-diagram layout does not carry any information on the actual geometry relations of the individual features, but it conveys valuable information on the relative placement of the transistors and their interconnections.

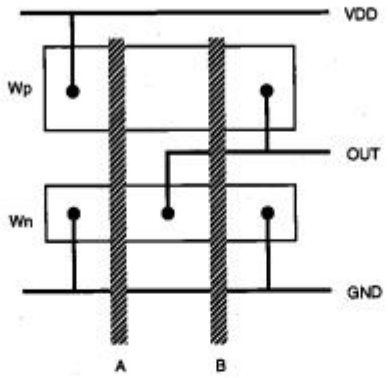


Figure 7.16. Stick-diagram layout of the CMOS NOR2 gate.

7.4. Complex Logic Circuits

To realize arbitrary Boolean functions of multiple input variables, the basic circuit structures and design principles developed for simple NOR and NAND gates in the previous sections can easily be extended to complex logic gates. The ability to realize complex logic functions using a small number of transistors is one of the most attractive features of nMOS and CMOS logic circuits.

Consider the following Boolean function as an example.

$$Z=\overline{A(D+E)+BC} \tag{7.41}$$

The nMOS depletion-load complex logic gate that is used to realize this function is shown in Fig. 7.17. Inspection of the circuit topology reveals the simple design principle of the pull-down network:

- OR operations are performed by parallel-connected drivers.
- AND operations are performed by series-connected drivers.
- Inversion is provided by the nature of MOS circuit operation.

The design principles stated here for individual inputs and corresponding driver transistors can also be extended to circuit sub-blocks, so that Boolean OR and AND operations can be performed in a nested circuit structure. Thus, we obtain a circuit topology which consists of series- and parallel-connected branches, as shown below.

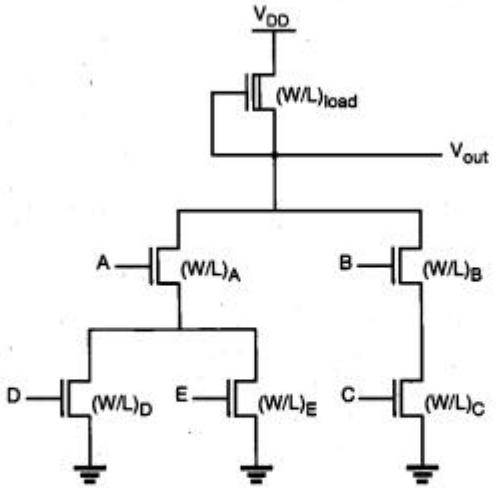


Figure 7.17. nMOS complex logic gate realizing the Boolean function given in (7.41).

In Fig. 7.17, the left nMOS driver branch consisting of three driver transistors is used to perform the logic function A(D + E), while the right-hand side branch performs the function BC. By connecting the two branches in parallel, and by placing the load transistor between the output node and the power supply voltage V_{DD}, we obtain the complex function given in (7.41). Each input variable is assigned to only one driver.

For the analysis and design of complex logic gates, we can employ the equivalent-inverter approach already used for the simpler NOR and NAND gates. It can be shown for the circuit in Fig. 7.17 that, if all input variables are logic-high, the equivalent-driver (*W/L*) ratio of the pull-down network consisting of five nMOS transistors is

$$\left(\frac{W}{L}\right)_{equivalent} = \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_B} + \frac{1}{\left(\frac{W}{L}\right)_C}} + \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_A} + \frac{1}{\left(\frac{W}{L}\right)_D + \left(\frac{W}{L}\right)_E}} \tag{7.42}$$

For calculating the logic-low voltage level V_{OL}, we have to consider various cases, since the value of V_{OL} actually depends on the number *and* the configuration of the conducting nMOS transistors in each case. All possible configurations are tabulated below. Each configuration is assigned a class number which reflects the total resistance of the current path from V_{out} node to ground.

A - D	Class 1
A - E	Class 1
B - C	Class 1
A - D - E	Class 2

A - D - B - C	Class 3
A - E - B - C	Class 3
A - D - E - B - C	Class 4

Assuming that all driver transistors have the same (W/L) ratio, a Class 1 path such as (B-C) has the highest series resistance, followed by Class 2, Class 3, etc. Consequently, the logic-low voltage levels corresponding to each class have the following order, where each subscript numeral represents the class number.

$$V_{OL1} > V_{OL2} > V_{OL3} > V_{OL4} \tag{7.43}$$

The design of complex logic gates is based on the same ideas as the design of NOR and NAND gates. We usually start by specifying a maximum V_{OL} value. The design objective is to determine the driver and load transistor sizes so that the complex logic gate achieves the specified V_{OL} value even in the worst case. The given V_{OL} value first allows us to find the $(W/L)_{load}$ and $(W/L)_{driver}$ ratios for an equivalent inverter. Next, we have to identify all worst-case (Class 1) paths in the circuit, and determine the transistor sizes in these worst-case paths such that each Class 1 path has the equivalent driver ratio of $(W/L)_{driver}$.

In this example, this design strategy yields the following ratios for the three worst-case paths.

$$\begin{aligned}
\left(\frac{W}{L}\right)_A &= \left(\frac{W}{L}\right)_D = 2\left(\frac{W}{L}\right)_{driver} \\
\left(\frac{W}{L}\right)_A &= \left(\frac{W}{L}\right)_E = 2\left(\frac{W}{L}\right)_{driver} \\
\left(\frac{W}{L}\right)_B &= \left(\frac{W}{L}\right)_C = 2\left(\frac{W}{L}\right)_{driver}
\end{aligned} \tag{7.44}$$

The transistor sizes found above guarantee that, for all other input combinations, the logic-low output voltage level will be less than the specified V_{OL}.

Complex CMOS Logic Gates

The realization of the n-net, or pull-down network, is based on the same basic design principles examined earlier. The pMOS pull-up network, on the other hand, must be the dual network of the n-net. This means that all parallel connections in the nMOS pull-down network will correspond to a series connection in the pMOS pull-up network, and all series connections in the pull-down network correspond to a parallel connection in the pull-up network.

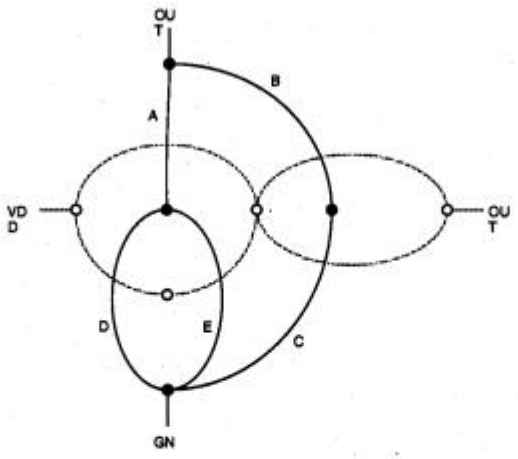


Figure 7.18. Construction of the dual pull-up graph from the pull-down graph, using the dual-graph concept.

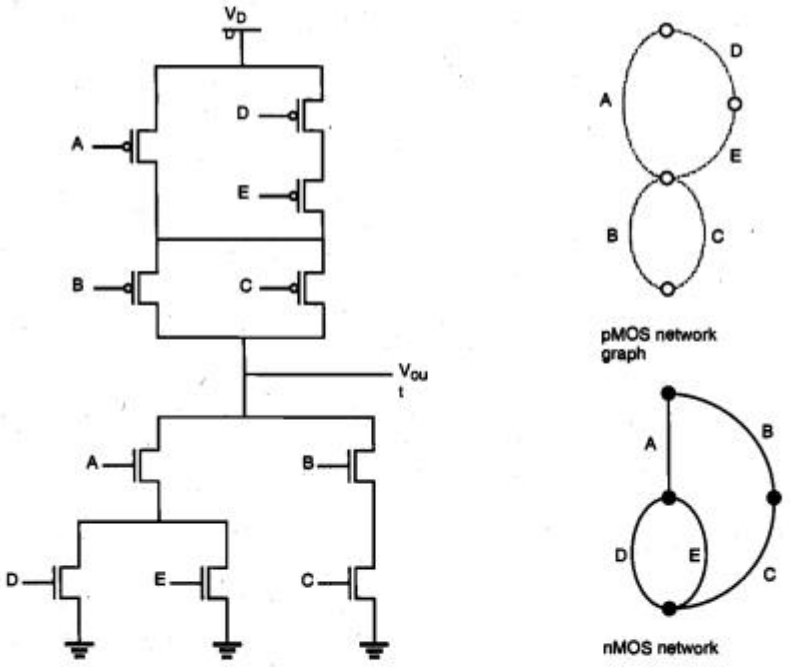


Figure 7.19. A complex CMOS logic gate realizing the Boolean function (7.41).

Figure 7.18 shows the simple construction of the dual p-net (pull-up) graph from the n-net (pull-down) graph. Each driver transistor in the pull-down network is represented by an edge, and each node is represented by a vertex in the pull-down graph. Next, a new vertex is created within each confined area in the pull-down graph, and neighboring vertices are connected by edges which cross each edge in the pull-down graph only once. This new graph represents the pull-up network. The resulting CMOS complex logic gate is shown in Fig. 7.19.

Layout of Complex CMOS Logic Gates

Now, we will investigate the problem of constructing a minimum-area layout for the complex CMOS logic gate. Figure 7.20 shows the stick-diagram layout of a "first attempt," using an arbitrary ordering of the polysilicon gate columns. Note that in this case, the separation between the polysilicon columns must allow for one diffusion-to-diffusion separation and two metal-to-diffusion contacts in between. This certainly consumes a considerable amount of extra silicon area.

If we can minimize the number of diffusion-area breaks both for nMOS and for pMOS transistors, the separation between the polysilicon gate columns can be made smaller, which will reduce the overall horizontal dimension and, hence, the circuit layout area. The number of diffusion breaks can be minimized by changing the *ordering* of the polysilicon columns.

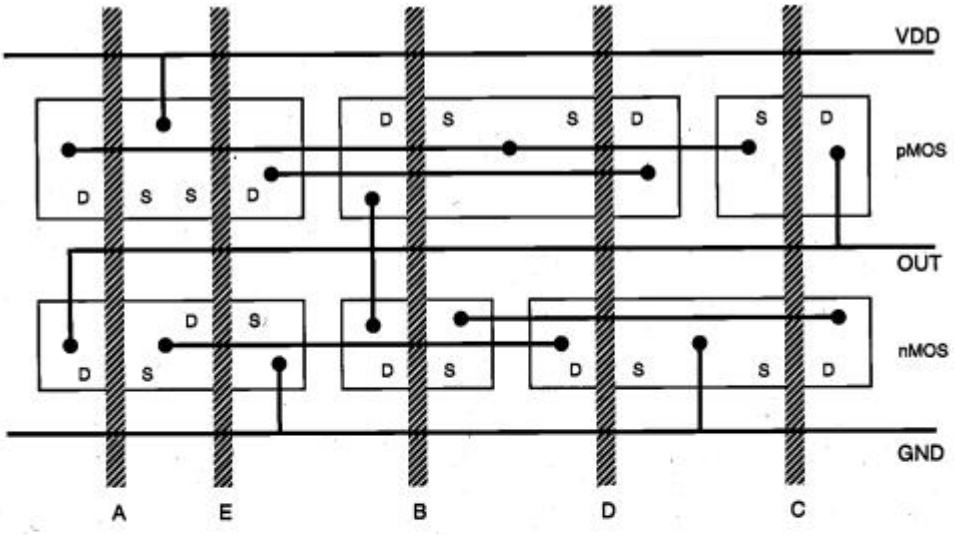


Figure 7.20. Stick-diagram layout of the complex CMOS logic gate, with an arbitrary ordering of the polysilicon gate columns.

A simple method for finding the optimum gate ordering is the Euler-path approach: find a Euler path in the pull-down graph and a Euler path in the pull-up graph with

identical ordering of input labels, i.e., find a common Euler path for both graphs. The Euler path is defined as an uninterrupted path that traverses each edge (branch) of the graph exactly once. Figure 7.21 shows the construction of a common Euler path for both graphs in our example.

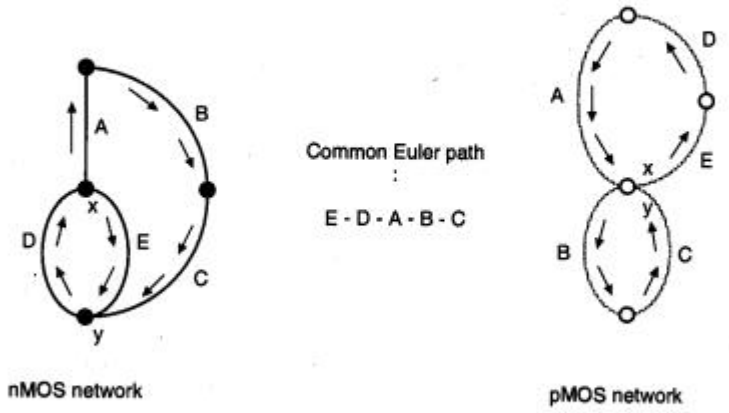


Figure 7.21. Finding a common Euler path in both graphs for n-net and p-net provides a gate ordering that minimizes the number of diffusion breaks and, thus, minimizes the logic-gate layout area. In both cases, the Euler path starts at (x) and ends at (y).

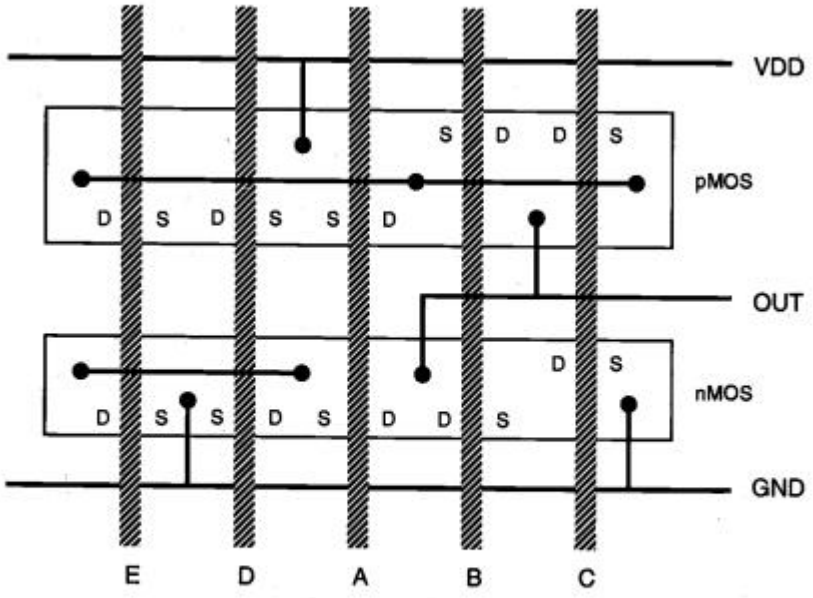


Figure 7.22. Optimized stick-diagram layout of the complex CMOS logic gate.

It is seen that there is a common sequence (E - D - A - B - C) in both graphs, i.e., a Euler path. The polysilicon gate columns can be arranged according to this sequence, which results in uninterrupted p-type and n-type diffusion areas. The stick diagram of the new layout is shown in Fig. 7.22. In this case, the polysilicon column separation Δd has to allow for only one metal-to-diffusion contact. The advantages of this new layout are more compact (smaller) layout area, simple routing of signals, and consequently, less parasitic capacitance.

As a further example of complex CMOS gates, the full-CMOS implementation of the exclusive-OR (XOR) function is shown in Fig. 7.23. Note that two additional inverters are also needed to obtain the inverse of both input variables (A and B). With these inverters, the CMOS XOR circuit in Fig. 7.23 requires a total of 12 transistors. Other CMOS realizations of the XOR gate that can be implemented with fewer transistors will be examined later.

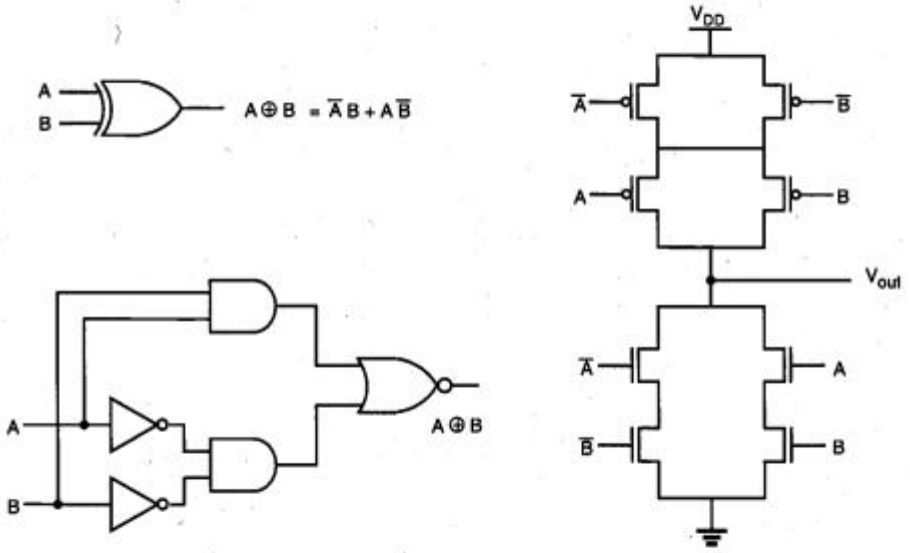


Figure 7.23. Full-CMOS implementation of the XOR function.

AOI and OAI Gates

While theoretically there are no strict limitations on the topology of the pull-down and the corresponding pull-up networks in a complex CMOS logic gate, we may recognize two important circuit categories as subsets of the general complex CMOS gate topology. These are the AND-OR-INVERT (AOI) gates and the OR-AND-INVERT (OAI) gates. The AOI gate, as its name suggests, enables the sum-of-products realization of a Boolean function in one logic stage (Fig. 7.24). The pull-down net of the AOI gate consists of parallel branches of series-connected nMOS driver transistors. The corresponding p-type pull-up network can simply be found using the dual-graph concept.

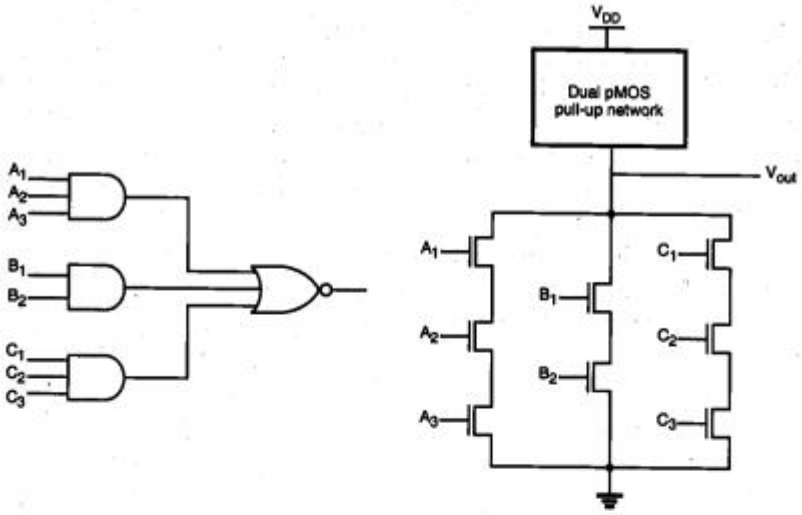


Figure 7.24. An AND-OR-INVERT (AOI) gate and the corresponding pull-down net.

The OAI gate, on the other hand, enables the product-of-sums realization of a Boolean function in one logic stage (Fig. 7.25). The pull-down net of the OAI gate consists of series branches of parallel-connected nMOS driver transistors, while the corresponding p-type pull-up network can be found using the dual-graph concept.

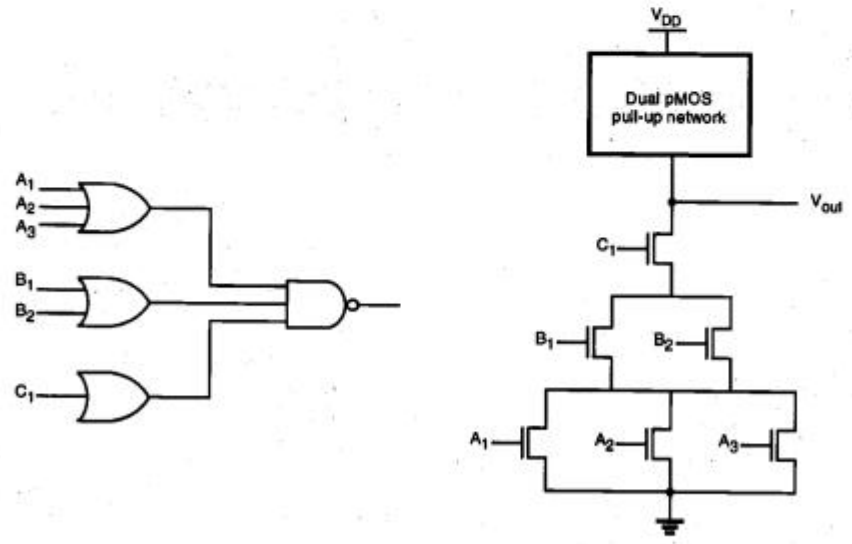


Figure 7.25. An OR-AND-INVERT (OAI) gate, and the corresponding pull-down net.

Pseudo-nMOS Gates

The large area requirements of complex CMOS gates present a problem in high-density designs, since two complementary transistors, one nMOS and one pMOS, are needed for every input. One possible approach to reduce the number of transistors is to use a single pMOS transistor, with its gate terminal connected to ground, as the *load* device (Fig. 7.26). With this simple pull-up arrangement, the complex gate can be implemented with much fewer transistors. The similarities of pseudo-nMOS gates to depletion-load nMOS logic gates are obvious.

The most significant disadvantage of using a pseudo-nMOS gate instead of a full-CMOS gate is the nonzero static power dissipation, since the always-on pMOS load device conducts a steady-state current when the output voltage is lower than V_{DD}. Also, the value of V_{OL} and the noise margins are now determined by the *ratio* of the pMOS load transconductance to the pull-down or driver transconductance.

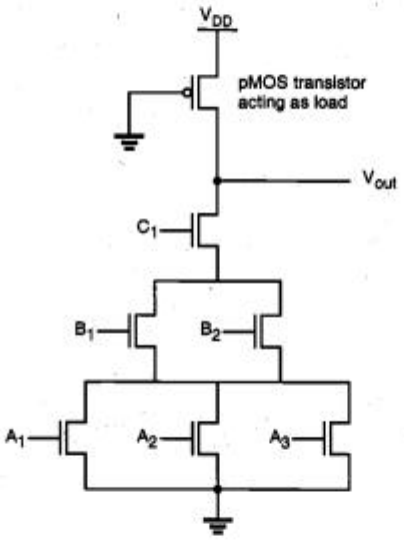
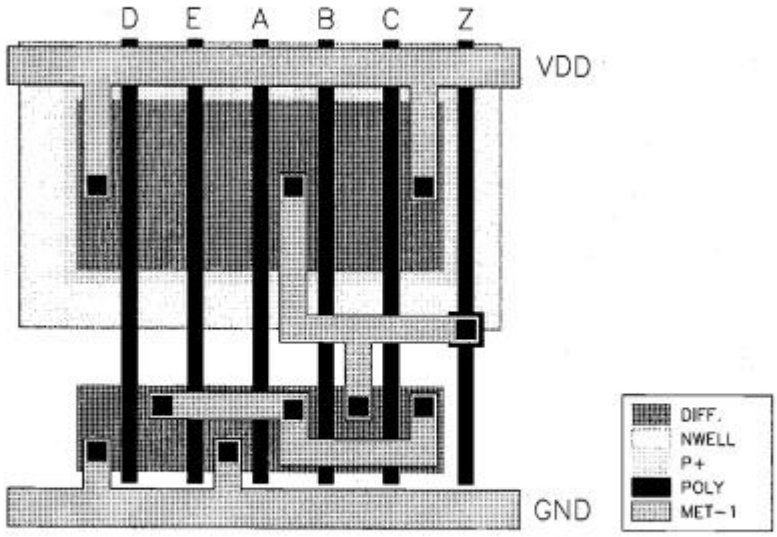


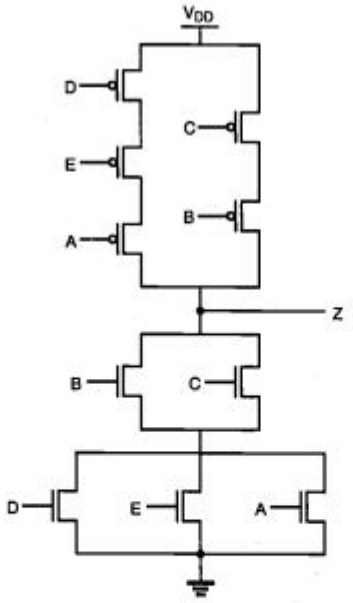
Figure 7.26. The pseudo-nMOS implementation of the OAI gate in Fig. 7.25.

Example 7.2.

The simplified layout of a CMOS complex logic circuit is given below. Draw the corresponding circuit diagram, and find an equivalent CMOS inverter circuit for simultaneous switching of all inputs, assuming that $(W/L)_p = 15$ for all pMOS transistors and $(W/L)_n = 10$ for all nMOS transistors.



The circuit diagram can be found from the layout by inspection:



The Boolean function realized by this circuit is

$$Z = \overline{(D+E+A)(B+C)}$$

The equivalent (W/L) ratios of the nMOS network and the pMOS network are determined by using the series-parallel equivalency rules discussed earlier in this chapter, as follows.

$$\left(\frac{W}{L}\right)_{n,eq} = \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_D + \left(\frac{W}{L}\right)_E + \left(\frac{W}{L}\right)_A} + \frac{1}{\left(\frac{W}{L}\right)_B + \left(\frac{W}{L}\right)_C}}$$

$$= \frac{1}{\frac{1}{30} + \frac{1}{20}} = 12$$

$$\left(\frac{W}{L}\right)_{p,eq} = \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_D} + \frac{1}{\left(\frac{W}{L}\right)_E} + \frac{1}{\left(\frac{W}{L}\right)_A}} + \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_B} + \frac{1}{\left(\frac{W}{L}\right)_C}}$$

$$= \frac{1}{\frac{1}{15} + \frac{1}{15} + \frac{1}{15}} + \frac{1}{\frac{1}{15} + \frac{1}{15}} = 12.5$$

CMOS Full-Adder Circuit

The one-bit full adder circuit is one of the most widely used building blocks in all data processing (arithmetic) and digital signal processing architectures. In the following, we will examine the circuit structure and the realization of the full adder using the conventional CMOS design style.

The sum_out and carry_out signals of the full adder are defined as the following two combinational Boolean functions of the three input variables, A, B, and C.

$$\begin{aligned} \text{sum\_out} &= A \oplus B \oplus C \\ &= ABC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}\,\overline{C}B \\ \text{carry\_out} &= AB + AC + BC \end{aligned}$$

A gate-level realization of these two functions is shown in Fig. 7.27. Note that instead of realizing the two functions independently, we use the carry_out signal to generate the sum output. This implementation will ultimately reduce the circuit complexity and, hence, save chip area. Also, we identify two separate sub-networks consisting of several gates (highlighted with dashed boxes) which will be utilized for the transistor-level realization of the full-adder circuit.

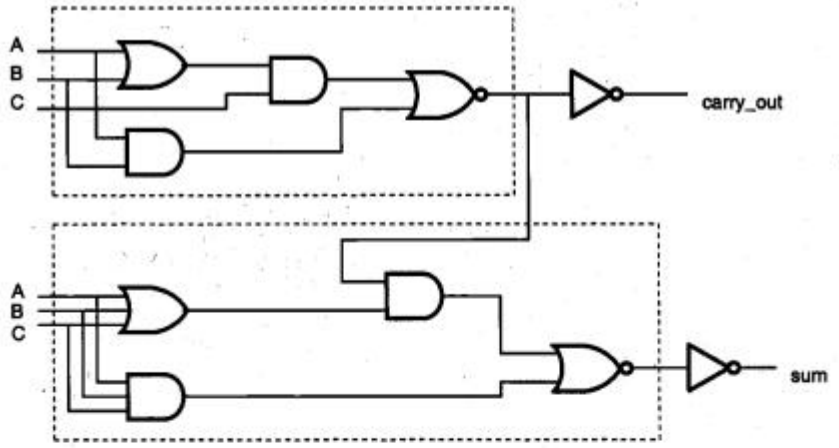


Figure 7.27. Gate-level schematic of the one-bit full-adder circuit.

The transistor-level design of the CMOS full-adder circuit is shown in Fig. 7.28. Note that the circuit contains a total of 14 nMOS and 14 pMOS transistors, together with the two CMOS inverters which are used to generate the outputs.

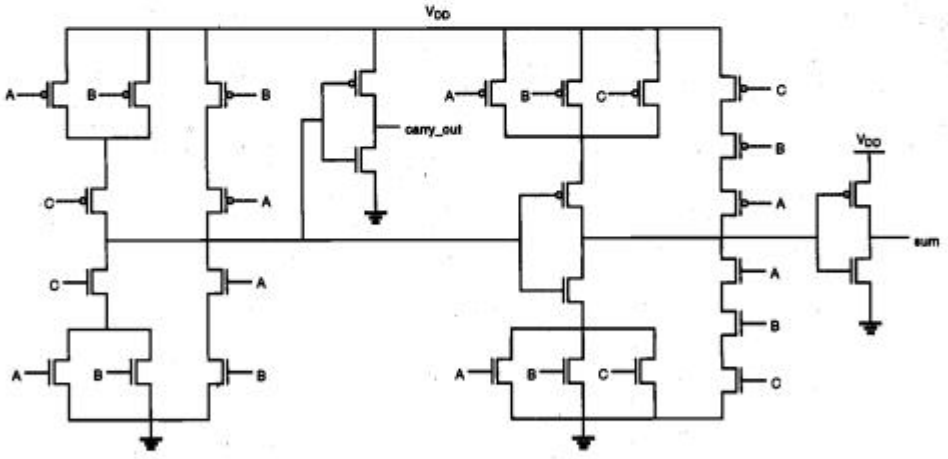


Figure 7.28. Transistor-level schematic of the one-bit full-adder circuit.

The mask layout of the full-adder circuit, which has been designed using the simple layout optimization strategy described earlier in this section, is shown in Fig. 7.29. Note, however, that all nMOS and pMOS transistors in this layout have the same (*W/L*) ratio. In order to optimize the transient (time-domain) performance of the circuit, it is usually necessary to adjust the transistor dimensions individually, as already shown in

Chapter 6. A performance-optimized and more compact mask layout of the same CMOS full-adder circuit is shown in Fig. 7.30.

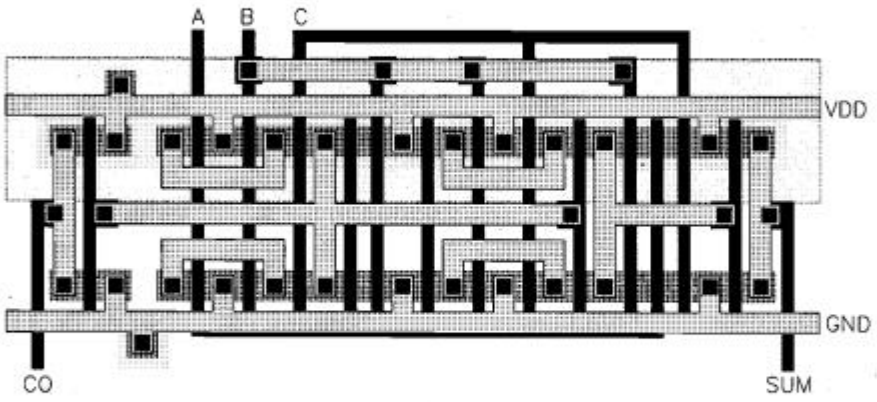


Figure 7.29. Mask layout of the CMOS full-adder circuit using minimum-size transistors.

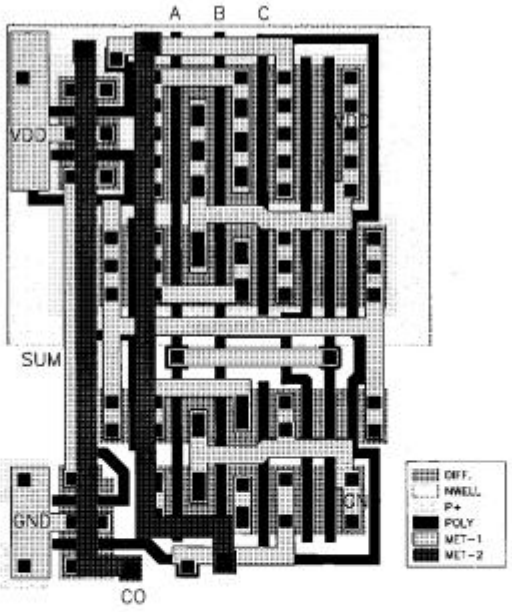


Figure 7.30. Mask layout of the optimized CMOS full adder circuit.

The simulated input and output voltage waveforms of the one-bit CMOS full adder are shown in Fig. 7.31. Please refer to the detailed design example that was presented in Chapter 1 for further design information.

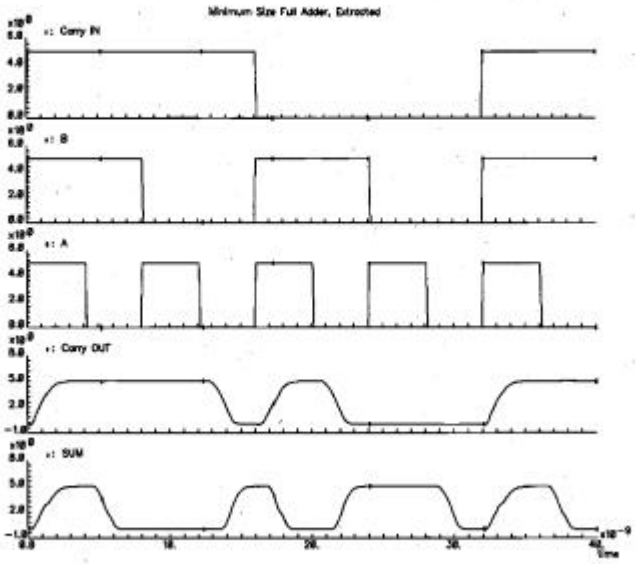


Figure 7.31. Simulated input and output waveforms of the CMOS full-adder circuit.

The full-adder circuit presented here can be used as the basic building block of a general n-bit binary adder, which accepts two n-bit binary numbers as input and produces the binary sum at the output. The simplest such adder can be constructed by a cascade-connection of full adders, where each adder stage performs a two-bit addition, produces the corresponding sum bit, and passes the carry output on to the next stage. Hence, this cascade-connected adder configuration is called the carry ripple adder (Fig. 7.32). The overall speed of the carry ripple adder is obviously limited by the delay of the carry bits rippling through the carry chain; therefore, a fast carry_out response becomes essential for the overall performance of the adder chain.

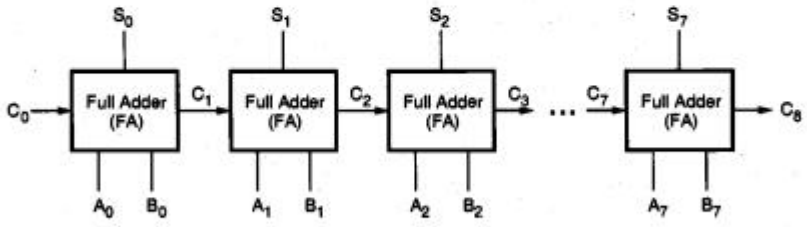


Figure 7.32. Block diagram of a carry ripple adder chain consisting of full adders.

7.5. CMOS Transmission Gates (Pass Gates)

In this section, we will examine a simple switch circuit called the CMOS transmission gate (TG) or pass gate, and present a new class of logic circuits which use the TGs as their basic building blocks. As shown in Fig. 7.33, the CMOS transmission gate consists of one nMOS and one pMOS transistor, connected in parallel. The gate voltages applied to these two transistors are also set to be complementary signals. As such, the CMOS TG operates as a bidirectional switch between the nodes A and B which is controlled by signal C.

If the control signal C is logic-high, i.e., equal to V_{DD}, then both transistors are turned on and provide a low-resistance current path between the nodes A and B. If, on the other hand, the control signal C is low, then both transistors will be off, and the path between the nodes A and B will be an open circuit. This condition is also called the high-impedance state.

Note that the substrate terminal of the nMOS transistor is connected to ground and the substrate terminal of the pMOS transistor is connected to V_{DD}. Thus, we must take into account the substrate-bias effect for both transistors, depending on the bias conditions. Figure 7.33 also shows three other commonly used symbolic representations of the CMOS transmission gate.

For a detailed DC analysis of the CMOS transmission gate, we will consider the following bias condition, shown in Fig. 7.34. The input node (A) is connected to a constant logic-high voltage, $V_{in} = V_{DD}$. The control signal is also logic-high, thus ensuring that both transistors are turned on. The output node (B) may be connected to a capacitor, which represents capacitive loading of the subsequent logic stages driven by the transmission gate. We will now investigate the input-output current-voltage relationship of the CMOS TG as a function of the output voltage V_{out}.

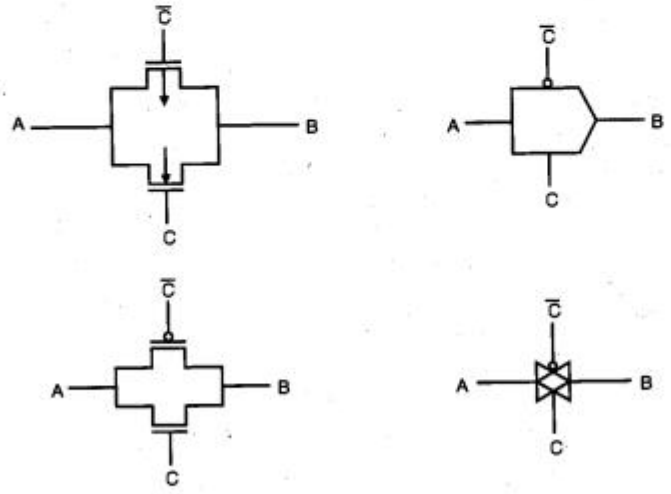


Figure 7.33. Four different representations of the CMOS transmission gate (TG).

It can be seen from Fig. 7.34 that the drain-to-source and the gate-to-source voltages of the nMOS transistor are

$$V_{DS,n} = V_{DD} - V_{out}$$
$$V_{GS,n} = V_{DD} - V_{out} \tag{7.45}$$

Thus, the nMOS transistor will be turned off for $V_{out} > V_{DD} - V_{T,n}$ and will operate in the saturation mode for $V_{out} < V_{DD} - V_{T,n}$. The V_{DS} and V_{GS} voltages of the pMOS transistor are

$$V_{DS,p} = V_{out} - V_{DD}$$
$$V_{GS,p} = -V_{DD} \tag{7.46}$$

Consequently, the pMOS transistor is in saturation for $V_{out} < |V_{T,p}|$, and it operates in the linear region for $V_{out} > |V_{T,p}|$. Note that, unlike the nMOS transistor, the pMOS transistor remains turned on, regardless of the output voltage level V_{out}.

This analysis has shown that we can identify three operating regions for the CMOS transmission gate, depending on the output voltage level. These operating regions are depicted in Fig. 7.34 as functions of V_{out}. The total current flowing through the transmission gate is the sum of the nMOS drain current and the pMOS drain current.

$$I_D = I_{DS,n} + I_{SD,p} \tag{7.47}$$

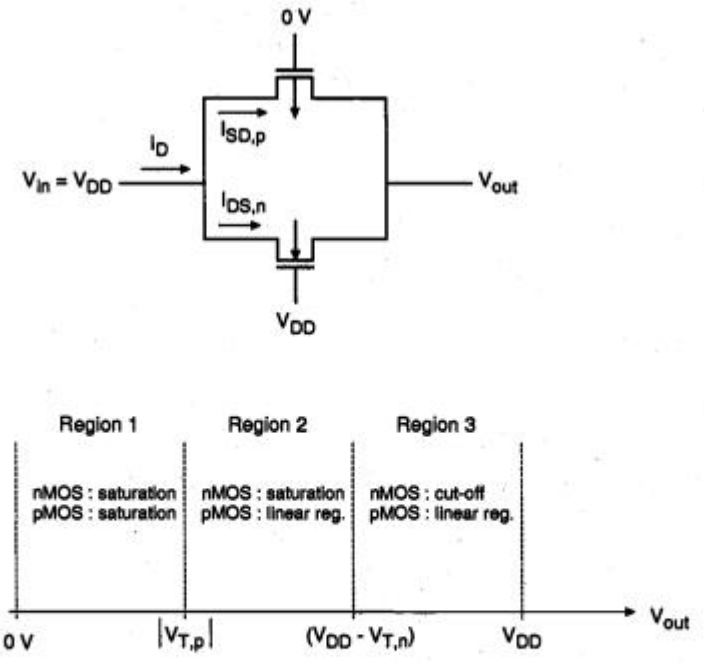


Figure 7.34. Bias conditions and operating regions of the CMOS transmission gate, shown as functions of the output voltage.

At this point, we may devise an *equivalent resistance* for each transistor in this structure, as follows.

$$R_{eq,n} = \frac{V_{DD} - V_{out}}{I_{DS,n}}$$
$$R_{eq,p} = \frac{V_{DD} - V_{out}}{I_{SD,p}} \tag{7.48}$$

The total equivalent resistance of the CMOS TG will then be the parallel equivalent of these two resistances, $R_{eq,n}$ and $R_{eq,p}$. Now, we will calculate the equivalent resistance values for the three operating regions of the transmission gate.

Region 1

Here, the output voltage is smaller than the absolute value of the pMOS transistor threshold voltage, i.e., $V_{out} < |V_{T,p}|$. According to Fig. 7.34, both transistors are in saturation. We obtain the equivalent resistance of both devices as

$$R_{eq,n} = \frac{2(V_{DD} - V_{out})}{k_n (V_{DD} - V_{out} - V_{T,n})^2} \tag{7.49}$$

$$R_{eq,p} = \frac{2(V_{DD} - V_{out})}{k_p (V_{DD} - |V_{T,p}|)^2} \tag{7.50}$$

Note that the source-to-substrate voltage of the nMOS transistor is equal to the output voltage V_{out}, while the source-to-substrate voltage of the pMOS transistor is equal to zero. Thus, we have to take into account the substrate-bias effect for the nMOS transistor in our calculations.

Region 2

In this region, $|V_{T,p}| < V_{out} < (V_{DD} - V_{T,n})$. Thus, the pMOS transistor now operates in the linear region, while the nMOS transistor continues to operate in saturation.

$$R_{eq,n} = \frac{2(V_{DD} - V_{out})}{k_n (V_{DD} - V_{out} - V_{T,n})^2} \tag{7.51}$$

$$R_{eq,p} = \frac{2(V_{DD} - V_{out})}{k_p\left[2\left(V_{DD} - |V_{T,p}|\right)(V_{DD} - V_{out}) - (V_{DD} - V_{out})^2\right]}$$
$$= \frac{2}{k_p\left[2\left(V_{DD} - |V_{T,p}|\right) - (V_{DD} - V_{out})\right]} \tag{7.52}$$

Region 3

Here, the output voltage is $V_{out} > (V_{DD} - V_{T,n})$. Consequently, the nMOS transistor will be turned off, which results in an open-circuit equivalent. The pMOS transistor will continue to operate in the linear region.

$$R_{eq,p} = \frac{2}{k_p\left[2\left(V_{DD} - |V_{T,p}|\right) - (V_{DD} - V_{out})\right]} \tag{7.53}$$

Combining the equivalent resistance values found for the three operating regions, we can now plot the total resistance of the CMOS transmission gate as a function of the output voltage V_{out}, as shown in Fig. 7.35.

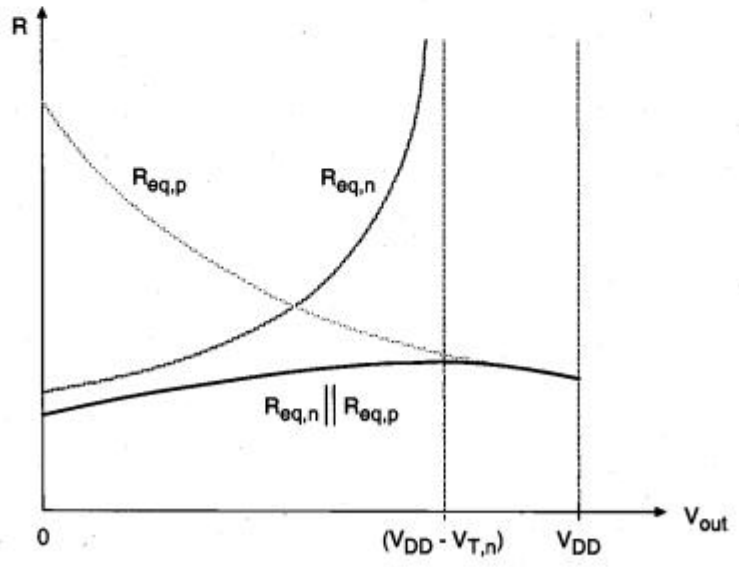


Figure 7.35. Equivalent resistance of the CMOS transmission gate, plotted as a function of the output voltage.

It can be seen that the total equivalent resistance of the TG remains relatively constant, i.e., its value is almost independent of the output voltage, whereas the individual equivalent resistances of both the nMOS and the pMOS transistors are strongly dependent on V_{out}. This property of the CMOS TG is naturally quite desirable. A CMOS pass gate which is turned on by a logic-high control signal can be replaced by its simple equivalent resistance for dynamic analysis, as shown in Fig. 7.36.

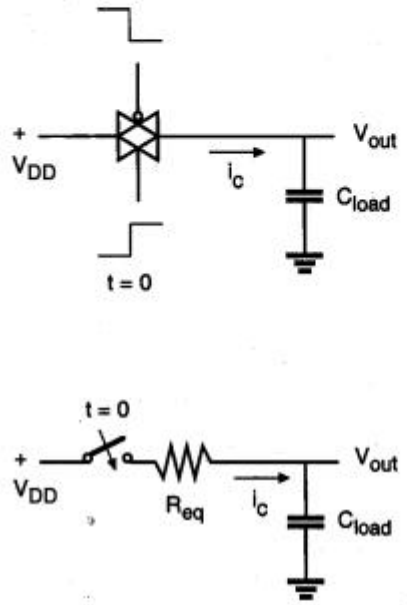


Figure 7.36. Replacing the CMOS TG with its resistor equivalent for transient analysis.

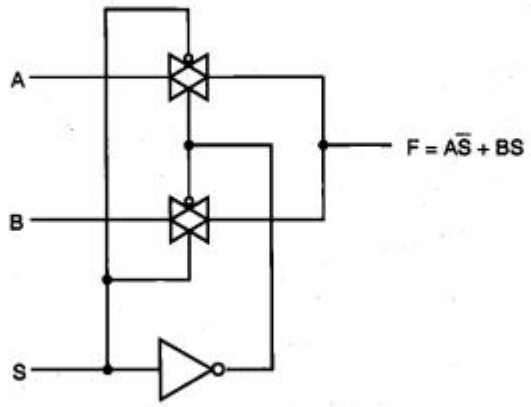


Figure 7.37. Two-input multiplexor circuit implemented using two CMOS TGs.

The implementation of CMOS transmission gates in logic circuit design usually results in compact circuit structures which may even require a smaller number of transistors than their standard CMOS counterparts. Note that the control signal *and* its complement must be available simultaneously for TG applications. Figure 7.37 shows a two-input multiplexor circuit consisting of two CMOS transmission gates. The operation of the multiplexor can be understood quite easily: If the control input S is logic-high, then the bottom TG will conduct, and the output will be equal to the input B. If the control signal is low, the bottom TG will turn off and the top TG will connect the input A to the output node.

Figure 7.38 shows an eight-transistor implementation of the logic XOR function, using two CMOS TGs and two CMOS inverters. The same function can also be implemented using only six transistors, as shown in Fig. 7.39.

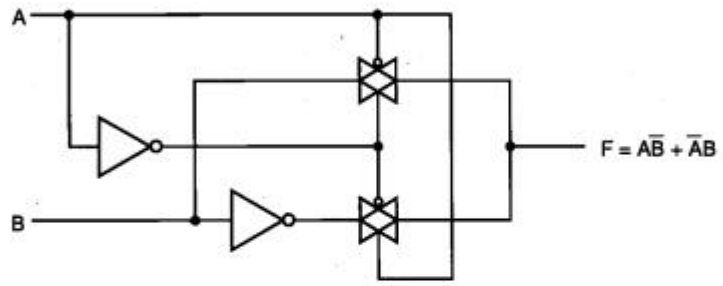


Figure 7.38. Eight-transistor CMOS TG implementation of the XOR function.

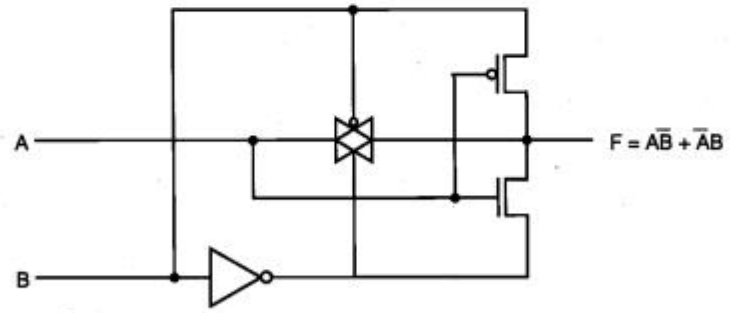


Figure 7.39. Six-transistor CMOS TG implementation of the XOR function.

Using the generalized multiplexor approach, each Boolean function can be realized with a TG logic circuit. As an example, Figure 7.40(a) shows the TG logic implementation of a three-variable Boolean function. Note that the three input variables *and* their inverses must be used to control the CMOS transmission gates. Including the three inverters not shown here, the TG implementation requires a total of 14 transistors. An

important point in TG logic design is that a conducting TG network (*low-impedance* path) should always be provided between the output node and one of the inputs, for all possible input combinations. This is to make sure that the output node with its capacitive load is never left in a *high-impedance* state.

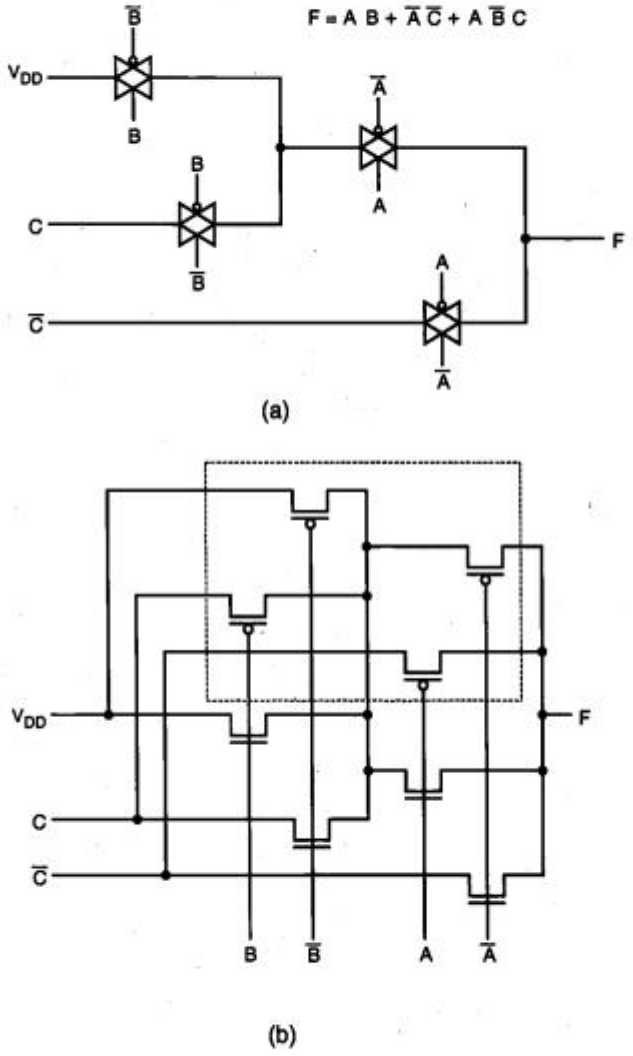


Figure 7.40. (a) CMOS TG realization of a three-variable Boolean function. (b) All pMOS transistors can be placed into one n-well to save area.

If each CMOS transmission gate in TG logic circuits is realized with a full nMOS-pMOS pair, the disjoint n-well structures of the pMOS transistors and the diffusion contacts may cause a significant overall area increase. In an attempt to reduce the silicon area occupied by TG circuits, the transmission gates can be laid out as separated nMOS-pMOS pairs with all pMOS transistors placed in one single n-well, as shown in Fig. 7.40(b). However, the *routing* area required for connecting the p-type diffusion regions to input signals must be carefully considered. The layout of the TG circuit is given in Fig. 7.41.

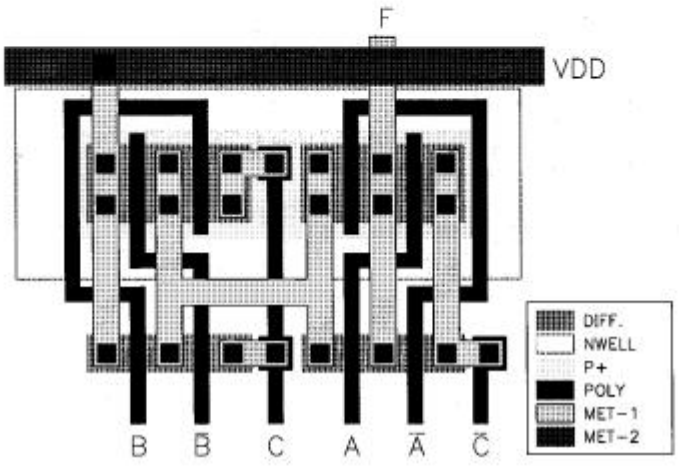


Figure 7.41. Mask layout of the CMOS TG circuit shown in Fig. 7.40.

Complementary Pass-Transistor Logic (CPL)

The complexity of full-CMOS pass-gate logic circuits can be reduced dramatically by adopting another circuit concept, called Complementary Pass-transistor Logic (CPL). The main idea behind CPL is to use a purely nMOS pass-transistor network for the logic operations, instead of a CMOS TG network. All inputs are applied in complementary form, i.e., every input signal and its inverse must be provided; the circuit also produces complementary outputs, to be used by subsequent CPL stages. Thus, the CPL circuit essentially consists of complementary inputs, an nMOS pass transistor logic network to generate complementary outputs, and CMOS output inverters to restore the output signals. The circuit diagrams of a CPL NOR2 and a CPL NAND2 are shown in Fig. 7.42.

The elimination of pMOS transistors from the pass-gate network significantly reduces the parasitic capacitances associated with each node in the circuit, thus, the operation speed is typically higher compared to a full-CMOS counterpart. But the improvement in transient characteristics comes at a price of increased process complexity. In CPL circuits, the threshold voltages of the nMOS transistors in the pass-gate network must be reduced to about 0 V through threshold-adjustment implants, in order

to eliminate the threshold-voltage drop. This, on the other hand, reduces the overall noise immunity and makes the transistors more susceptible to subthreshold conduction in the off-mode. Also note that the CPL design style is highly modular, a wide range of functions can be realized by using the same basic pass-transistor structures.

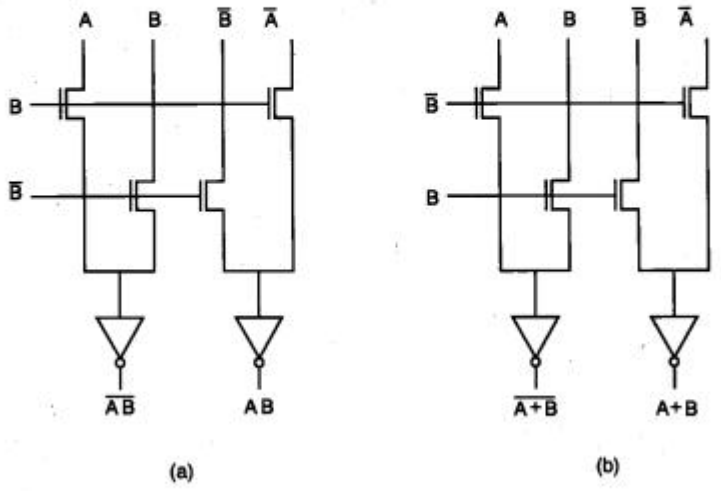


Figure 7.42. Circuit diagram of (a) CPL NAND2 gate and (b) CPL NOR2 gate.

Regarding the transistor count, CPL circuits do not always offer a marked advantage over conventional CMOS. The NAND2 and NOR2 circuits shown in Fig. 7.42 each consist of 8 transistors. XOR and XNOR functions realized with CPL have a similar complexity (i.e., transistor count) as conventional CMOS realizations. The same observation is true for the realization of full adders with CPL. The circuit diagram of a CPL-based XOR gate is shown in Fig. 7.43. Here, the cross-coupled pMOS pull-up transistors are used to speed up the output response. Transistor widths are given in λ-units. Figure 7.44 shows the circuit diagram of a CPL full-adder circuit consisting of 32 transistors. The mask layout of this CPL circuit is given in Fig. 7.45.

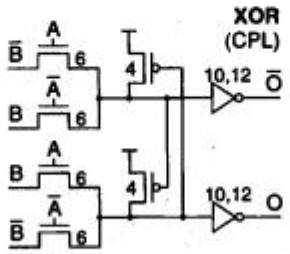


Figure 7.43. Circuit diagram of a CPL-based XOR gate.

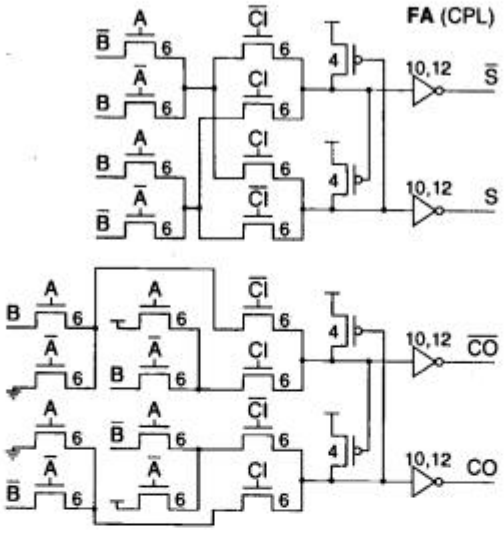


Figure 7.44. Circuit diagram of a CPL full adder.

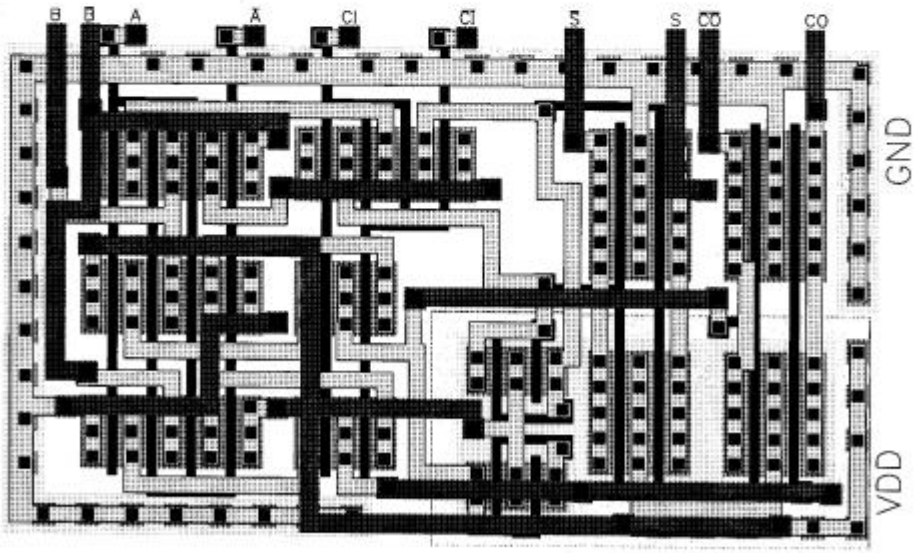


Figure 7.45. Mask layout of the CPL full adder shown in Fig. 7.44.

Behavior of Bistable Elements

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

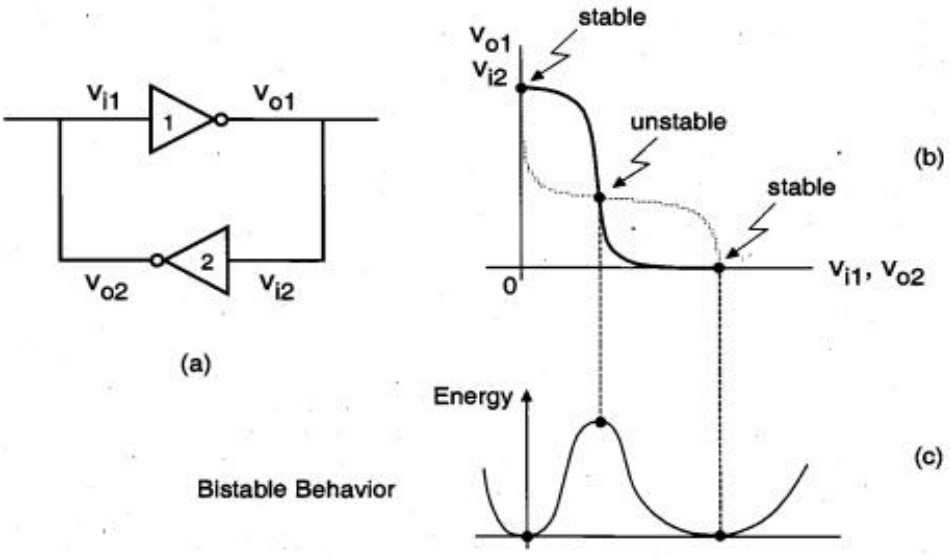


Figure 8.2. Static behavior of the two-inverter basic bistable element: (a) Circuit schematic. (b) Intersecting voltage transfer curves of the two inverters, showing the three possible operating points. (c) Qualitative view of the potential energy levels corresponding to the three operating points.

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

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

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

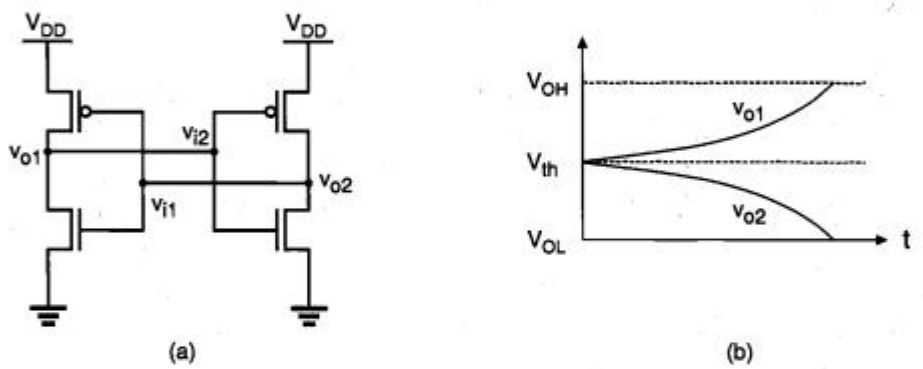


Figure 8.3. (a) Circuit diagram of a CMOS bistable element. (b) One possibility for the expected time-domain behavior of the output voltages, if the circuit is initially set at its unstable operating point.

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

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

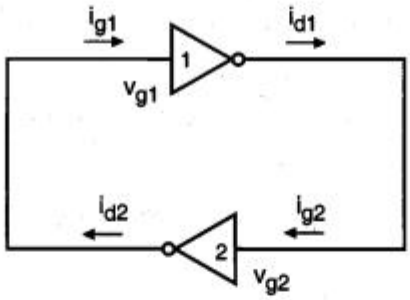


Figure 8.4. Small-signal input and output currents of the inverters.

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

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

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

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

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

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

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

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

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

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

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

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

Both differential equations given in (8.6) and (8.7) can now be combined to yield a second-order differential equation describing the time behavior of gate charge q_1.

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

This equation can also be expressed in a more simplified form by using τ_0, the transit time constant.

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

The time-domain solution of (8.9) for q_1 yields

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

where the initial condition is given as:

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

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

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

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

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

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

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

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

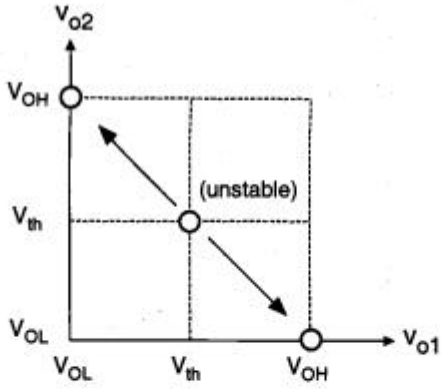


Figure 8.5. Phase-plane representation of the bistable circuit behavior.

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

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

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

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

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

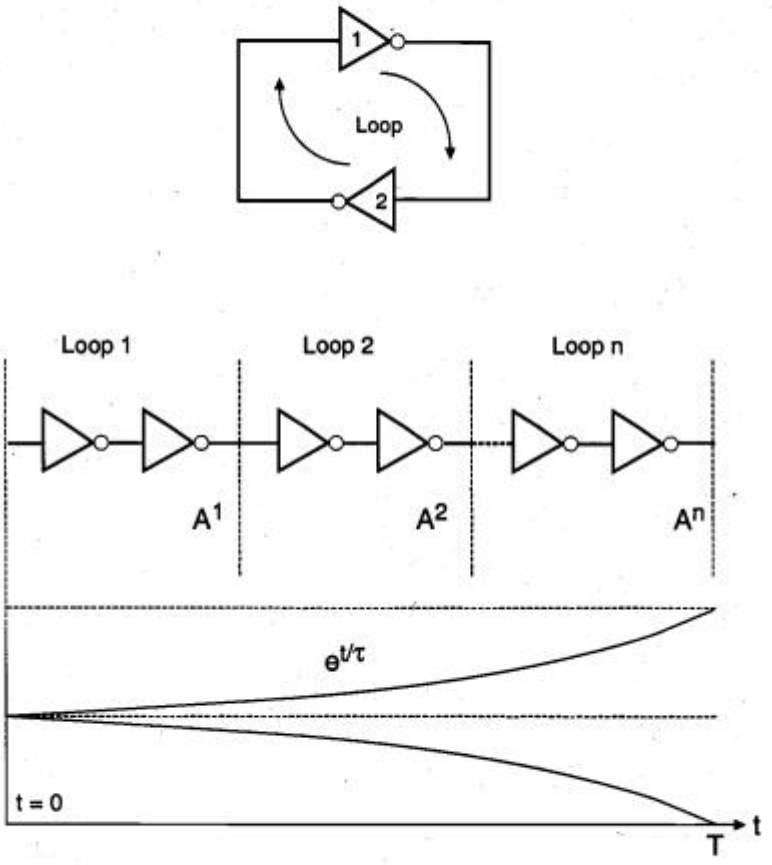


Figure 8.6. Propagation of a transient signal in the two-inverter loop during settling.

$$A^n = e^{+\frac{T}{\tau_0}} \tag{8.17}$$

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

The SR Latch Circuit

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

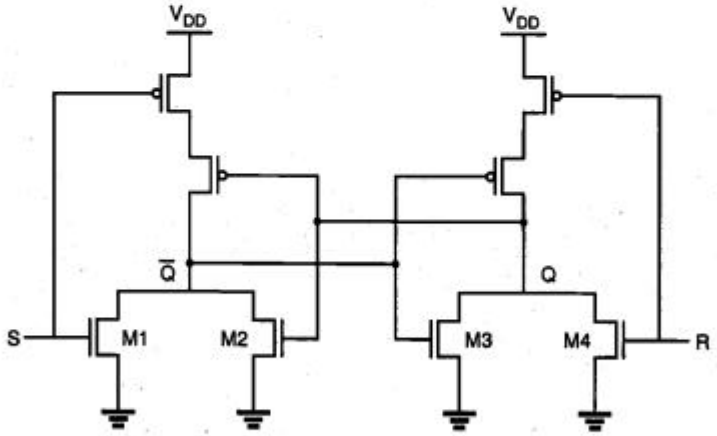


Figure 8.7. CMOS SR latch circuit based on NOR2 gates.

The SR latch circuit has two complementary outputs, Q and $\overline{Q}$. By definition, the latch is said to be in its *set* state when Q is equal to logic "1" and $\overline{Q}$ is equal to logic "0." Conversely, the latch is in its *reset* state when the output Q is equal to logic "0" and $\overline{Q}$ is equal to "1." The gate-level schematic of the SR latch consisting of two NOR2 gates, and the corresponding block diagram representation are shown in Fig. 8.8. It can easily be seen that when both input signals are equal to logic "0," the SR latch will operate exactly

like the simple cross-coupled bistable element examined earlier, i.e., it will preserve (hold) either one of its two stable operating points (states) as determined by the previous inputs.

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

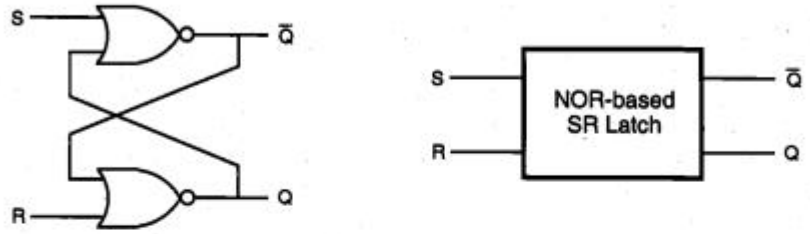


Figure 8.8. Gate-level schematic and block diagram of the NOR-based SR latch.

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

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

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

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

and M4 are turned off, which results in a logic-high voltage V_{OH} at node Q. If the reset input (R) is equal to V_{OH} and the set input (S) is equal to V_{OL}, the situation will be reversed (M1 and M2 turned off and M3 and M4 turned on).

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

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

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

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

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

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

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

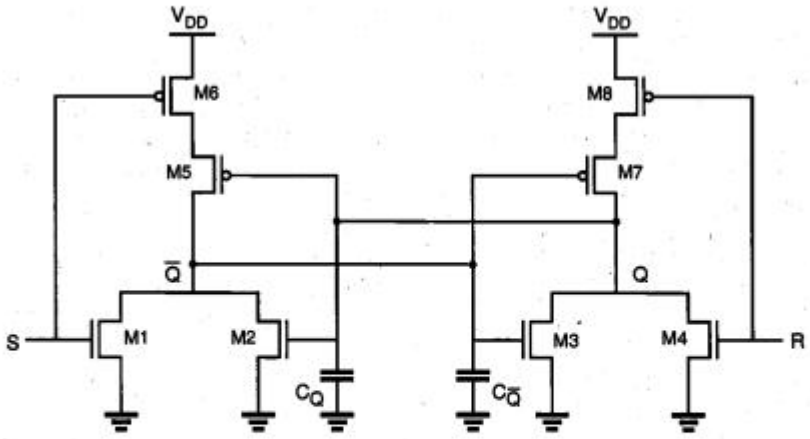


Figure 8.9. Circuit diagram of the CMOS SR latch showing the lumped load capacitances at both output nodes.

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

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

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

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

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

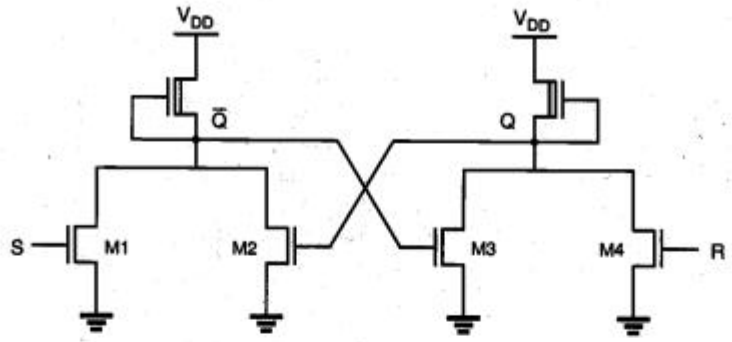


Figure 8.10. Depletion-load nMOS SR latch circuit based on NOR2 gates.

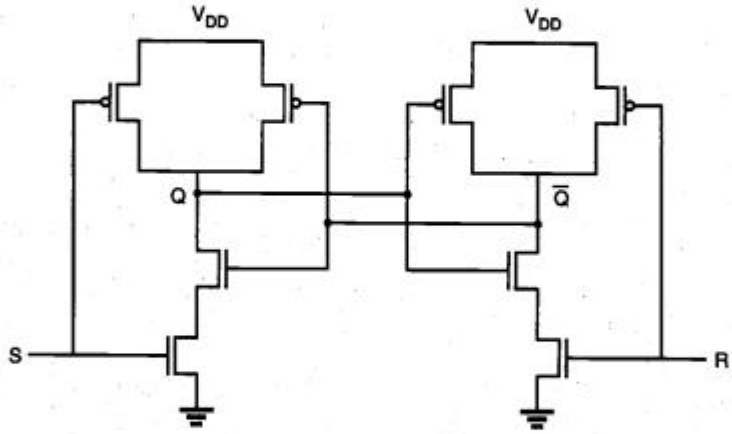
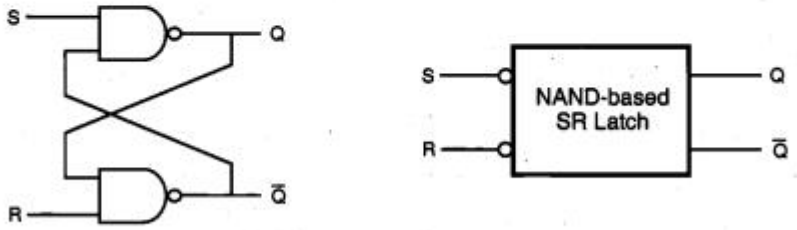


Figure 8.11. CMOS SR latch circuit based on NAND2 gates.

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

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

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



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

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

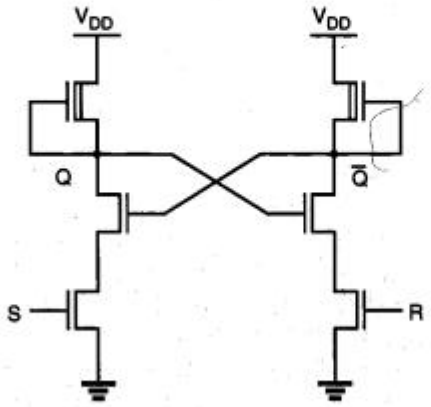


Figure 8.13. Depletion-load nMOS NAND-based SR latch circuit.

Clocked Latch and Flip-Flop Circuits

Clocked SR Latch

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

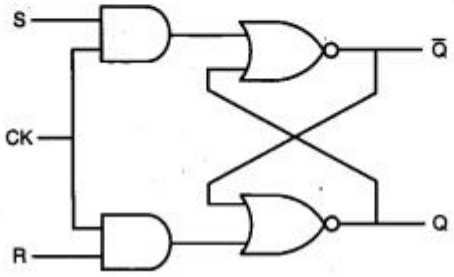


Figure 8.14. Gate-level schematic of the clocked NOR-based SR latch.

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

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

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

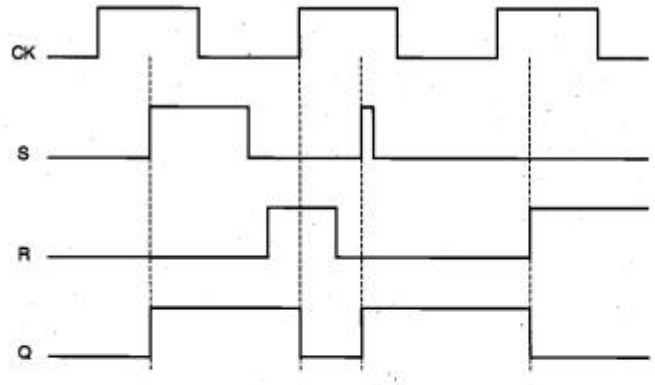


Figure 8.15. Sample input and output waveforms illustrating the operation of the clocked NOR-based SR latch circuit.

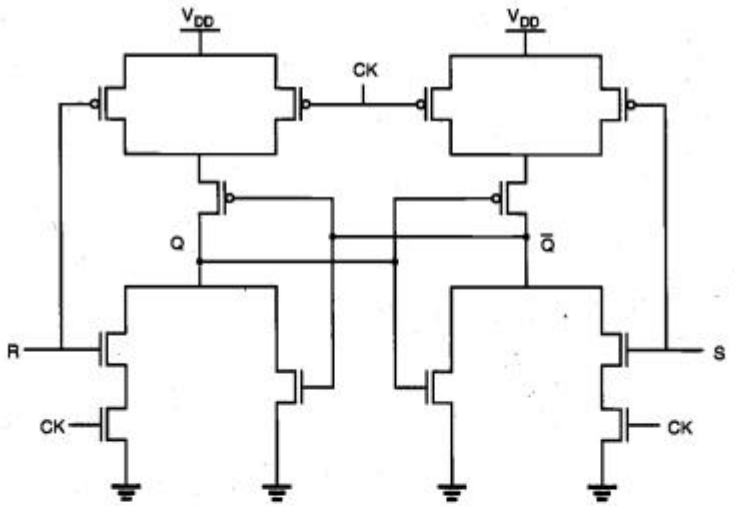


Figure 8.16. AOI-based implementation of the clocked NOR-based SR latch circuit.

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

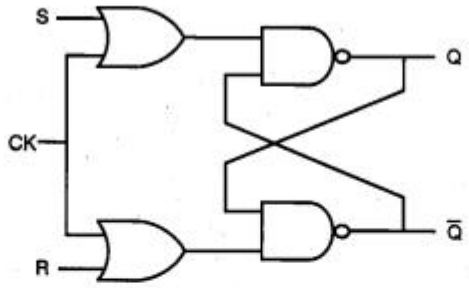


Figure 8.17. Gate-level schematic of the clocked NAND-based SR latch circuit, with active low inputs.

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

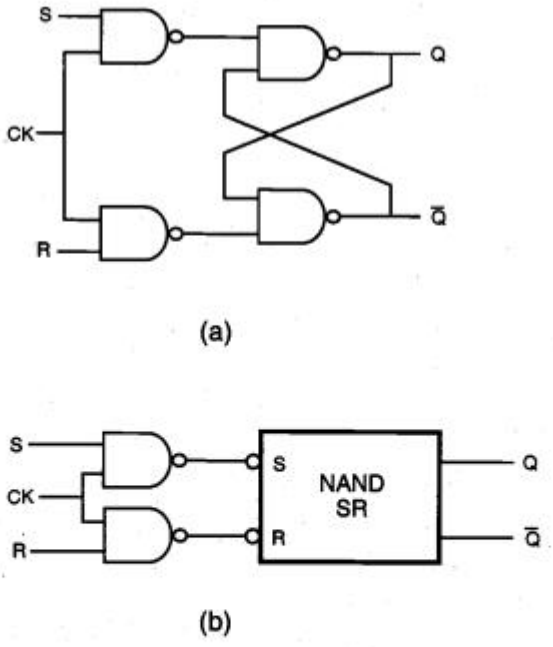


Figure 8.18. (a) Gate-level schematic of the clocked NAND-based SR latch circuit, with active high inputs. (b) Partial block diagram representation of the same circuit.

Clocked JK Latch

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

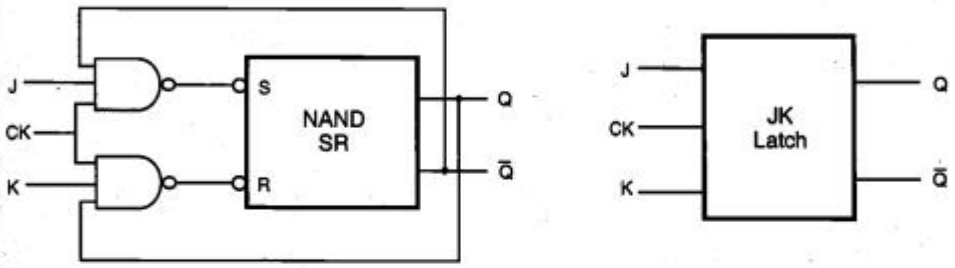


Figure 8.19. Gate-level schematic of the clocked NAND-based JK latch circuit.

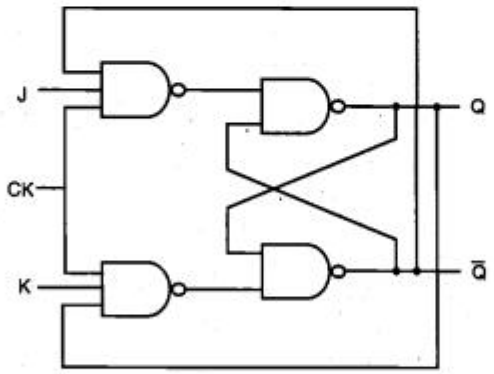


Figure 8.20. All-NAND implementation of the clocked JK latch circuit.

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

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

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

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

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

(a)

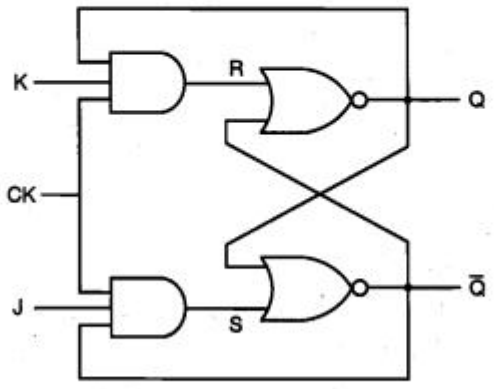


Figure 8.21. (a) Gate-level schematic of the clocked NOR-based JK latch circuit.

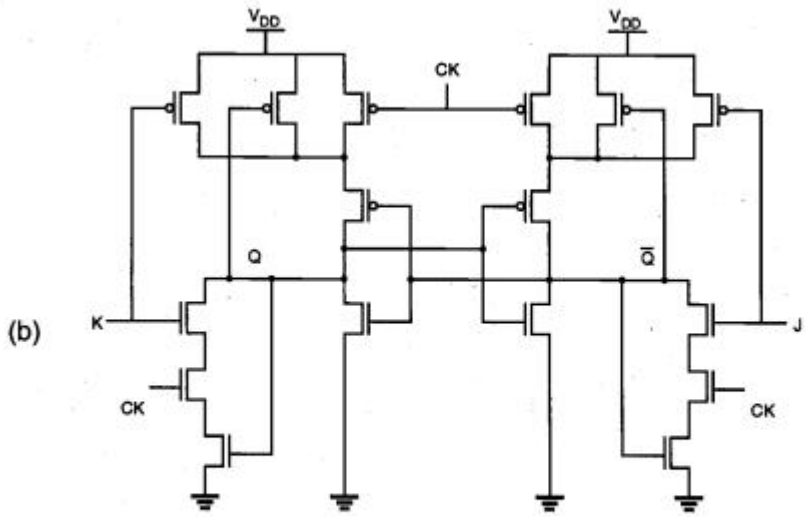


Figure 8.21. (continued) (b) CMOS AOI realization of the JK latch.

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

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

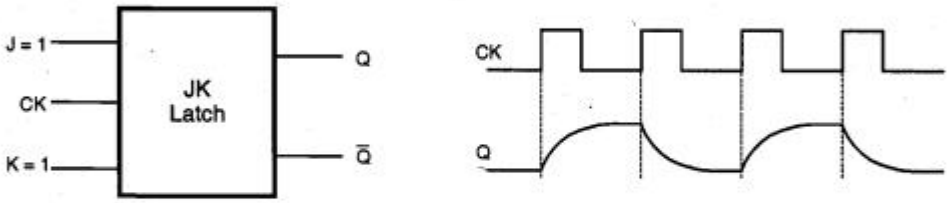


Figure 8.22. Operation of the JK latch as a toggle switch.

Master-Slave Flip-Flop

Most of the timing limitations encountered in the previously examined clocked latch circuits can be prevented by using two latch stages in a cascaded configuration. The key

operation principle is that the two cascaded stages are activated with opposite clock phases. This configuration is called the *master-slave flip-flop*. Our definition of flip-flop is designed to distinguish it from latches discussed previously, although they are mostly used interchangeably in the literature.

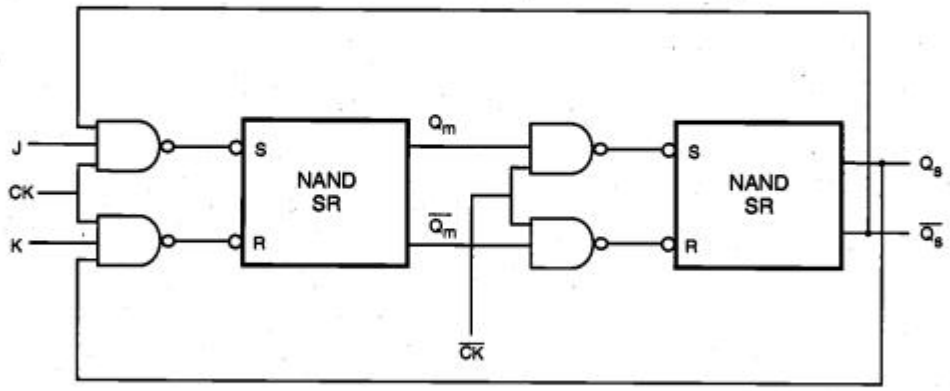


Figure 8.23. Master-slave flip-flop consisting of NAND-based JK latches.

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

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

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

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

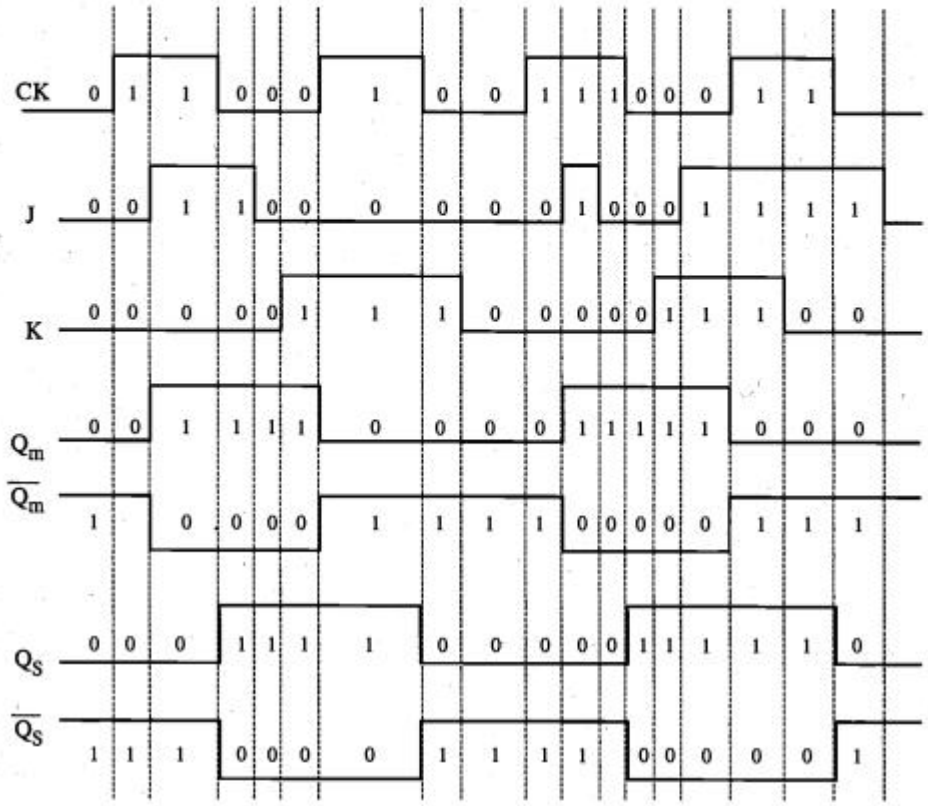


Figure 8.24. Sample input and output waveforms of the master-slave flip-flop circuit.

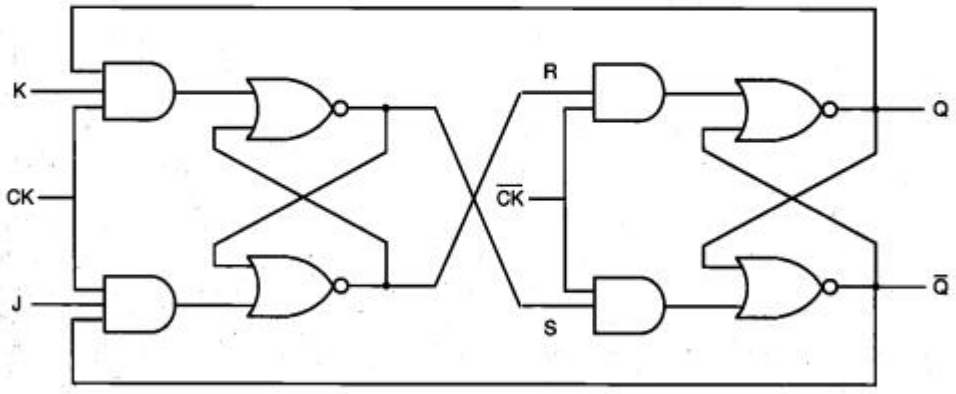


Figure 8.25. NOR-based realization of the JK master-slave flip-flop.

CMOS D-Latch and Edge-Triggered Flip-Flop

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

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

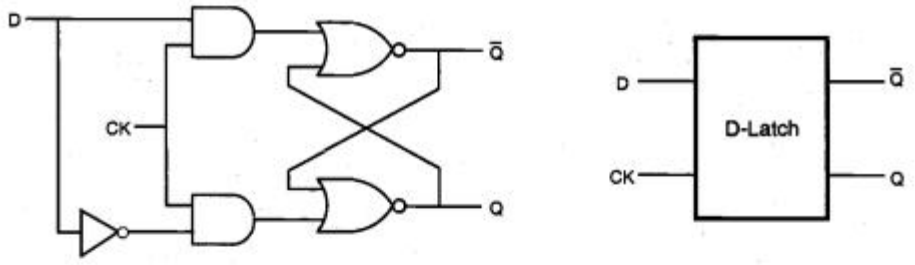


Figure 8.26. Gate-level schematic and the block diagram view of the D-latch.

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

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

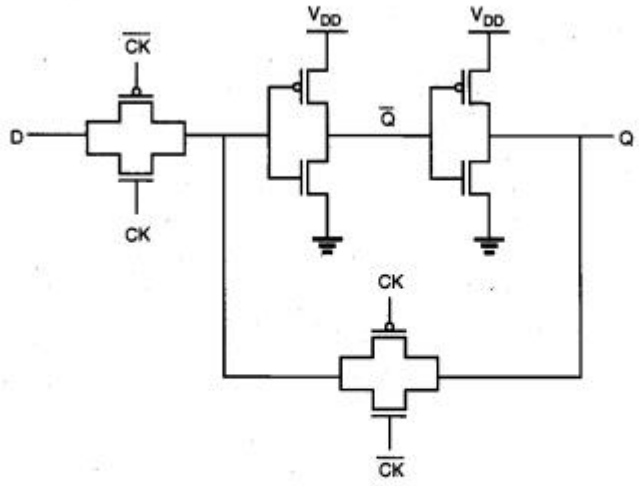


Figure 8.27. CMOS implementation of the D-latch (version 1).

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

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

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

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

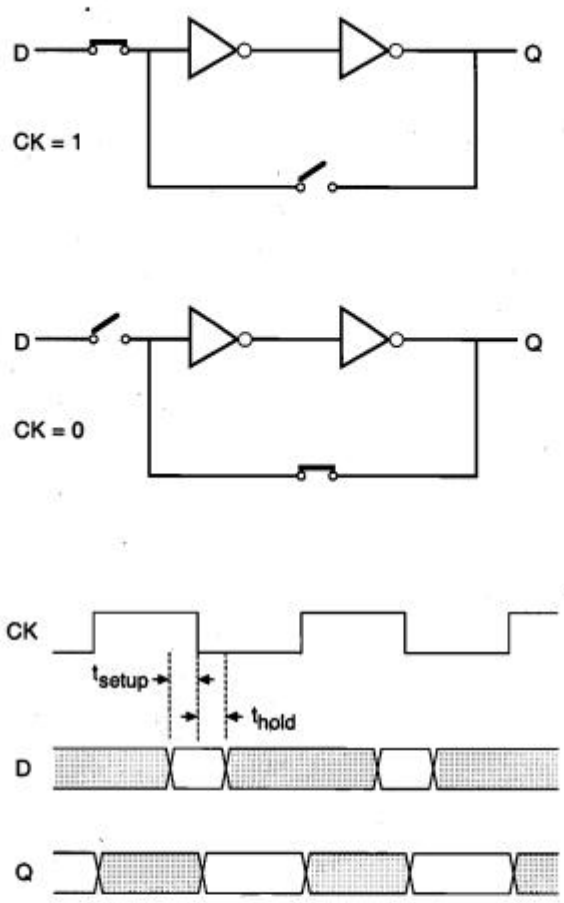


Figure 8.28. Simplified schematic view and the corresponding timing diagram of the CMOS D-latch circuit, showing the setup time and the hold time.

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

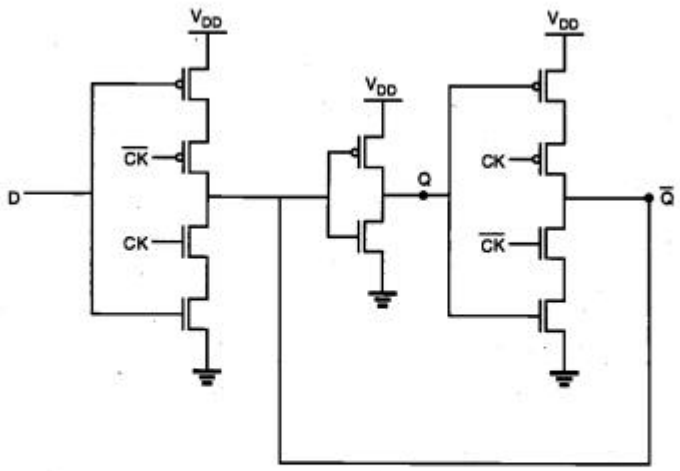


Figure 8.29. CMOS implementation of the D-latch (version 2).

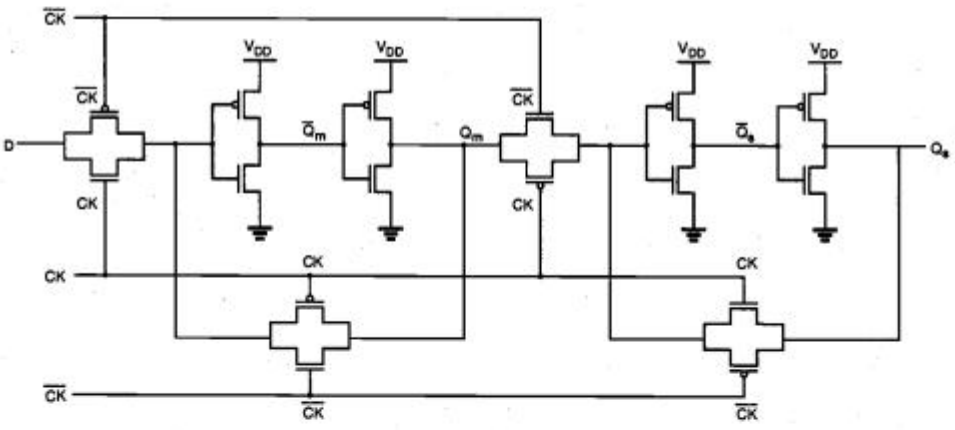


Figure 8.30. CMOS negative (falling) edge-triggered master-slave D flip-flop (DFF).

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

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

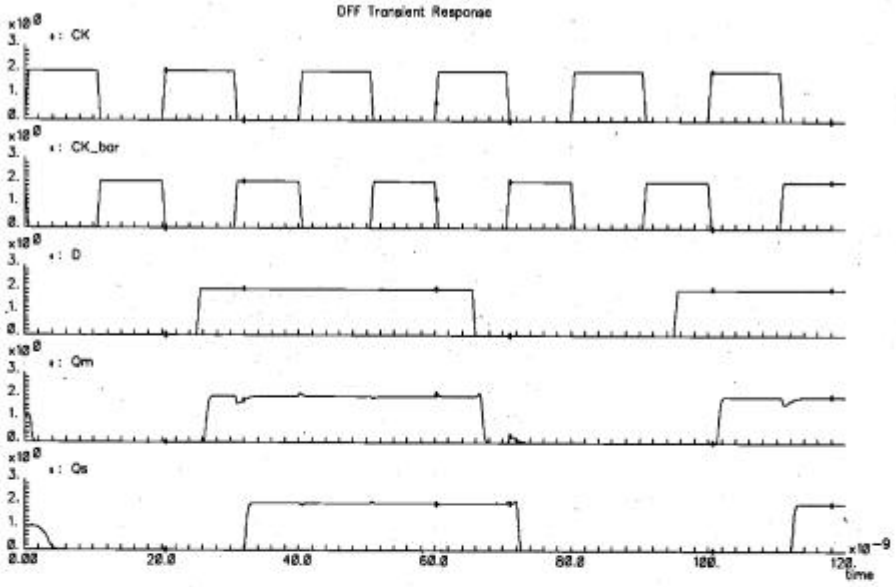


Figure 8.31. Simulated input and output waveforms of the CMOS DFF circuit in Fig. 8.30.

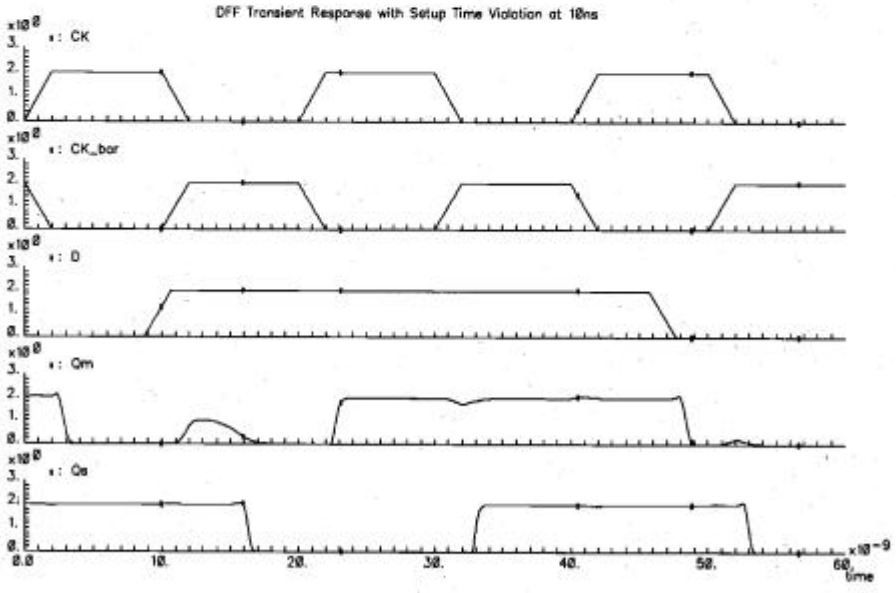


Figure 8.32. Simulated waveforms of the CMOS DFF circuit, showing a set-up time violation for the master stage input at 10 ns. The output of the master stage fails to settle at the correct level.

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

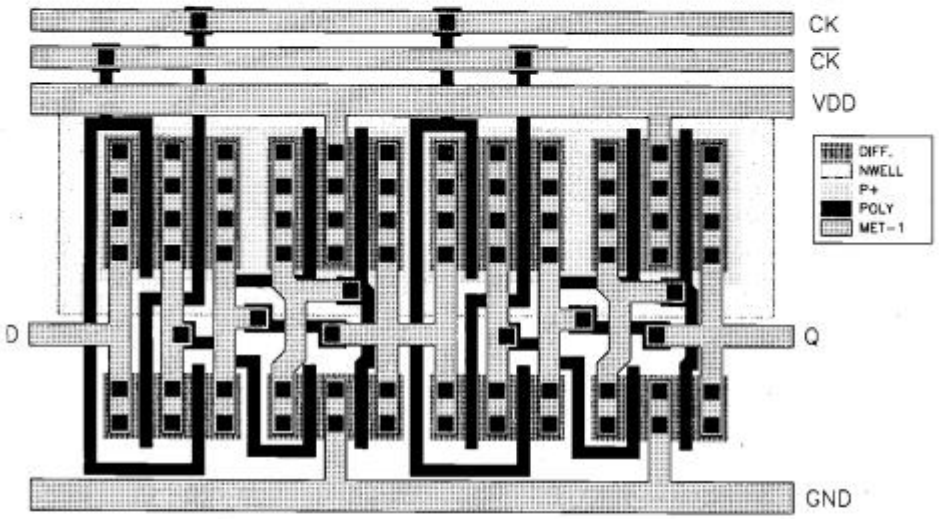


Figure 8.33. Layout of the CMOS DFF shown in Fig. 8.30.

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

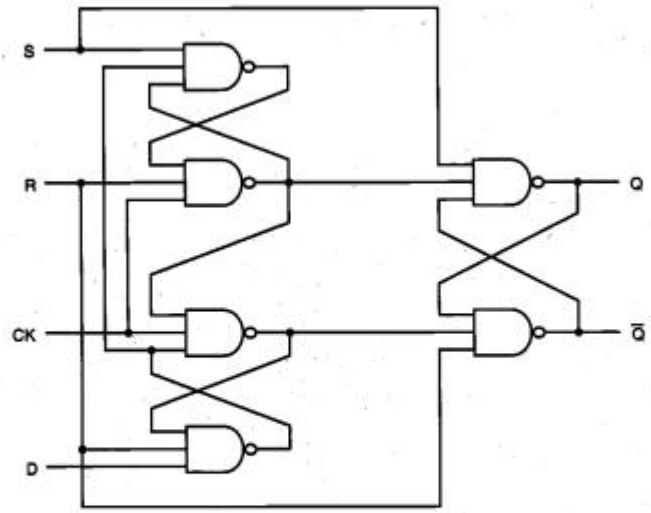


Figure 8.34. NAND3-based positive edge-triggered D flip-flop circuit.

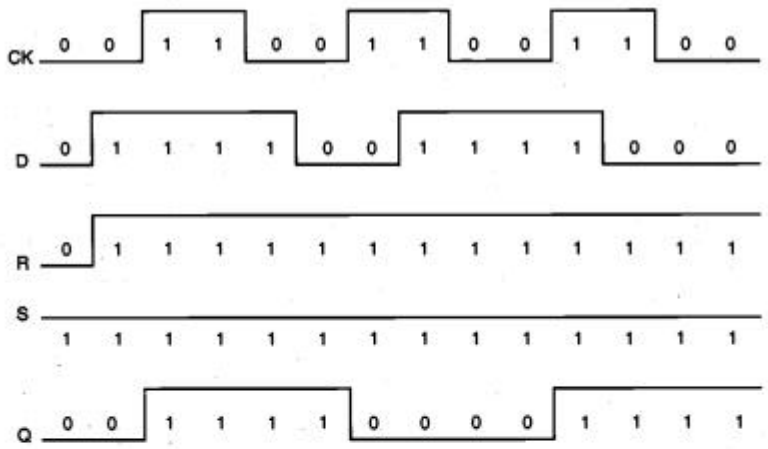


Figure 8.35. Timing diagram of the positive edge-triggered D flip-flop.

Voltage Bootstrapping

In this section, we will briefly examine a very useful dynamic circuit technique for overcoming threshold voltage drops in digital circuits, which is called *voltage bootstrapping*. We have already seen that output voltage levels may suffer from threshold voltage drops in several circuit structures, such as pass transistor gates or enhancement-load inverters and logic gates.

Dynamic voltage bootstrapping techniques offer a simple yet effective way to overcome threshold voltage drops which occur in most situations. Consider the circuit shown in Fig. 9.11, where the voltage V_x is equal to or smaller than the power supply voltage, $V_x \leq V_{DD}$. Consequently, the enhancement-type nMOS transistor M2 will operate in saturation.

When the input voltage V_{in} is low, the maximum value that the output voltage can attain is limited by

$$V_{out}(max) = V_x - V_{T2}(V_{out}) \tag{9.24}$$

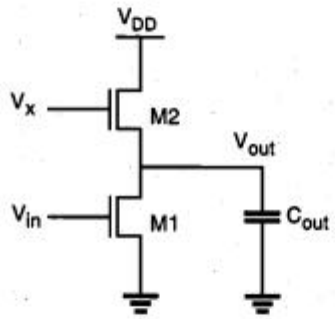


Figure 9.11. Enhancement-type circuit in which the output node is weakly driven.

To overcome the threshold voltage drop and to obtain a full logic-high level (V_{DD}) at the output node, the voltage V_x must be increased. Now consider the circuit shown in Fig. 9.12, where a third transistor M3 has been added to the circuit. The two capacitors C_S and C_{boot} seen in the circuit diagram represent the capacitances which dynamically couple the voltage V_x to the ground and to the output, respectively. We will see that this circuit can produce a high V_x during switching, so that the threshold voltage drop can be overcome at the output node.

$$V_x \geq V_{DD} + V_{T2}(V_{out}) \tag{9.25}$$

Initially assume that the input voltage V_{in} is logic-high, so that M1 and M2 have a nonzero drain current and that the output voltage is low. At this point, M1 is in the linear region and M2 is in saturation. Since $I_{D3} = 0$, the initial condition for the voltage V_x can be found as

$$V_x = V_{DD} - V_{T3}(V_x) \tag{9.26}$$

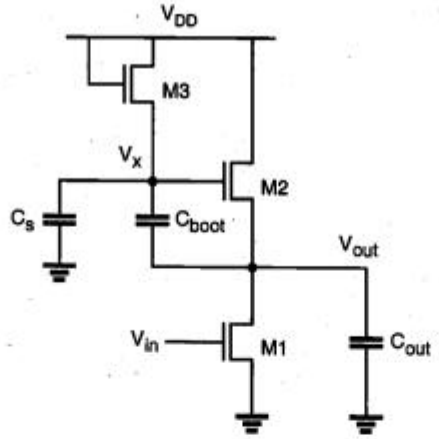


Figure 9.12. Dynamic bootstrapping arrangement to boost V_x during switching.

Now, assume that the input switches from its logic-high level to 0 V at $t = 0$. As a result, the driver transistor M1 will turn off and the output voltage V_{out} will start to rise. This change in the output voltage level will now be coupled to V_x through the bootstrap capacitor, C_{boot}. Let i_{Cboot} represent the transient current flowing through the capacitor C_{boot} during this charge-up event, and let i_{Cs} be the current through C_S. Assuming that the two current components are approximately equal, we obtain

$$i_{Cs} = i_{Cboot} \Leftrightarrow C_S \frac{dV_x}{dt} \approx C_{boot} \frac{d(V_{out} - V_x)}{dt} \tag{9.27}$$

Reorganizing (9.27) yields the following equation.

$$(C_S + C_{boot}) \frac{dV_x}{dt} \approx C_{boot} \frac{dV_{out}}{dt} \tag{9.28}$$

$$\frac{dV_x}{dt} \approx \frac{C_{boot}}{(C_S + C_{boot})} \cdot \frac{dV_{out}}{dt} \tag{9.29}$$

It can be seen from (9.29) that the increase in the output voltage V_{out} during this switching event will generate a proportional increase in the voltage level V_x. Integrating both sides of (9.29), we obtain

$$\int_{V_{DD} - V_{T3}}^{V_x} dV_x = \frac{C_{boot}}{(C_S + C_{boot})} \cdot \int_{V_{OL}}^{V_{DD}} dV_{out} \tag{9.30}$$

$$V_x = (V_{DD} - V_{T3}) + \frac{C_{boot}}{(C_S + C_{boot})} (V_{DD} - V_{OL}) \tag{9.31}$$

If the capacitor C_{boot} is much larger than C_S ($C_{boot} >> C_S$), the maximum value of V_x can be approximated as

$$V_x(max) = 2V_{DD} - V_{T3} - V_{OL} \tag{9.32}$$

which proves that voltage bootstrapping can significantly boost the voltage level V_x. Now remember that in order to overcome the threshold voltage drop at the output, the minimum required voltage level V_x is

$$\begin{aligned} V_x(min) &= V_{DD} + V_{T2}\big|_{V_{out}=V_{DD}} \\ &= \left(V_{DD} - V_{T3}(V_x)\right) + \frac{C_{boot}}{(C_S + C_{boot})}(V_{DD} - V_{OL}) \end{aligned} \tag{9.33}$$

This equation can be rearranged to give the required capacitance ratio, as follows.

$$\frac{C_{boot}}{(C_S + C_{boot})} = \frac{V_{T2}\big|_{V_{out}=V_{DD}} + V_{T3}\big|_{V_x}}{(V_{DD} - V_{OL})} \tag{9.34}$$

$$\frac{C_{boot}}{C_S} = \frac{V_{T2}\big|_{V_{out}=V_{DD}} + V_{T3}\big|_{V_x}}{V_{DD} - V_{OL} - V_{T2}\big|_{V_{out}=V_{DD}} - V_{T3}\big|_{V_x}} \tag{9.35}$$

Note that C_S is essentially the sum of the parasitic source-to-substrate capacitance of M3 and the gate-to-substrate capacitance of M2. To obtain a sufficiently large bootstrap capacitance C_{boot} in comparison to C_S, an extra "dummy" transistor is typically added to the circuit, as shown in Fig. 9.13.

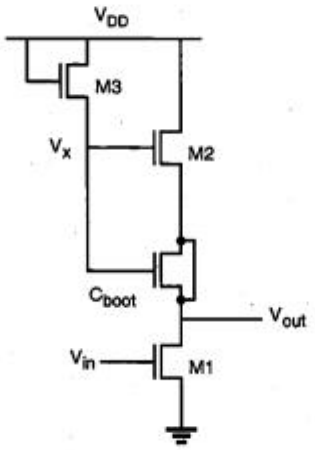


Figure 9.13. Realization of the bootstrapping capacitor with a dummy MOS device.

Since its drain and source terminals are connected together, the dummy transistor simply acts as an MOS capacitor between V_x and V_{out}. Although this circuit arrangement contains two additional transistors to achieve voltage bootstrapping, the resulting circuit-performance improvement is usually well worth the extra silicon area used for the bootstrapping devices.

Synchronous Dynamic Circuit Techniques

Having examined the basic concepts associated with temporary storage of logic levels in capacitive circuit nodes, we now turn our attention to digital circuit design techniques which take advantage of this simple yet effective principle. In the following, we will investigate different examples of synchronous dynamic circuits implemented using depletion-load nMOS, enhancement-load nMOS, and CMOS building blocks.

Dynamic Pass Transistor Circuits

Consider the generalized view of a multi-stage synchronous circuit shown in Fig. 9.14. The circuit consists of cascaded combinational logic stages, which are interconnected through nMOS pass transistors. All inputs of each combinational logic block are driven by a single clock signal. Individual input capacitances are not shown in this figure for simplicity, but the operation of the circuit obviously depends on temporary charge storage in the parasitic input capacitances.

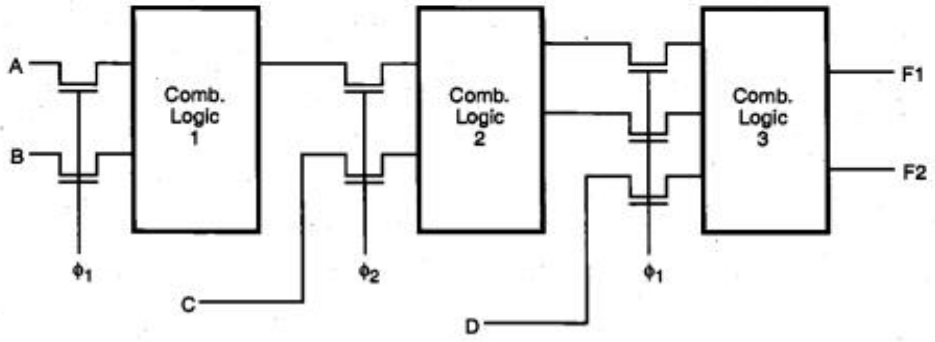


Figure 9.14. Multi-stage pass transistor logic driven by two nonoverlapping clocks.

To drive the pass transistors in this system, two nonoverlapping clock signals, ϕ_1 and ϕ_2, are used. The nonoverlapping property of the two clock signals guarantees that at any given time point, only one of the two clock signals can be active, as illustrated in Fig. 9.15. When clock ϕ_1 is active, the input levels of Stage 1 (and also of Stage 3) are applied through the pass transistors, while the input capacitances of Stage 2 retain their previously set logic levels. During the next phase, when clock ϕ_2 is active, the input levels of Stage 2 will be applied through the pass transistors, while the input capacitances of Stage 1 and Stage 3 retain their logic levels. This allows us to incorporate the simple dynamic memory function at each stage input, and at the same time, to facilitate synchronous operation by controlling the signal flow in the circuit using the two periodic clock signals. This signal timing scheme is also called *two-phase clocking* and is one of the most widely used timing strategies.

By introducing the two-phase clocking scheme, we have not made any specific assumptions about the internal structure of the combinational logic stages. It will be seen that depletion-load nMOS, enhancement-load nMOS, or CMOS logic circuits can be used for implementing the combinational logic. Figure 9.16 shows a depletion-load dynamic shift register circuit, in which the input data are inverted once and transferred, or *shifted* into the next stage during each clock phase.

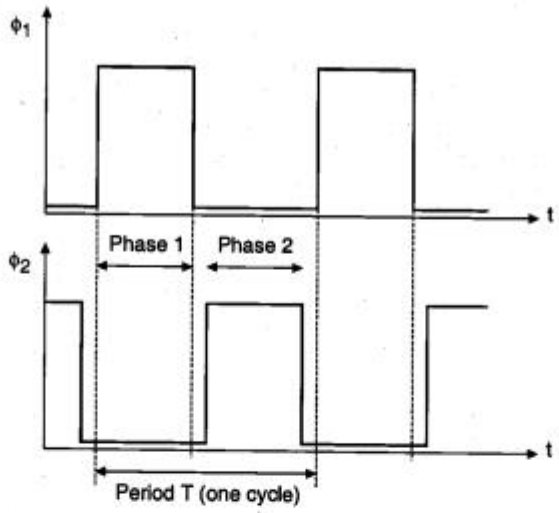


Figure 9.15. Nonoverlapping clock signals used for two-phase synchronous operation.

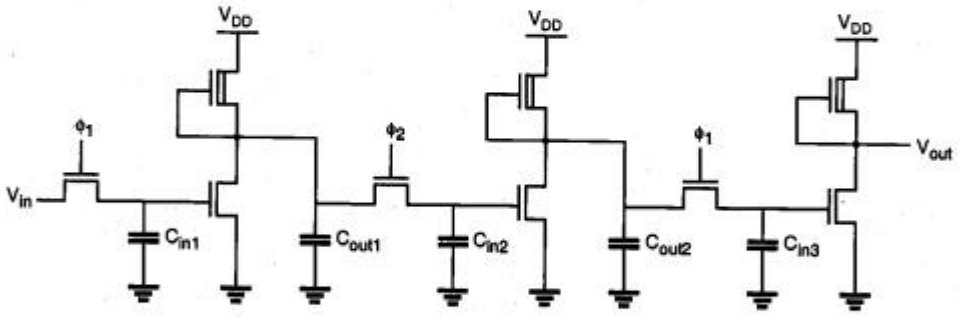


Figure 9.16. Three stages of a depletion-load nMOS dynamic shift register circuit driven with two-phase clocking.

The operation of the shift register circuit is as follows. During the active phase of ϕ_1, the input voltage level V_{in} is transferred into the input capacitance C_{in1}. Thus, the valid output voltage level of the first stage is determined as the inverse of the current input during this cycle. When ϕ_2 becomes active during the next phase, the output voltage level of the first stage is transferred into the second stage input capacitance C_{in2}, and the valid output voltage level of the second stage is determined. During the active ϕ_2 phase, the first-stage input capacitance continues to retain its previous level via charge storage. When ϕ_1 becomes active again, the original data bit *written* into the register during the previous cycle is transferred into the third stage, and the first stage can now accept the next data bit.

In this circuit, the maximum clock frequency is determined by the signal propagation delay through one inverter stage. One half-period of the clock signal must be long enough to allow the input capacitance C_{in} to charge up or down, and the logic level to propagate to the output by charging C_{out}. Also notice that the logic-high input level of each inverter stage in this circuit is one threshold voltage lower than the power supply voltage level.

The same operation principle used in the simple shift register circuit can easily be extended to synchronous complex logic. Figures 9.17 and 9.18 show a two-stage circuit example implemented using depletion-load nMOS complex logic gates.

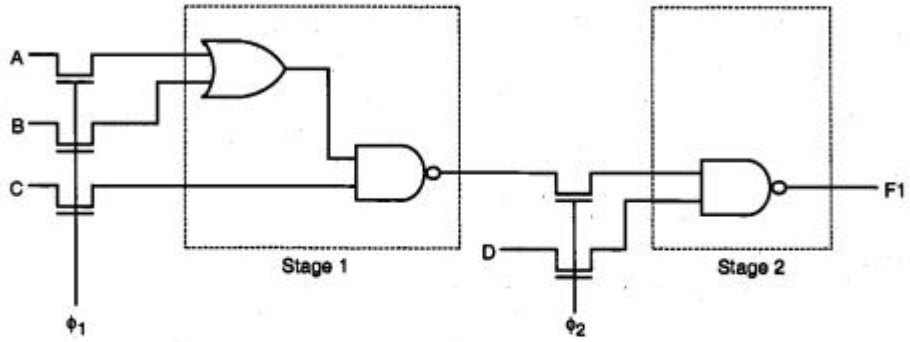


Figure 9.17. A two-stage synchronous complex logic circuit example.

In a complex logic circuit such as the one shown in Fig. 9.18, we see that the signal propagation delay of each stage may be different. Thus, in order to guarantee that correct logic levels are propagated during each active clock cycle, the half-period length of the clock signal must be longer than the largest single-stage signal propagation delay found in the circuit.

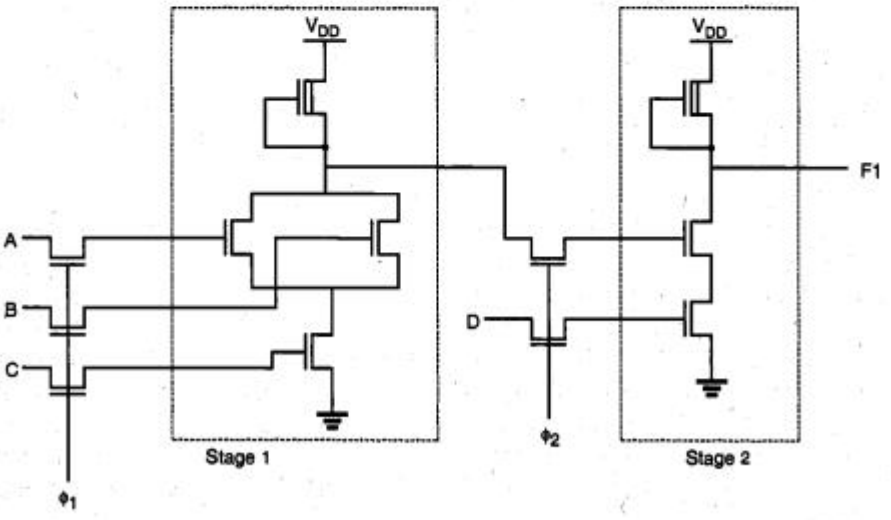


Figure 9.18. Depletion-load nMOS implementation of synchronous complex logic.

Now consider a different implementation of the simple shift register circuit, using enhancement-load nMOS inverters. One important difference is that, instead of biasing the load transistors with a constant gate voltage, we apply the clock signal to the gate of the load transistor as well. It can be shown that the power dissipation and the silicon area can be reduced significantly by using this dynamic (clocked) load approach. Two variants of the dynamic enhancement-load shift register will be examined in the following, both of which are driven by two non-overlapping clock signals. Figure 9.19 shows the first implementation, where in each stage the input pass transistor and the load transistor are driven by opposite clock phases, ϕ_1 and ϕ_2.

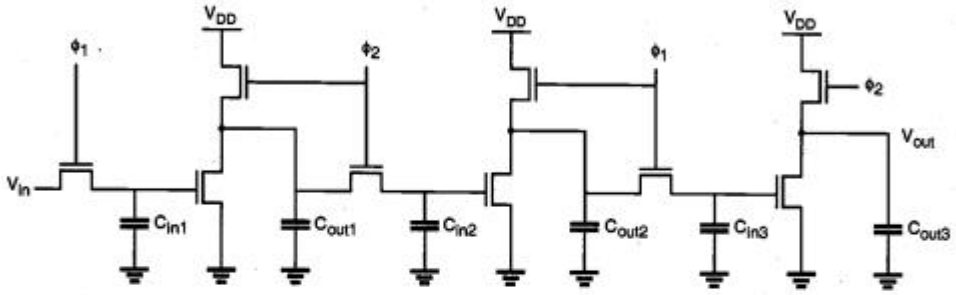


Figure 9.19. Enhancement-load dynamic shift register (ratioed logic).

When ϕ_1 is active, the input voltage level V_{in} is transferred into the first-stage input capacitance C_{in1} through the pass transistor. In this phase, the enhancement-type nMOS load transistor of the first-stage inverter is not active yet. During the next phase (active ϕ_2), the load transistor is turned on. Since the input logic level is still being preserved in C_{in1}, the output of the first inverter stage attains its valid logic level. At the same time, the input pass transistor of the second stage is also turned on, which allows this newly determined output level to be transferred into the input capacitance C_{in2} of the second stage. When clock ϕ_1 becomes active again, the valid output level across C_{out2} is determined, and transferred into C_{in3}. Also, a new input level can be accepted (*pipelined*) into C_{in1} during this phase.

In this circuit, the valid low-output voltage level V_{OL} of each stage is strictly determined by the driver-to-load ratio, since the output pass transistor (input pass transistor of next stage) turns on in phase with the load transistor. Therefore, this circuit arrangement is also called *ratioed dynamic logic*. The basic operation principle can obviously be extended to arbitrary complex logic, as shown in Fig. 9.20. Since the power supply current flows only when the load devices are activated by the clock signal, the overall power consumption of dynamic enhancement-load logic is generally lower than for depletion-load nMOS logic.

Next, consider the second dynamic enhancement-load shift register implementation where, in each stage, the input pass transistor and the load transistor are driven by the same clock phase (Fig. 9.21).

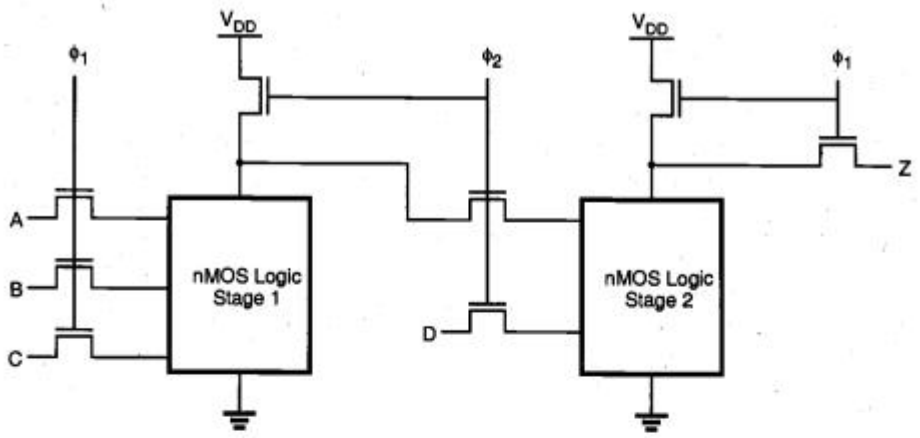


Figure 9.20. General circuit structure of ratioed synchronous dynamic logic.

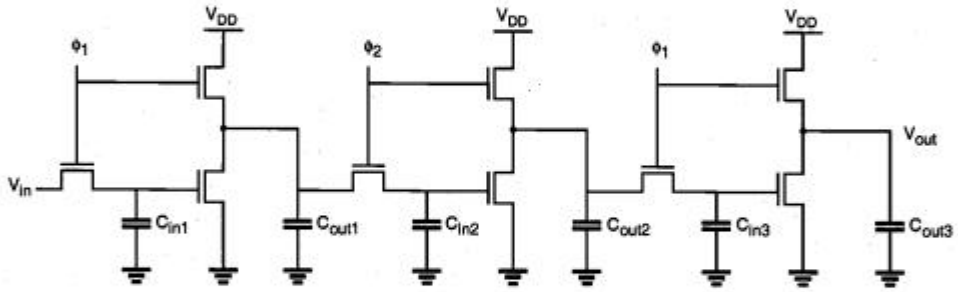


Figure 9.21. Enhancement-load dynamic shift register (ratioless logic).

When ϕ_1 is active, the input voltage level V_{in} is transferred into the first-stage input capacitance C_{in1} through the pass transistor. Note that at the same time, the enhancement-type nMOS load transistor of the first-stage inverter is active. Therefore, the output of the first inverter stage attains its valid logic level. During the next phase (active ϕ_2), the input pass transistor of the next stage is turned on, and the logic level is transferred onto the next stage. Here, we have to consider two cases, as follows.

If the output level across C_{out1} is logic-high at the end of the active ϕ_1 phase, this voltage level is transferred to C_{in2} via charge sharing over the pass transistor during the active ϕ_2 phase. Note that the logic-high level at the output node is subject to threshold voltage drop, i.e., it is one threshold voltage lower than the power supply voltage. To correctly transfer a logic-high level after charge sharing, the ratio of the capacitors (C_{out}/ C_{in}) must be made large enough during circuit design.

If, on the other hand, the output level of the first stage is logic-low at the end of the active ϕ_1 phase, then the output capacitor C_{out1} will be completely drained to a voltage of

$V_{OL} = 0$ V when ϕ_1 turns off. This can be achieved because a logic-high level is being stored in the input capacitance C_{in1} in this case, which forces the driver transistor to remain in conduction. Obviously, the logic-low level of $V_{OL} = 0$ V is also transferred into the next stage via the pass transistor during the active ϕ_2 phase.

When clock ϕ_1 becomes active again, the valid output level across C_{out2} is determined and transferred into C_{in3}. Also, a new input level can be accepted into C_{in1} during this phase. Since the valid logic-low level of $V_{OL} = 0$ V can be achieved regardless of the driver-to-load ratio, this circuit arrangement is called *ratioless dynamic logic*. The basic operation principle can be extended to arbitrary complex logic, as shown in Fig. 9.22.

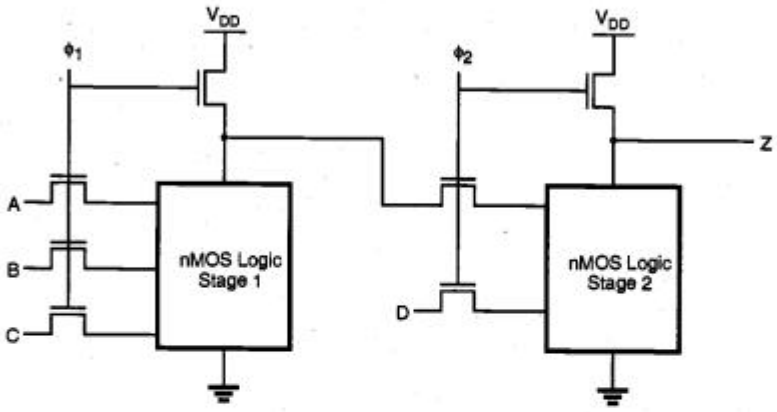


Figure 9.22. General circuit structure of ratioless synchronous dynamic logic.

CMOS Transmission Gate Logic

The basic two-phase synchronous logic circuit principle, in which individual logic blocks are cascaded via clock-controlled switches, can easily be adopted to CMOS structures as well. Here, static CMOS gates are used for implementing the logic blocks, and CMOS transmission gates are used for transferring the output levels of one stage to the inputs of the next stage (Fig. 9.23). Notice that each transmission gate is actually controlled by the clock signal *and* its complement. As a result, two-phase clocking in CMOS transmission gate logic requires that a total of four clock signals are generated and routed throughout the circuit.

As in the nMOS-based dynamic circuit structures, the operation of CMOS dynamic logic relies on charge storage in the parasitic input capacitances during the inactive clock cycles. To illustrate the basic operation principles, the fundamental building block of a dynamic CMOS transmission gate shift register is shown in Fig. 9.24. It consists of a CMOS inverter, which is driven by a CMOS transmission gate. During the active clock phase (CK =1), the input voltage V_{in} is transferred onto the parasitic input capacitance C_x via the transmission gate. Note that the low on-resistance of the CMOS transmission gate usually results in a smaller transfer time compared to those for nMOS-only switches. Also, there is no threshold voltage drop across the CMOS transmission gate. When the

clock signal becomes inactive, the CMOS transmission gate turns off and the voltage level across C_x can be preserved until the next cycle.

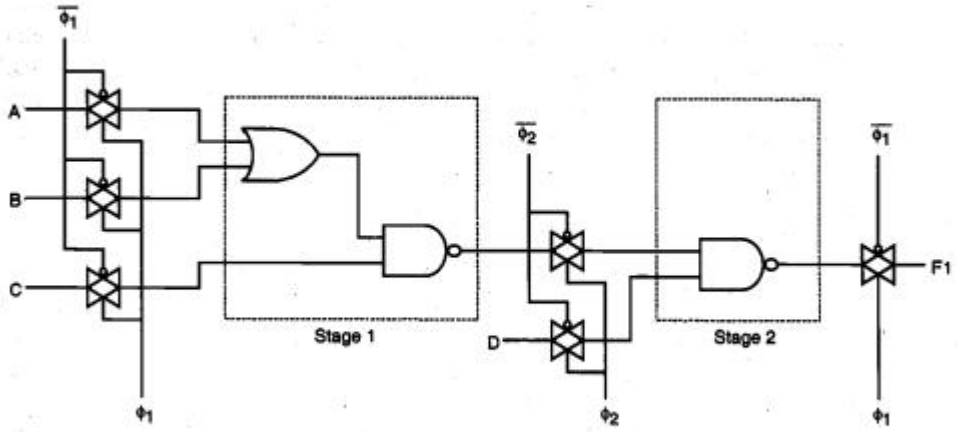


Figure 9.23. Typical example of dynamic CMOS transmission gate logic.

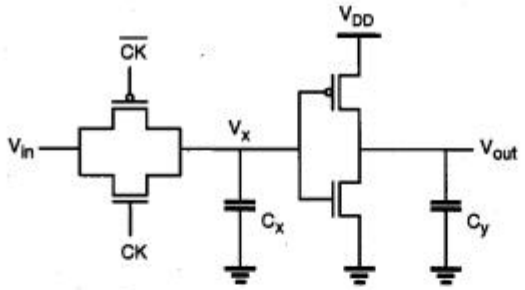


Figure 9.24. Basic building block of a CMOS transmission gate dynamic shift register.

Figure 9.25 shows a single-phase CMOS shift register, which is built by cascading identical units as in Fig. 9.24 and by driving each stage alternately with the clock signal and its complement.

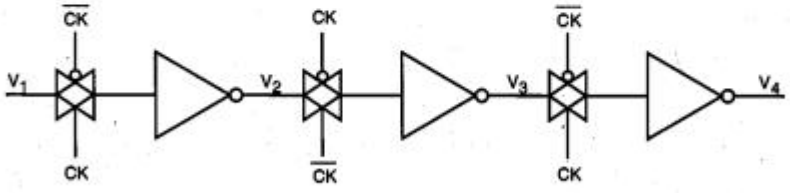


Figure 9.25. Single-phase CMOS transmission gate dynamic shift register.

Ideally, the transmission gates of the odd-numbered stages would conduct during the active clock phase (when CK = 1), while the transmission gates of the even-numbered stages are off, so that the cascaded inverter stages in the chain are alternately isolated. This would ensure that inputs are permitted in alternating half cycles. In practice, however, the clock signal and its complement do not constitute a truly nonoverlapping signal pair, since the clock voltage waveform has finite rise and fall times. Also, the clock skew between CK and $\overline{CK}$ may be unavoidable because one of the signals is generated by inverting the other. Therefore, true two-phase clocking with two nonoverlapping clock signals (ϕ_1 and ϕ_2) *and* their complements is usually preferred over single-phase clocking in dynamic CMOS transmission gate logic.

Dynamic CMOS Logic (Precharge-Evaluate Logic)

In the following, we will introduce a dynamic CMOS circuit technique which allows us to significantly reduce the number of transistors used to implement any logic function. The circuit operation is based on first *precharging* the output node capacitance and subsequently, *evaluating* the output level according to the applied inputs. Both of these operations are scheduled by a single clock signal, which drives one nMOS and one pMOS transistor in each dynamic stage. A dynamic CMOS logic gate which implements the function $F = \overline{(A_1A_2A_3 + B_1B_2)}$ is shown in Fig. 9.26.

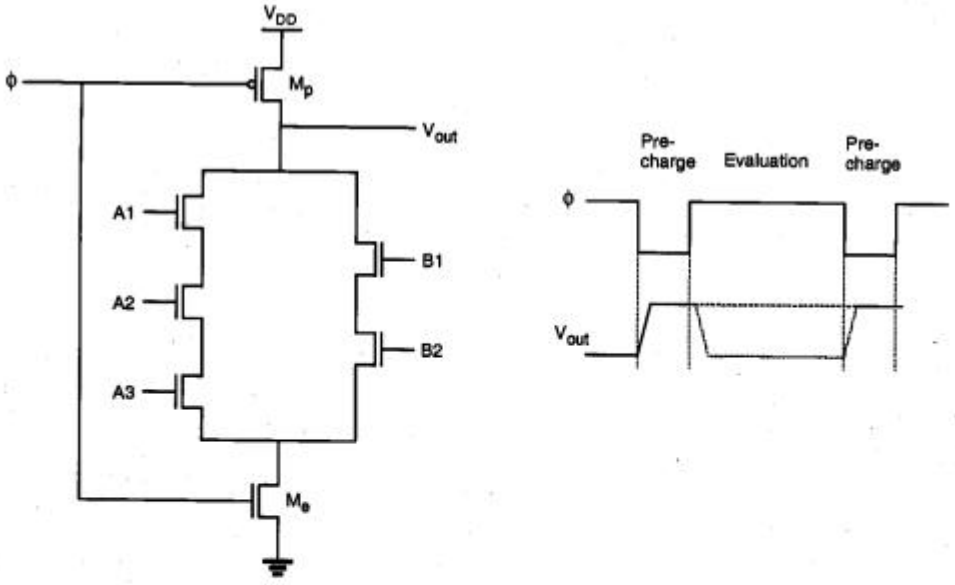


Figure 9.26. Dynamic CMOS logic gate implementing a complex Boolean function.

When the clock signal is low (precharge phase), the pMOS precharge transistor M_p is conducting, while the complementary nMOS transistor M_e is off. The parasitic output

capacitance of the circuit is charged up through the conducting pMOS transistor to a logic-high level of $V_{out} = V_{DD}$. The input voltages are also applied during this phase, but they have no influence yet upon the output level since M_e is turned off.

When the clock signal becomes high (evaluate phase), the precharge transistor M_p turns off and M_e turns on. The output node voltage may now remain at the logic- high level or drop to a logic low, depending on the input voltage levels. If the input signals create a conducting path between the output node and the ground, the output capacitance will discharge toward $V_{OL} = 0$ V. The final discharged output level depends on the time span of the evaluation phase. Otherwise, V_{out} remains at V_{DD}.

The operation of the single-stage dynamic CMOS logic gate is quite straightforward. For practical multi-stage applications, however, the dynamic CMOS gate presents a significant problem. To examine this fundamental limitation, consider the two-stage cascaded structure shown in Fig. 9.27. Here, the output of the first dynamic CMOS stage drives one of the inputs of the second dynamic CMOS stage, which is assumed to be a two-input NAND gate for simplicity.

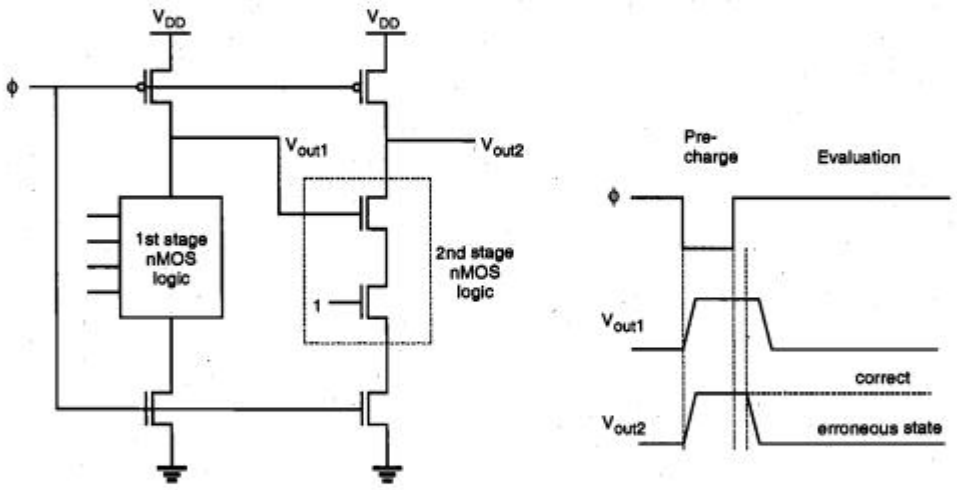


Figure 9.27. Illustration of the cascading problem in dynamic CMOS logic.

During the precharge phase, both output voltages V_{out1} and V_{out2} are pulled up by the respective pMOS precharge devices. Also, the external inputs are applied during this phase. The input variables of the first stage are assumed to be such that the output V_{out1} will drop to logic "0" during the evaluation phase. On the other hand, the external input of the second-stage NAND2 gate is assumed to be a logic "1," as shown in Fig. 9.27. When the evaluation phase begins, both output voltages V_{out1} and V_{out2} are logic-high. The output of the first stage (V_{out1}) eventually drops to its correct logic level after a certain time delay. However, since the evaluation in the second stage is done concurrently, starting with the high value of V_{out1} at the beginning of the evaluation phase, the output voltage V_{out2} at the end of the evaluation phase will be *erroneously* low. Although the first stage

output subsequently assumes its correct output value once the stored charge is drained, the correction of the second-stage output is not possible.

This example illustrates that dynamic CMOS logic stages driven by the same clock signal cannot be cascaded directly. This severe limitation seems to undermine all the other advantages of dynamic CMOS logic, such as low power dissipation, large noise margins, and low transistor count. Alternative clocking schemes and circuit structures must be developed to overcome this problem. In fact, the search for viable circuit alternatives has spawned a large array of high-performance dynamic CMOS circuit techniques, some of which will be examined in the following section.

High-Performance Dynamic CMOS Circuits

The circuits presented here are variants of the basic dynamic CMOS logic gate structure. We will see that they are designed to take full advantage of the obvious benefits of dynamic operation and at the same time, to allow unrestricted cascading of multiple stages. The ultimate goal is to achieve reliable, high-speed, compact circuits using the least complicated clocking scheme possible.

Domino CMOS Logic

Consider the generalized circuit diagram of a domino CMOS logic gate shown in Fig. 9.28. A dynamic CMOS logic stage, such as the one shown in Fig. 9.26, is cascaded with a static CMOS inverter stage. The addition of the inverter allows us to operate a number of such structures in cascade, as explained in the following.

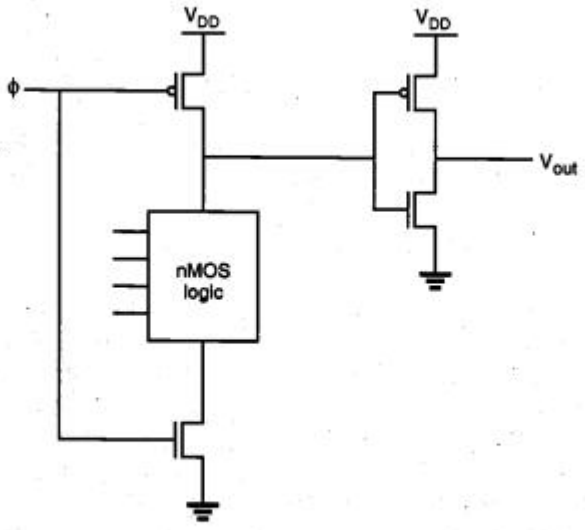


Figure 9.28. Generalized circuit diagram of a domino CMOS logic gate.

During the precharge phase (when CK = 0), the output node of the dynamic CMOS stage is precharged to a high logic level, and the output of the CMOS inverter (buffer) becomes low. When the clock signal rises at the beginning of the evaluation phase, there are two possibilities: The output node of the dynamic CMOS stage is either discharged to a low level through the nMOS circuitry (1 to 0 transition), or it remains high. Consequently, the inverter output voltage can also make at most one transition during the evaluation phase, from 0 to 1. Regardless of the input voltages applied to the dynamic CMOS stage, it is not possible for the buffer output to make a 1 to 0 transition during the evaluation phase.

Remember that the problem in cascading conventional dynamic CMOS stages occurs when one or more inputs of a stage make a 1 to 0 transition *during* the evaluation phase, as illustrated in Fig. 9.27. On the other hand, if we build a system by cascading domino CMOS logic gates as shown in Fig. 9.29, all input transistors in subsequent logic blocks will be turned off during the precharge phase, since all buffer outputs are equal to 0. During the evaluation phase, each buffer output can make at most one transition

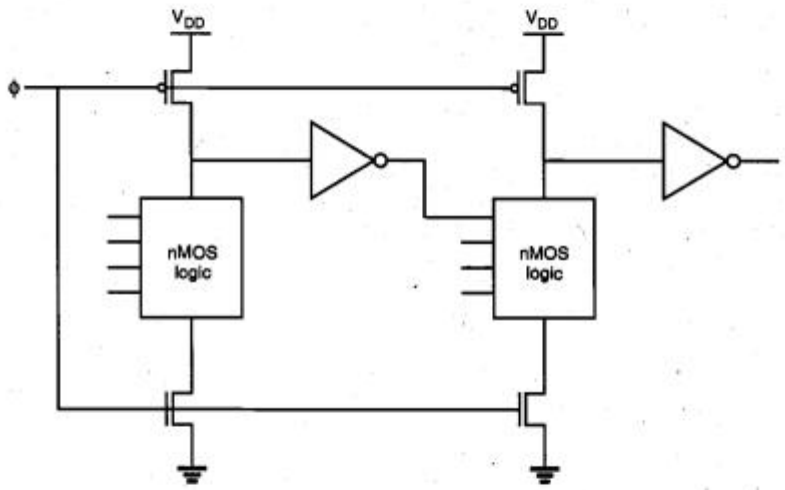


Figure 9.29. Cascaded domino CMOS logic gates.

(from 0 to 1), and thus each input of all subsequent logic stages can also make at most one (0 to 1) transition. In a cascade structure consisting of several such stages, the evaluation of each stage ripples the next stage evaluation, similar to a chain of dominos falling one after the other. The structure is hence called *domino CMOS logic*.

Domino CMOS logic gates allow a significant reduction in the number of transistors required to realize any complex Boolean function. The implementation of the 8-input Boolean function, $Z = AB + (C + D)(C + D) + GH$, using standard CMOS and domino CMOS, is shown in Fig. 9.30, where the reduction of circuit complexity is obvious. The distribution of the clock signal within the system is quite straightforward, since a single clock can be used to precharge and evaluate any number of cascaded stages, as long as the signal propagation delay from the first stage to the last stage does not exceed the time span of the evaluation phase. Also, conventional static CMOS logic gates can be used

together with domino CMOS gates in a cascaded configuration (Fig. 9.31). The limitation is that the number of inverting static logic stages in cascade must be even, so that the inputs of the next domino CMOS stage experience only 0 to 1 transitions during the evaluation.

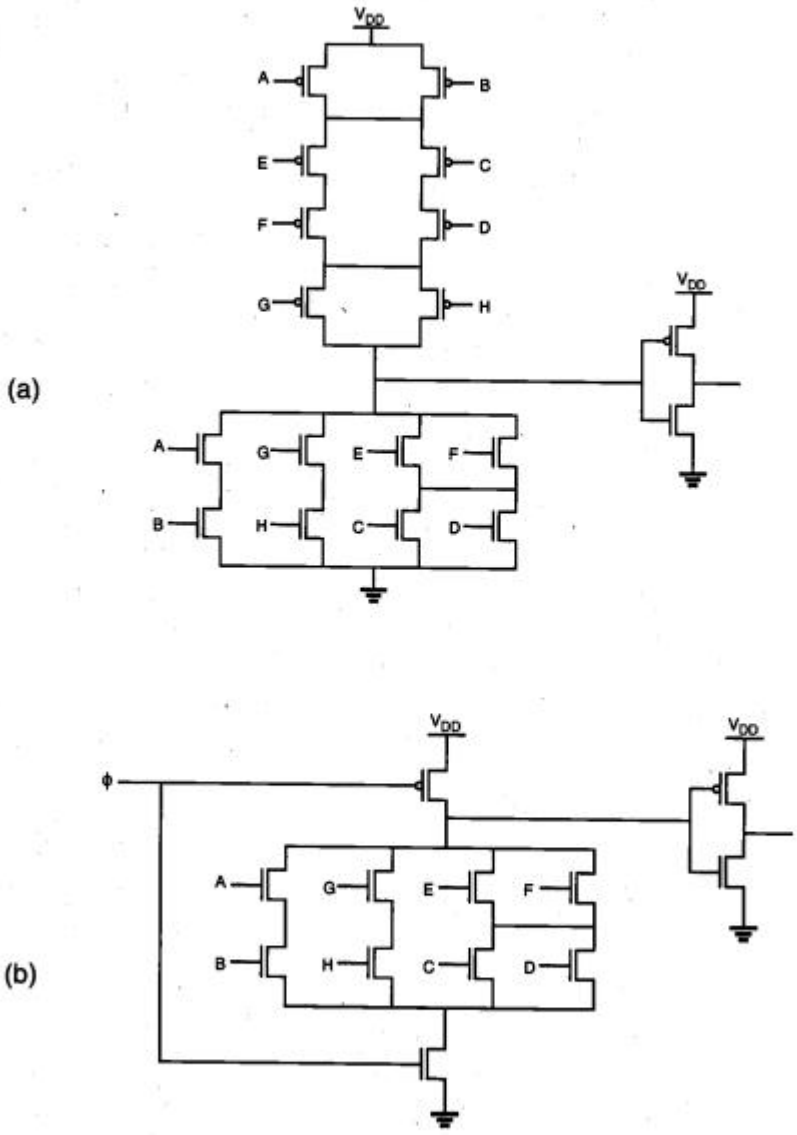


Figure 9.30. (a) An 8-input complex logic gate, realized using conventional CMOS logic and (b) domino CMOS logic.

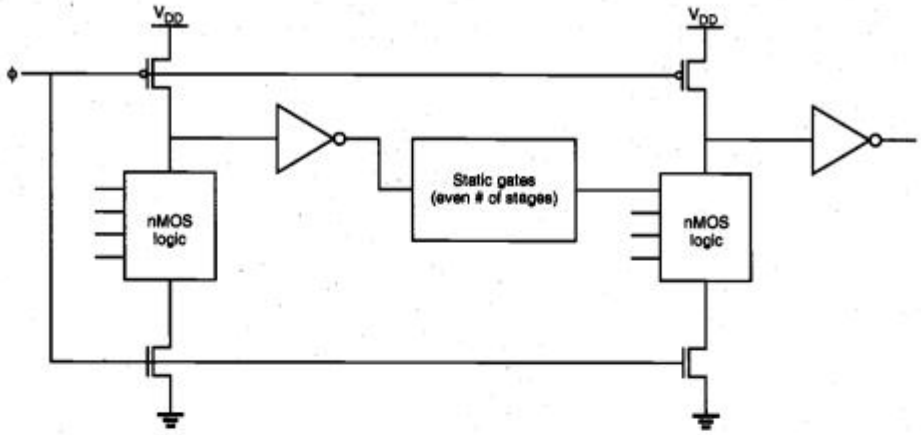


Figure 9.31. Cascading domino CMOS logic gates with static CMOS logic gates.

There are also some other limitations associated with domino CMOS logic gates. First, only non-inverting structures can be implemented using domino CMOS. If necessary, inversion must be carried out using conventional CMOS logic. Also, charge sharing between the dynamic stage output node and the intermediate nodes of the nMOS logic block during the evaluation phase may cause erroneous outputs, as will be explained in the following.

Consider the domino CMOS logic gate shown in Fig. 9.32, in which the intermediate node capacitance C_2 is comparable in size to the output node capacitance C_1. We will assume that all inputs are low initially, and that the intermediate node voltage across C_2 has an initial value of 0 V. During the precharge phase, the output node capacitance C_1 is charged up to its logic-high level of V_{DD} through the pMOS transistor. In the next phase, the clock signal becomes high and the evaluation begins. If the input signal of the uppermost nMOS transistor switches from low to high during this evaluation phase, as shown in Fig. 9.32, the charge initially stored in the output capacitance C_1 will now be shared by C_2, leading to the so-called *charge-sharing* phenomenon. The output node voltage after charge sharing becomes $V_{DD}/(1 + C_2/C_1)$. For example, if $C_1 = C_2$, the output voltage becomes $V_{DD}/2$ in the evaluation phase. Unless its logic threshold voltage is less than $V_{DD}/2$, the output voltage of the *following inverter* will then inadvertently switch high, which is a logic error. Thus, it is important to have C_2 much smaller than C_1.

Several measures can be taken in order to prevent erroneous output levels due to charge sharing in domino CMOS gates. One simple solution is to add a weak pMOS pull-up device (with a small (W/L) ratio) to the dynamic CMOS stage output, which essentially forces a high output level unless there is a strong pull-down path between the output and the ground (Fig. 9.33). It can be observed that the weak pMOS transistor will be turned on only when the precharge node voltage is kept high. Otherwise, it will be turned off as V_{out} becomes high.

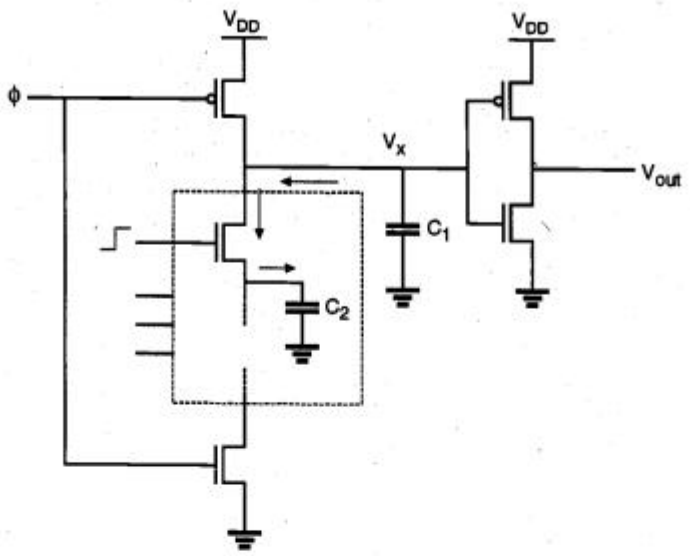


Figure 9.32. Charge sharing between the output capacitance C_1 and an intermediate node capacitance C_2 during the evaluation cycle may reduce the output voltage level.

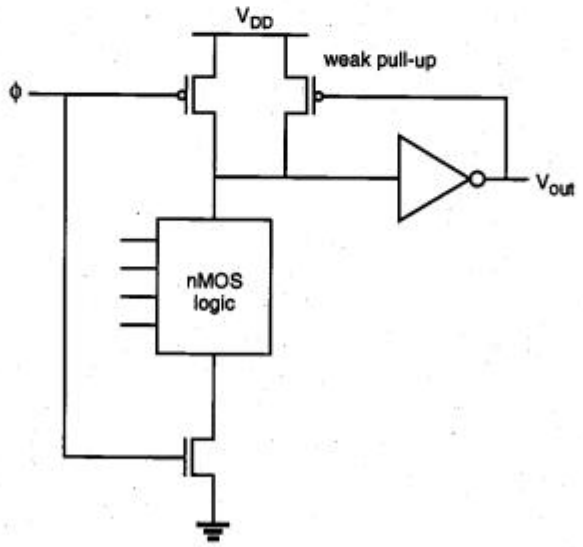


Figure 9.33. A weak pMOS pull-up device in a feedback loop can be used to prevent the loss of output voltage level due to charge sharing.

Another solution is to use separate pMOS transistors to precharge all intermediate nodes in the nMOS pull-down tree which have a large parasitic capacitance. The precharging of all high-capacitance nodes within the circuit effectively eliminates all potential charge-sharing problems during evaluation. However, it can also cause additional delay time since the nMOS logic tree now has to drain a larger charge in order to pull down the node voltage V_x. Another way of preventing logic errors due to charge sharing is to make the logic threshold voltage of the inverter smaller, such that the final stage output is not affected by lowering of V_x due to charge sharing. It should be noted that this design approach would trade off the pull-up speed (weaker pMOS transistor) for lower sensitivity to the charge-sharing problem.

The use of multiple precharge transistors also enables us to use the precharged intermediate nodes as resources for additional outputs. Thus, additional logic functions can be realized by tapping the internal nodes of the dynamic CMOS stage, as illustrated by two series-connected logic blocks in Fig. 9.34. The resulting multiple-output domino CMOS logic gate allows us to simultaneously realize several complex functions using a small number of transistors. Figure 9.35 shows the realization of four Boolean functions of nine variables, using a single domino CMOS logic gate. The four functions to be realized are listed in the following.

$$C_1 = G_1 + P_1 C_0$$
$$C_2 = G_2 + P_2 G_1 + P_2 P_1 C_0$$
$$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$$
$$C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$$

It can be shown that the functions C_1 through C_4 are the four carry terms to be used in a four-stage carry-lookahead adder, where the variables G_i and P_i are defined as

$$G_i = A_i \cdot B_i$$
$$P_i = A_i \oplus B_i$$

and A_i and B_i are the input bits associated with the i_{th} stage. Hence, this circuit is also known as the Manchester carry chain. The generation of the four carry terms using four separate standard CMOS logic gates or four separate single-output domino CMOS circuits, on the other hand, would require a large number of transistors and consequently a much larger silicon area. Variants of the multiple-output dynamic CMOS circuit shown in Fig. 9.35 are used widely in high-performance adder structures.

The transient performance of domino CMOS logic gates can be improved by adjusting the nMOS transistor sizes in the pull-down path, with the objective of reducing the discharge time. Shoji has shown that the best performance is obtained with a graded sizing of nMOS transistors in series structures, where the nMOS transistor closest to the output node also has the smallest (W/L) ratio. The domino CMOS circuit diagram and the corresponding stick-diagram layout of an optimized example are shown in Fig. 9.36. The fact that a graded reduction of transistor sizes from bottom to top ultimately leads to a

better transient performance may seem counterintuitive. But, this effect can be explained by observing the RC delay of the combined pull-down path consisting of series-connected nMOS transistors.

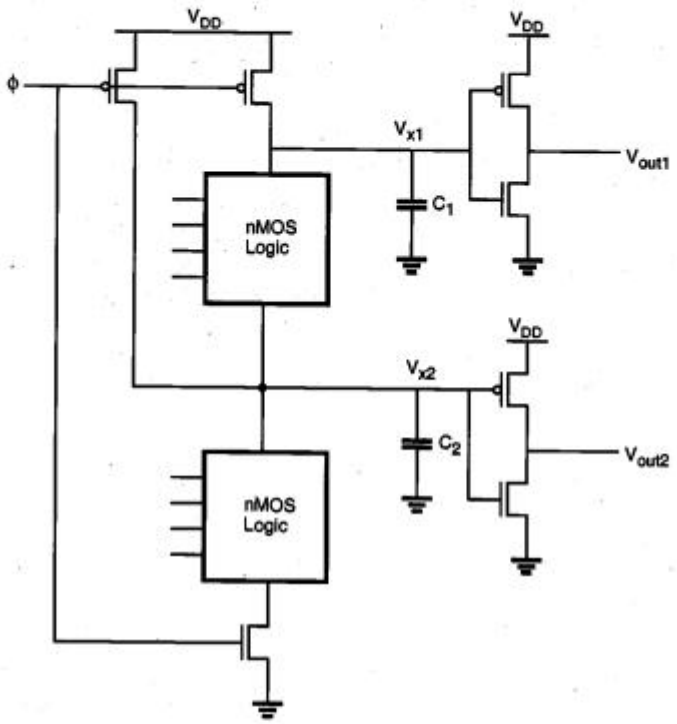


Figure 9.34. Precharging of internal nodes to prevent charge sharing also allows implementation of multiple-output domino CMOS structures.

Consider first the nMOS transistor closest to the output node. If the (W/L) ratio of this transistor is reduced by a certain factor, two effects occur. First, the current- driving capability will decrease, i.e., the equivalent resistance of the nMOS transistor will increase. Second, the parasitic drain capacitance associated with this transistor will decrease. If the length of the nMOS chain is sufficiently long, the increase in resistance has little influence upon the combined RC delay time, whereas a reduction of the capacitance significantly decreases the delay.

In fact, by applying Elmore's RC delay formula (see Chapter 6) to series-connected nMOS structures, one can determine if a reduction of nMOS transistor sizes will improve the transient performance. Let C_L represent the precharge-node load capacitance of the domino CMOS gate, and let C_1 represent the parasitic drain capacitance of the nMOS transistor closest to the precharge node. It is assumed that the inverter transistor sizes are fixed and that the pull-down chain contains N series-connected nMOS transistors. Shoji has shown that if the following condition

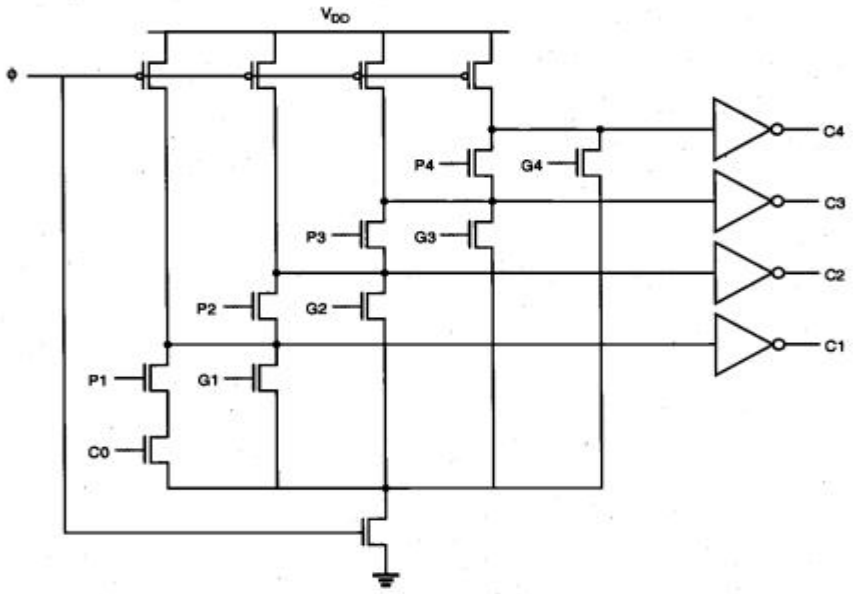


Figure 9.35. Example of a multiple-output domino CMOS gate realizing four functions.

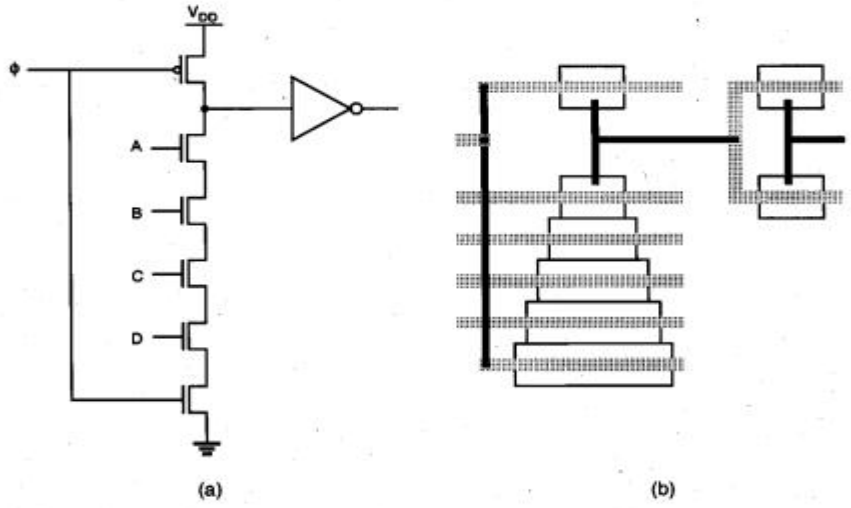


Figure 9.36. (a) Four-input domino CMOS NAND gate and (b) the corresponding stick-diagram layout to show the graded scaling of nMOS transistor sizes for improving the transient performance.

$$C_L < (N-1)\frac{C_1}{2} \tag{9.36}$$

is satisfied, the overall delay time can be reduced by decreasing the size of the nMOS transistor closest to the output node. This result can be iteratively applied to the other transistors in the pull-down chain, which leads to graded sizing of all nMOS devices. On the other hand, if the inverter gate transistors are allowed to be optimized along with the series-connected transistors, even shorter delays can be achieved with smaller chip area.

NORA CMOS Logic (NP-Domino Logic)

In domino CMOS logic gates, all logic operations are performed by the nMOS transistors acting as pull-down networks, while the role of pMOS transistors is limited to precharging

the dynamic nodes. As an alternative and a complement to nMOS-based domino CMOS logic, we can construct dynamic logic stages using pMOS transistors as well. Consider the circuit shown in Fig. 9.37, with alternating nMOS and pMOS logic stages.

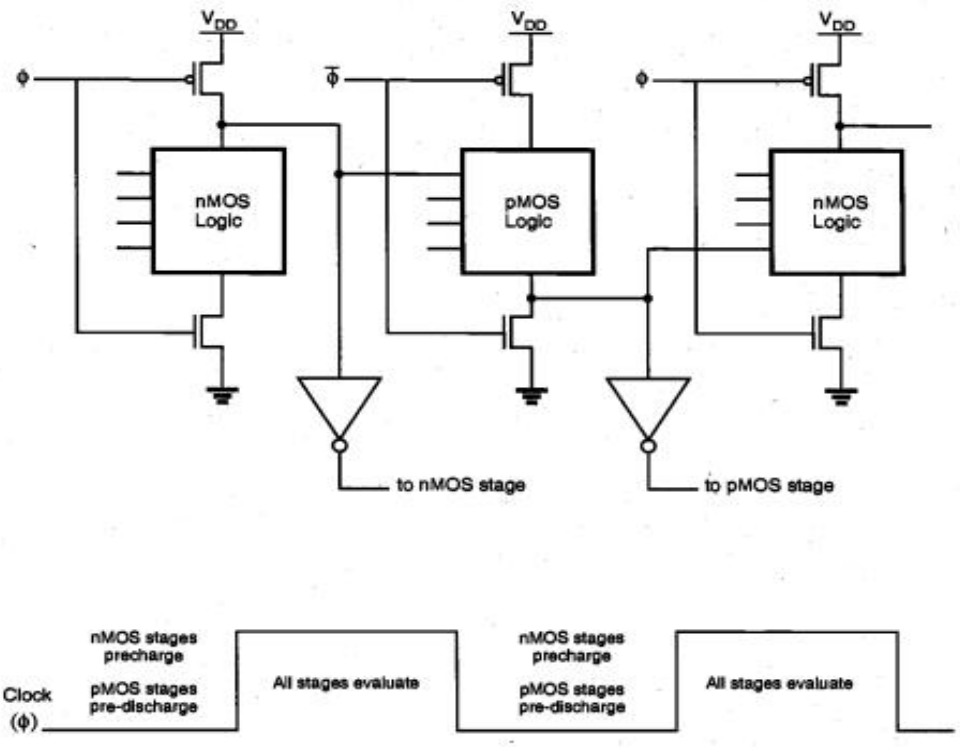


Figure 9.37. NORA CMOS logic consisting of alternating nMOS and pMOS stages, and the scheduling of precharge/evaluation phases.

Note that the precharge-and-evaluate timing of nMOS logic stages is accomplished by the clock signal ϕ, whereas the pMOS logic stages are controlled by the inverted clock signal, $\overline{\phi}$. The operation of the NORA CMOS circuit is as follows: When the clock signal is low, the output nodes of nMOS logic blocks are precharged to V_{DD} through the pMOS precharge transistors, whereas the output nodes of pMOS logic blocks are pre-discharged to 0 V through the nMOS discharge transistors, driven by $\overline{\phi}$. When the clock signal makes a low-to-high transition (note that the inverted clock signal $\overline{\phi}$ makes a high-to-low transition simultaneously), all cascaded nMOS and pMOS logic stages evaluate one after the other, much like the domino CMOS examined earlier. A simple NORA CMOS circuit example is shown in Fig. 9.38.

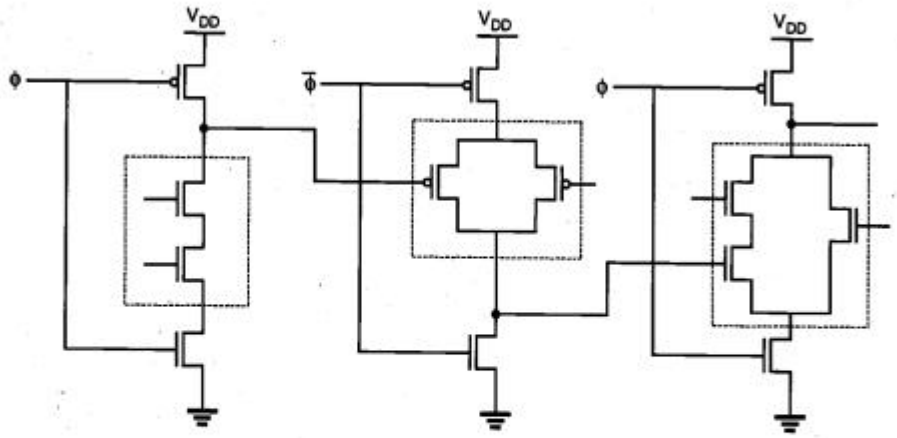


Figure 9.38. NORA CMOS logic circuit example.

The advantage of NORA CMOS logic is that a static CMOS inverter is not required at the output of every dynamic logic stage. Instead, direct coupling of logic blocks is feasible by alternating nMOS and pMOS logic blocks. NORA logic is also compatible with domino CMOS logic. Outputs of NORA nMOS logic blocks can be inverted, and then applied to the input of a domino CMOS block, which is also driven by the clock signal ϕ. Similarly, the buffered output of a domino CMOS stage can be applied directly to the input of a NORA nMOS stage.

The second important advantage of NORA CMOS logic is that it allows pipelined system architecture. Consider the circuit shown in Fig. 9.39(a), which consists of an nMOS-pMOS logic sequence similar to the one shown in Fig. 9.37, and a clocked CMOS (C^2MOS) output buffer. It can easily be seen that all stages of this circuit perform the precharge-discharge operation when the clock is low, and all stages of the circuit evaluate output levels when the clock is high. Therefore, we will call this circuit a ϕ-section, meaning that evaluation occurs during active ϕ.

Now consider the circuit shown in Fig. 9.39(b), which is essentially the same circuit shown in Fig. 9.39(a), with the only difference being that the signals ϕ and $\overline{\phi}$ have been interchanged. In this circuit, all logic stages perform the precharge-discharge operation when the clock is high, and all stages evaluate output levels when the clock is low. Therefore, we will call this circuit a $\overline{\phi}$-section, meaning that evaluation occurs during active $\overline{\phi}$.

A pipelined system can be constructed by simply cascading alternating ϕ- and $\overline{\phi}$-sections, as shown in Fig. 9.39(c). Note that each of the sections may consist of several logic stages, and that all logic stages in one section are evaluated during the same clock cycle. When the clock is low, the ϕ-sections in the pipelined system undergo the precharge cycle, while the $\overline{\phi}$-sections undergo evaluation. When the clock signal changes from low to high, the ϕ-sections start the evaluation cycle, while the $\overline{\phi}$-sections undergo precharge. Thus, consecutive sets of input data can be processed in alternating sections of the pipelined system.

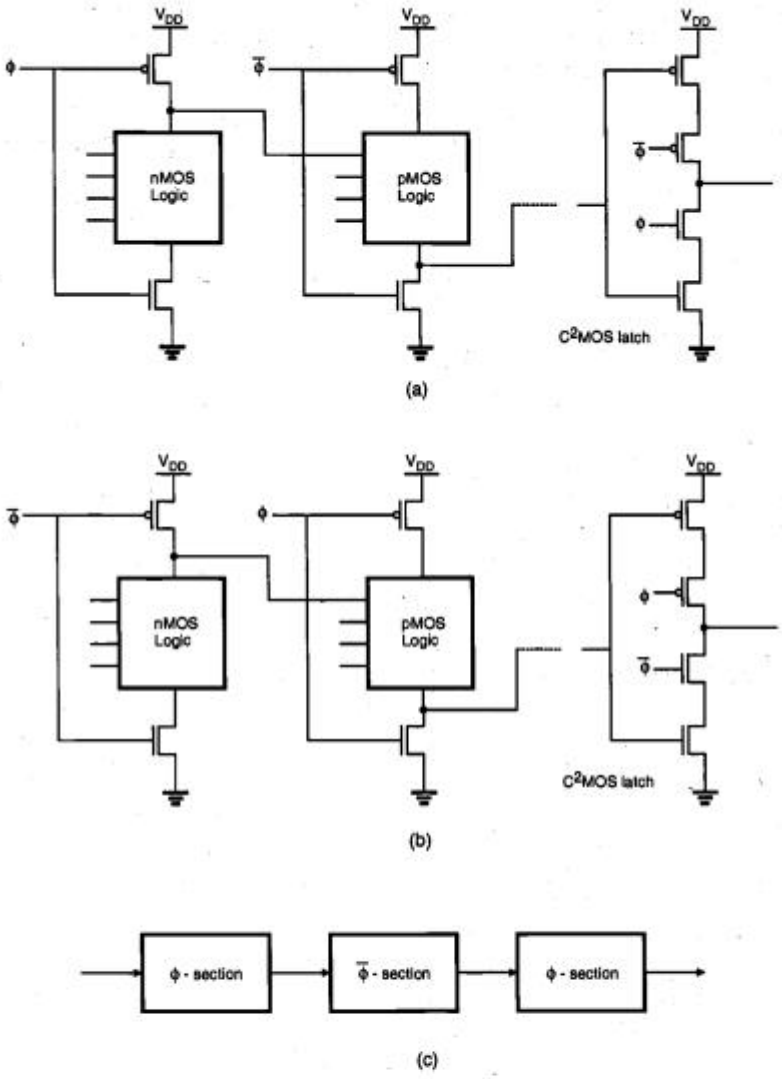


Figure 9.39. (a) NORA CMOS ϕ-section; evaluation occurs during $\phi = 1$. (b) NORA CMOS ϕ-section; evaluation occurs during $\phi = 0$. (c) A pipelined NORA CMOS system.

As in all dynamic CMOS structures, NORA CMOS logic gates also suffer from charge sharing and leakage. To overcome the dynamic charge sharing and soft- node leakage problems in NORA CMOS structures, a circuit technique called Zipper CMOS can be used.

Zipper CMOS Circuits

The basic circuit architecture of Zipper CMOS is essentially identical to NORA CMOS, with the exception of the clock signals. The Zipper CMOS clock scheme requires the generation of slightly different clock signals for the precharge (discharge) transistors and for the pull-down (pull-up) transistors. In particular, the clock signals which drive the pMOS precharge and nMOS discharge transistors allow these transistors to remain in weak conduction or near cut-off during the evaluation phase, thus compensating for the charge leakage and charge-sharing problems. The generalized circuit diagram and the clock signals of the Zipper CMOS architecture are shown in Fig. 9.40.

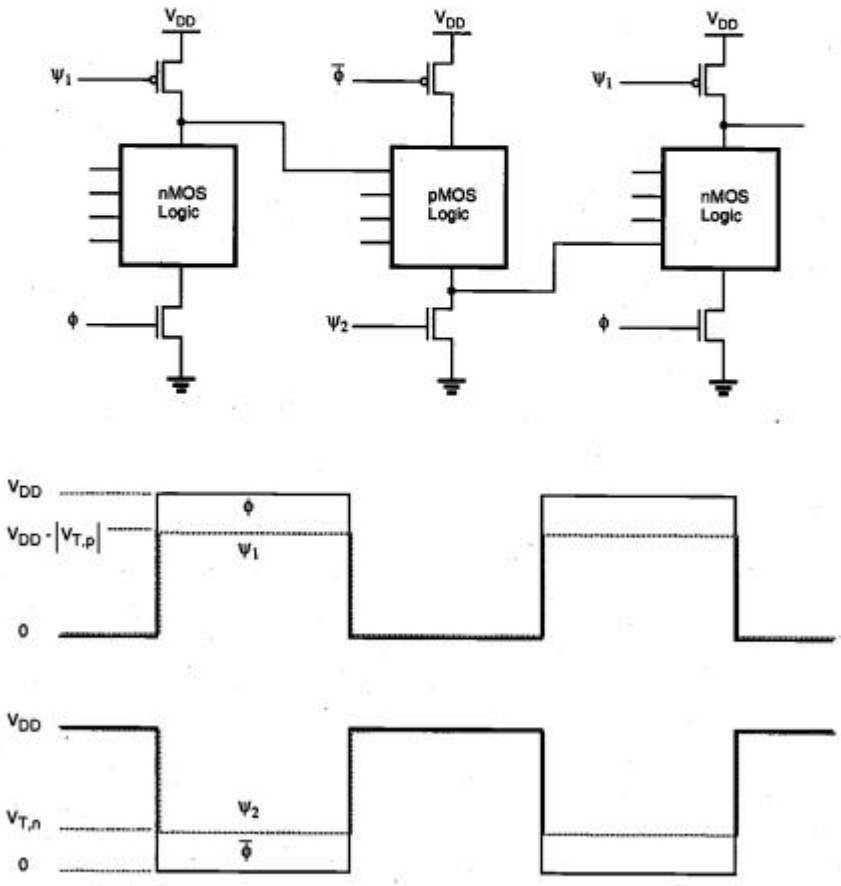


Figure 9.40. General circuit structure and the clock signals of Zipper CMOS.

True Single-Phase Clock (TSPC) Dynamic CMOS

The dynamic circuit technique to be presented in the following is distinctly different from the NORA CMOS circuit architecture in that it uses only one clock signal which is never inverted. Since the inverted clock signal ϕ is not used anywhere in the system, no clock skew problem exists. Consequently, higher clock frequencies can be reached for dynamic pipelined operation.

Consider the circuit diagram shown in Fig. 9.41. The circuit consists of alternating stages called n-blocks and p-blocks, and each block is being driven by the same clock signal ϕ. An n-block is constructed by cascading a dynamic nMOS stage and a dynamic latch, while a p-block is constructed by cascading a dynamic pMOS stage and a dynamic latch.

When the clock signal is low, the output node of the n-block is being precharged to V_{DD} by the pMOS precharge transistor. When the clock signal switches from low to high, the logic stage output is evaluated and the output latch generates a valid output level. On the other hand, we can see by inspection that the p-block pre-discharges when the clock is high, and evaluates when the clock is low. This means that a cascade-connection of alternating n-block and p-block circuits as shown in Fig. 9.41 will allow pipelined operation using a single clock signal. Compared to NORA CMOS, we need two extra transistors per stage, but the ability to operate with a true single-phase clock signal offers very attractive possibilities from the system-design point of view.

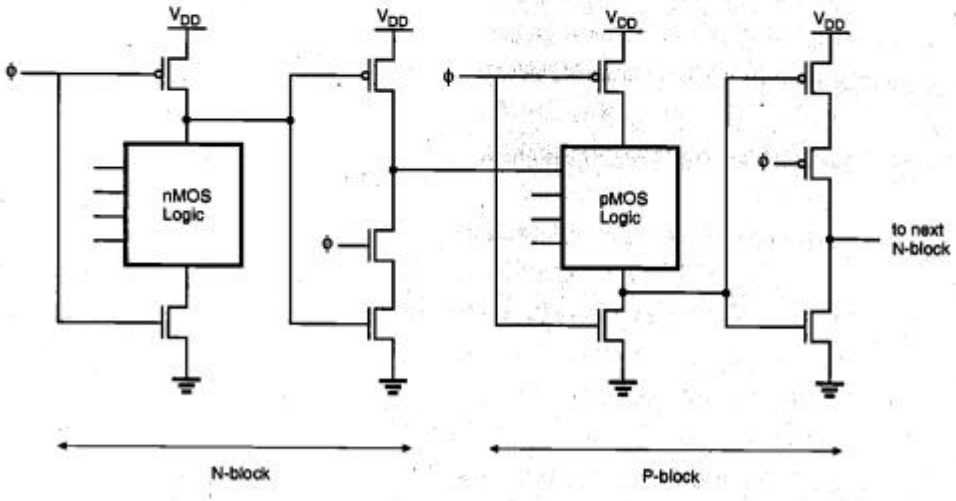


Figure 9.41. A pipelined true single-phase clock CMOS system.

Figure 9.42 shows the circuit diagram of a rising edge-triggered D-type flip-flop (DFF) which is constructed with the TSPC principle. The circuit consists of eleven transistors, in four simple stages. When the clock signal is low, the first stage acts as a transparent latch to receive the input signal, while the output node of the second stage is

being precharged. During this cycle, the third and fourth stages simply keep the previous output state. When the clock signal switches from low to high, the first stage ceases to be transparent and the second stage starts evaluation. At the same time, the third stage becomes transparent and transmits the sampled value to the output. Note that the final stage (inverter) is only used to obtain the non-inverted output level.

The mask layout of this circuit (using 0.8 μm CMOS technology design rules) is shown in Fig. 9.43. The device dimensions are $(W/L)_n$ = (2 μm/0.8 μm) for nMOS transistors, and $(W/L)_p$ = (5.6 μm/0.8 μm) for pMOS transistors. The layout parasitics were determined through circuit extraction, and the extracted circuit file was simulated with SPICE to determine its performance. The simulated input and output waveforms of the TSPC-based DFF circuit are given in Fig. 9.44. It can be seen that the circuit is capable of operating with a clock frequency of 500 MHz. With their relatively simple design, low transistor count and high speed, TSPC-based circuits offer a very favorable alternative to conventional CMOS circuits, especially in high-performance designs.

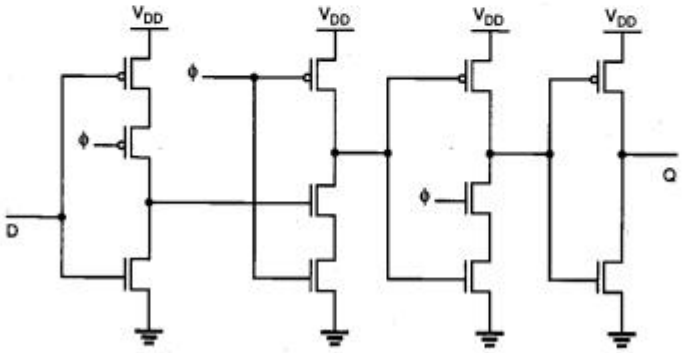


Figure 9.42. Circuit diagram of a TSPC-based rising edge-triggered DFF.

UNIT V - SEMICONDUCTOR MEMORIES

Types of Memories

Memory circuits are generally classified according to the type of data storage and the type of data access. *Read-Only Memory* (ROM) circuits allow, as the name implies, only the retrieval of previously stored data and do not permit modifications of the stored information contents during normal operation. ROMs are *non-volatile* memories, i.e., the data storage function is not lost even when the power supply voltage is off. Depending on the type of data storage (data write) method, ROMs are classified as mask-programmed ROMs, Programmable ROMs (PROM), Erasable PROMs (EPROM), and Electrically Erasable PROMs (EEPROM).

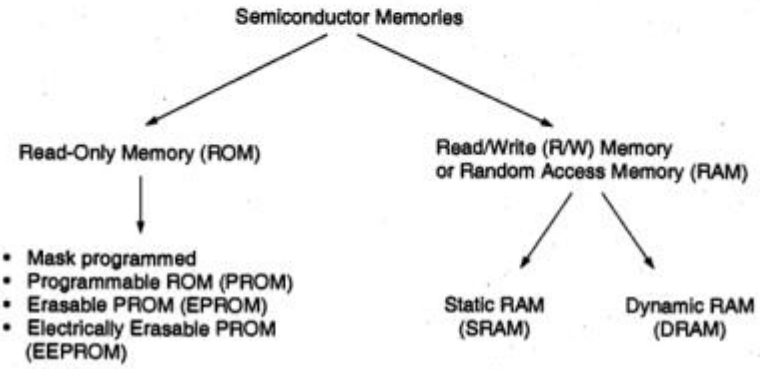


Figure 10.1. Overview of semiconductor memory types.

Read-write (R/W) memory circuits, on the other hand, must permit the modification (writing) of data bits stored in the memory array, as well as their retrieval (reading) on demand. This requires that the data storage function be *volatile*, i.e., the stored data are lost when the power supply voltage is turned off. The read-write memory circuit is commonly called *Random Access Memory* (RAM), mostly due to historical reasons. Compared to sequential-access memories such as magnetic tapes, any cell in the R/W memory array can be accessed with nearly equal access time. Based on the operation type of individual data storage cells, RAMs are classified into two main categories: *Static* RAMs (SRAM) and *Dynamic* RAMs (DRAM). Figure 10.1 shows an overview of the different memory types and their classifications.

A typical memory array organization is shown in Fig. 10.2. The data storage structure, or *core*, consists of individual memory cells arranged in an array of horizontal rows and vertical columns. Each *cell* is capable of storing one bit of binary information. Also, each memory cell shares a common connection with the other cells in the same row, and another common connection with the other cells in the same column. In this structure, there are 2^N rows, also called *word lines*, and 2^M columns, also called *bit lines*. Thus, the total number of memory cells in this array is $2^M \times 2^N$.

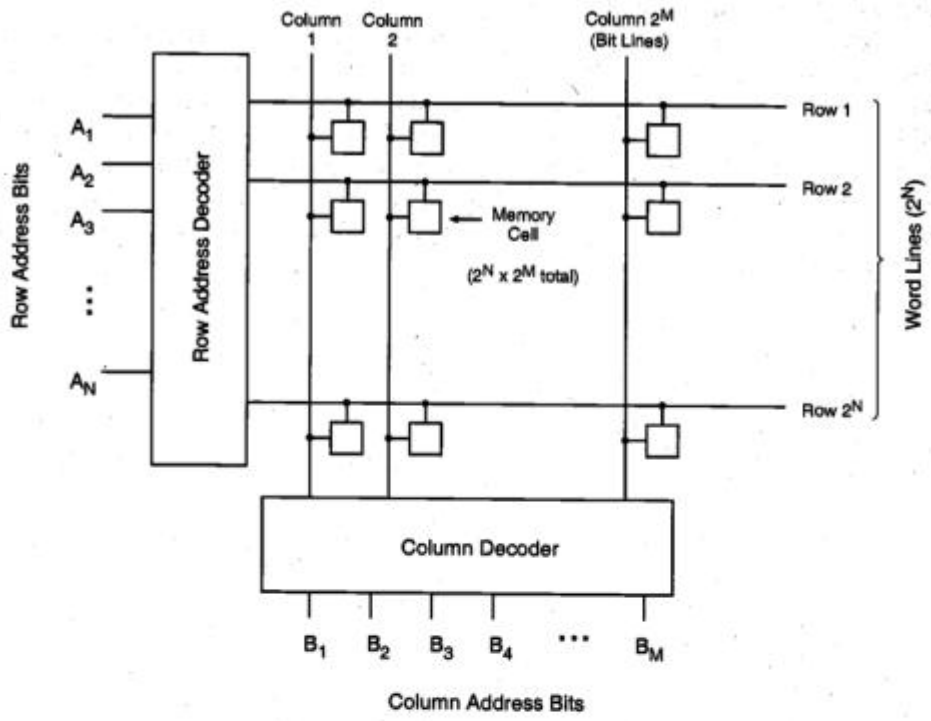


Figure 10.2. Typical random-access memory array organization.

To access a particular memory cell, i.e., a particular data bit in this array, the corresponding bit line *and* the corresponding word line must be activated (selected). The row and column selection operations are accomplished by row and column *decoders*, respectively. The row decoder circuit selects one out of 2^N word lines according to an N-bit row address, while the column decoder circuit selects one out of 2^M bit lines according to an M-bit column address. Once a memory cell or a group of memory cells are selected in this fashion, a data read and/or a data write operation may be performed on the selected single bit or multiple bits on a particular row. The column decoder circuit serves the double duties of selecting the particular columns and routing the corresponding data content in a selected row to the output.

We can see from this simple discussion that individual memory cells can be accessed for data read and/or data write operations in random order, independent of their physical locations in the memory array. Thus, the array organization examined here is called a *Random Access Memory* (RAM) structure. Notice that this organization can be used for both read-write memory arrays and read-only memory arrays. In the following sections, however, we will use the acronym RAM specifically for read-write memories, because it is the universally accepted abbreviation for this particular type of memory array.

Dynamic Read-Write Memory (DRAM) Circuits

All of the static RAM cells examined in the previous section consist of a two-inverter latch circuit, which is accessed for "read" and "write" operations via two pass transistors. Consequently, the SRAM cells require four to six transistors per bit, and four to five lines connecting to each cell, including the power and ground connections. To satisfy these requirements, a substantial silicon area must be reserved for each memory cell. In addition, most SRAM cells have non-negligible standby (static) power dissipation, with the exception of the full CMOS SRAM cell.

As the trend for high-density RAM arrays forces the memory cell size to shrink, alternative data storage concepts must be considered to accommodate these demands. In a *dynamic* RAM cell, binary data is stored simply as charge in a capacitor, where the presence or absence of stored charge determines the value of the stored bit. Note that the data stored as charge in a capacitor cannot be retained indefinitely, because the leakage currents eventually remove or modify the stored charge. Thus, all dynamic memory cells require a periodic refreshing of the stored data, so that unwanted modifications due to leakage are prevented before they occur.

The use of a capacitor as the primary storage device generally enables the DRAM cell to be realized on a much smaller silicon area compared to the typical SRAM cell. Notice that even as the binary data is stored as charge in a capacitor, the DRAM cell must have access devices, or switches, which can be activated externally for "read" and "write" operations. But this requirement does not significantly affect the area advantage over the SRAM cell, since the cell access circuitry is usually very simple. Also, no static power is dissipated for storing charge on the capacitance. Consequently, dynamic RAM arrays can achieve higher integration densities than SRAM arrays. Note that a DRAM array also

processors wait until the SRAM is free. The added wait state, however, significantly reduces the advantages of the high-speed processor. The dual-port RAM architecture is implemented in systems in which a main memory array must serve multiple high-speed processors and peripheral devices with minimum delay.

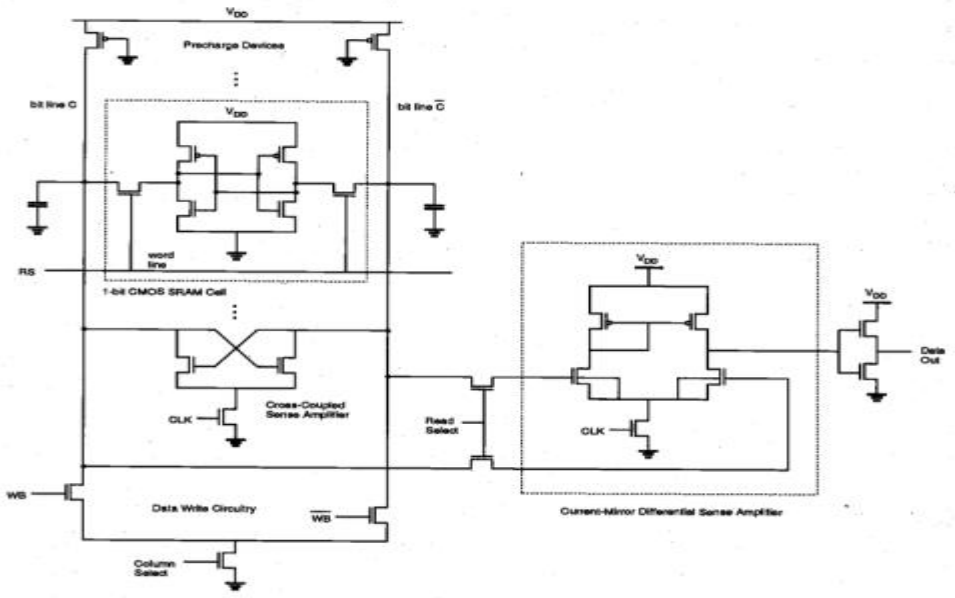


Figure 10.34. Complete circuit diagram of a CMOS static RAM column with data write and data read circuitry.

The ideal dual-port SRAM allows simultaneous access to the same location in the memory array, by using two independent sets of bit lines and associated access switches for each memory cell. The circuit structure of a typical CMOS dual-port SRAM cell is shown in Fig. 10.35. Here, "word line 1" is used to access one set of complementary bit lines (bit line 1), while "word line 2" allows access to the other set of bit lines (bit line 2). The capability of simultaneous access eliminates wait states for the processors during "data read" operations. However, contention may still occur if both external processors accessing the same memory location simultaneously attempt to write data onto the accessed cell, or if one of the processors attempts to read data while the other processor

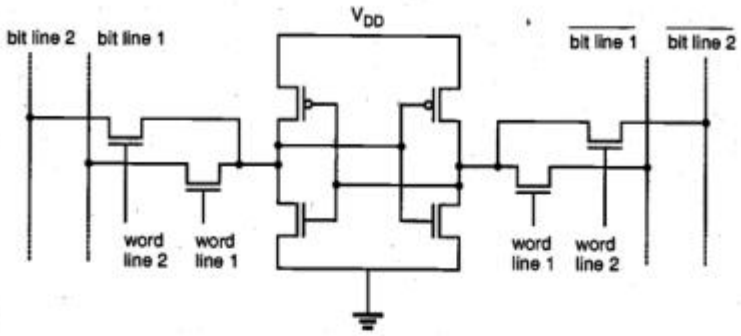


Figure 10.35. Circuit diagram of the CMOS dual-port SRAM cell.

writes data onto the same cell. In most cases, overlapping operations to the same memory location can be eliminated by a contention arbitration logic. It can either allow contention to be ignored and both operations to proceed, or it can arbitrate and delay one port until the operation on the other port is completed.

Dynamic Read-Write Memory (DRAM) Circuits

All of the static RAM cells examined in the previous section consist of a two-inverter latch circuit, which is accessed for "read" and "write" operations via two pass transistors. Consequently, the SRAM cells require four to six transistors per bit, and four to five lines connecting to each cell, including the power and ground connections. To satisfy these requirements, a substantial silicon area must be reserved for each memory cell. In addition, most SRAM cells have non-negligible standby (static) power dissipation, with the exception of the full CMOS SRAM cell.

As the trend for high-density RAM arrays forces the memory cell size to shrink, alternative data storage concepts must be considered to accommodate these demands. In a *dynamic* RAM cell, binary data is stored simply as charge in a capacitor, where the presence or absence of stored charge determines the value of the stored bit. Note that the data stored as charge in a capacitor cannot be retained indefinitely, because the leakage currents eventually remove or modify the stored charge. Thus, all dynamic memory cells require a periodic refreshing of the stored data, so that unwanted modifications due to leakage are prevented before they occur.

The use of a capacitor as the primary storage device generally enables the DRAM cell to be realized on a much smaller silicon area compared to the typical SRAM cell. Notice that even as the binary data is stored as charge in a capacitor, the DRAM cell must have access devices, or switches, which can be activated externally for "read" and "write" operations. But this requirement does not significantly affect the area advantage over the SRAM cell, since the cell access circuitry is usually very simple. Also, no static power is dissipated for storing charge on the capacitance. Consequently, dynamic RAM arrays can achieve higher integration densities than SRAM arrays. Note that a DRAM array also

requires additional peripheral circuitry for scheduling and performing the periodic data refresh operations. The hardware overhead of the refresh circuitry, however, does not overshadow the area advantages gained by the small cell size.

Figure 10.36 shows some of the steps in the historical evolution of the DRAM cell. The four-transistor cell shown in Fig. 10.36(a) is the simplest and one of the earliest dynamic memory cells. This cell is derived from the six-transistor static RAM cell by removing the load devices. The cell has in fact two storage nodes, i.e., the parasitic oxide

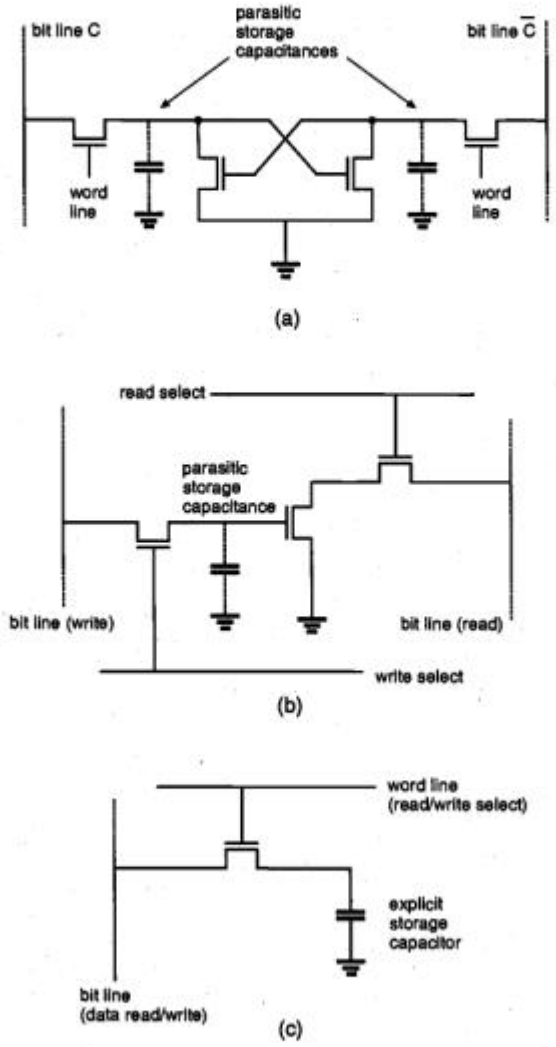


Figure 10.36. Various configurations of the dynamic RAM cell. (a) Four-transistor DRAM cell with two storage nodes. (b) Three-transistor DRAM cell with two bit lines and two word lines. (c) One-transistor DRAM cell with one bit line and one word line.

and diffusion capacitances of the nodes indicated in the circuit diagram. Since no current path is provided to the storage nodes for restoring the charge being lost to leakage, the cell must be refreshed periodically. It is obvious that the four-transistor dynamic RAM cell can have only a marginal area advantage over the six-transistor SRAM cell.

The three-transistor DRAM cell shown in Fig. 10.36(b) was the first widely used dynamic memory cell. It utilizes a single transistor as the storage device (where the transistor is turned on or off depending on the charge stored on its gate capacitance), and one transistor each for the "read" and "write" access switches. The cell has two control and two I/O lines. Its separate read and write select lines make it relatively fast, but the four lines with their additional contacts tend to increase the cell area.

The one-transistor DRAM cell shown in Fig. 10.36(c) has become the industry-standard dynamic RAM cell in high-density DRAM arrays. With only one transistor and one capacitor, it has the lowest component count and, hence, the smallest silicon area of all the dynamic memory cells. The cell has one read-write control line (word line) and one I/O line (bit line). We have to emphasize at this point that, unlike in the other dynamic memory cells shown in Figs. 10.36(a) and 10.36(b), the storage capacitance of the one-transistor DRAM cell is *explicit*. This means that a separate capacitor must be manufactured for each storage cell, instead of relying on the parasitic oxide and diffusion capacitances of the transistors for data storage. The word line of the one-transistor DRAM cell is controlled by the row address decoder. Once the selected transistor is turned on, the charge stored in the capacitor can be detected and/or modified through the bit line.

Before we examine the operation of the one-transistor DRAM cell, we will first investigate the three-transistor cell shown in Fig. 10.36(b), which has a very straightforward operation principle. This will help us to illuminate the basic issues involved in the design and operation of dynamic memory cells in general.

Three-Transistor DRAM Cell

The circuit diagram of a typical three-transistor dynamic RAM cell is shown in Fig. 10.37 as well as the column pull-up (precharge) transistors and the column read/write circuitry. Here, the binary information is stored in the form of charge in the parasitic node capacitance C_1. The storage transistor M2 is turned on or off depending on the charge stored in C_1, and the pass transistors M1 and M3 act as access switches for data read and write operations. The cell has two separate bit lines for "data read" and "data write," and two separate word lines to control the access transistors.

The operation of the three-transistor DRAM cell and its peripheral circuitry is based on a two-phase non-overlapping clock scheme. The precharge events are driven by ϕ_1, whereas the "read" and "write" events are driven by ϕ_2. Every "data read" and "data write" operation is preceded by a precharge cycle, which is initiated with the precharge signal PC going high. During the precharge cycle, the column pull-up transistors are activated, and the corresponding column capacitances C_2 and C_3 are charged up to logic-high level. With typical enhancement type nMOS pull-up transistors ($V_{T0} \approx 1.0$ V) and a power supply voltage of 5 V, the voltage level of both columns after the precharge is approximately equal to 3.5 V.

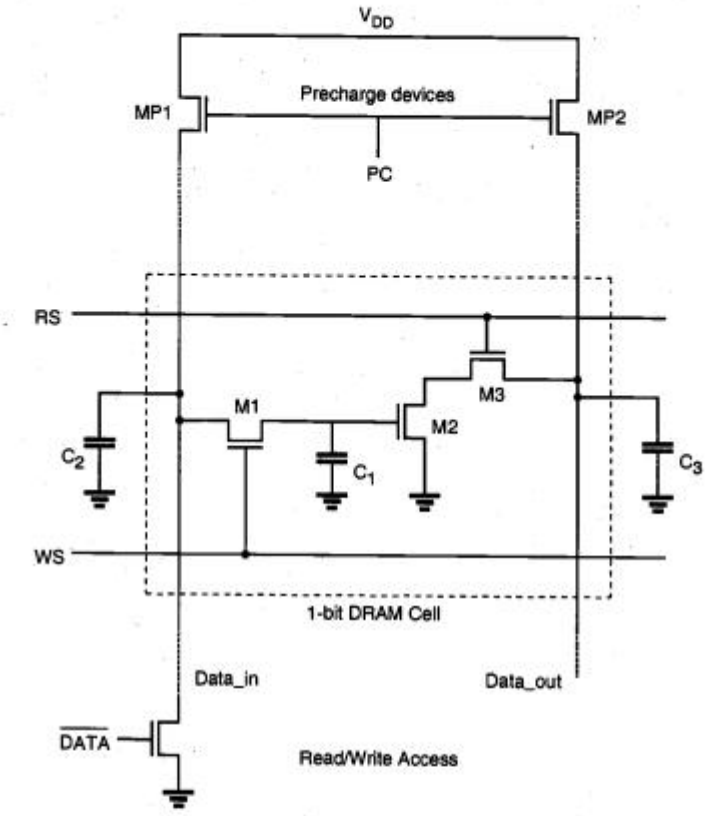


Figure 10.37. Three-transistor DRAM cell with the pull-up and read/write circuitry.

All "data read" and "data write" operations are performed during the active ϕ_2 phase, i.e., when PC is low. Figure 10.38 depicts the typical voltage waveforms associated with the 3-T DRAM cell during a sequence of four consecutive operations: write "1," read "1," write "0," and read "0." The four precharge cycles shown in Fig. 10.38 are numbered 1, 3, 5, and 7, respectively. Figure 10.39 illustrates the transient currents charging up the two columns (D_{in} and D_{out}) during a precharge cycle. The precharge cycle is effectively completed when both capacitance voltages reach their steady-state values. Note here that the two column capacitances C_2 and C_3 are at least one order of magnitude larger than the internal storage capacitance C_1.

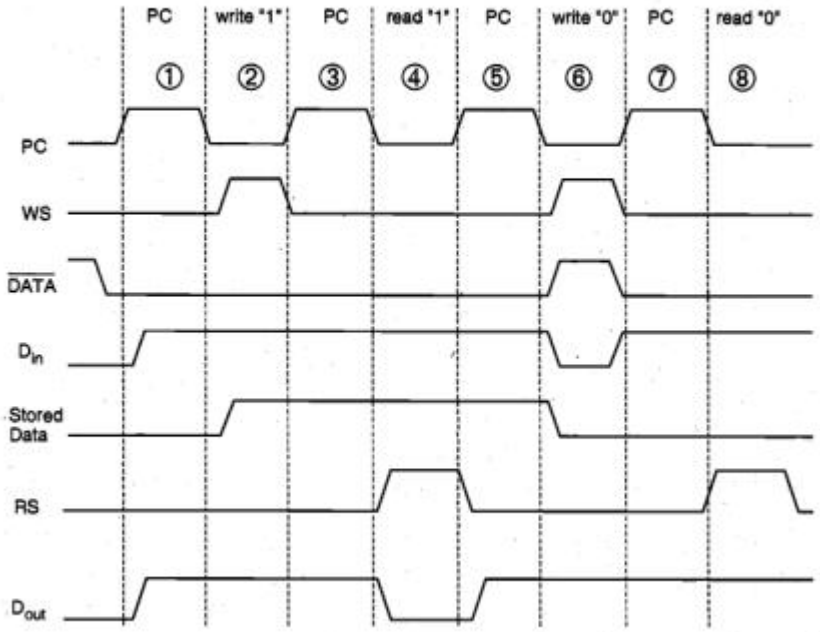


Figure 10.38. Typical voltage waveforms associated with the 3-T DRAM cell during four consecutive operations: write "1," read "1," write "0," and read "0."

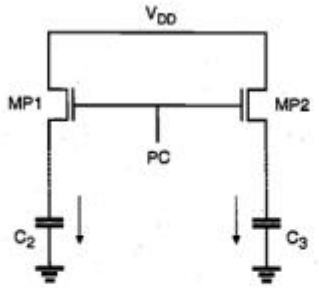


Figure 10.39. Column capacitances C_2 and C_3 are being charged-up through MP1 and MP2 during the precharge cycle.

For the write "1" operation, the *inverse* data input is at the logic-low level, because the data to be written onto the DRAM cell is logic "1." Consequently, the "data write" transistor MD is turned off, and the voltage level on column D_{in} remains high. Now, the "write select" signal WS is pulled high during the active phase of ϕ_2. As a result, the write access transistor M1 is turned on. With M1 conducting, the charge on C_2 is now shared

with C_1 (Fig. 10.40). Since the capacitance C_2 is very large compared to C_1, the storage node capacitance C_1 attains approximately the same logic-high level as the column capacitance C_2 at the end of the charge-sharing process.

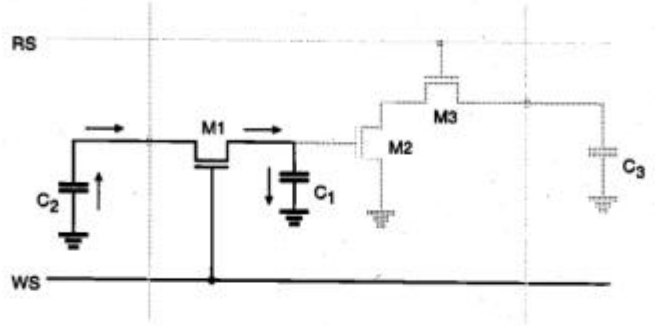


Figure 10.40. Charge sharing between C_2 and C_1 during the write "1" sequence.

After the write "1" operation is completed, the write access transistor M1 is turned off. With the storage capacitance C_1 charged-up to a logic-high level, transistor M2 is now conducting. In order to read this stored "1," the "read select" signal RS must be pulled high during the active phase of ϕ_2, following a precharge cycle. As the read access transistor M3 turns on, M2 and M3 create a conducting path between the "data read" column capacitance C_3 and the ground. The capacitance C_3 discharges through M2 and M3, and the falling column voltage is interpreted by the "data read" circuitry as a stored logic "1." The active portion of the DRAM cell during the read "1" cycle is shown in Fig. 10.41. Note that the 3-T DRAM cell may be read repeatedly in this fashion without disturbing the charge stored in C_1.

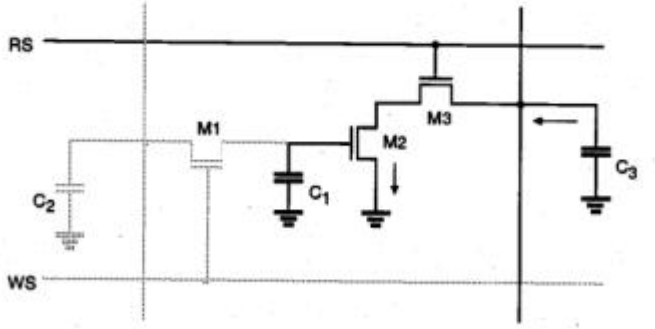


Figure 10.41. The column capacitance C_3 is discharged through the transistors M2 and M3 during the read "1" operation.

For the write "0" operation, the inverse data input is at the logic-high level, because the data to be written onto the DRAM cell is a logic "0." Consequently, the data write

transistor is turned on, and the voltage level on column D_{in} is pulled to logic "0." Now, the "write select" signal WS is pulled high during the active phase of ϕ_2. As a result, the write access transistor M1 is turned on. The voltage level on C_2, as well as that on the storage node C_1, is pulled to logic "0" through M1 and the data write transistor, as shown in Fig. 10.42. Thus, at the end of the write "0" sequence, the storage capacitance C_1 contains a very low charge, and the transistor M2 is turned off since its gate voltage is approximately equal to zero.

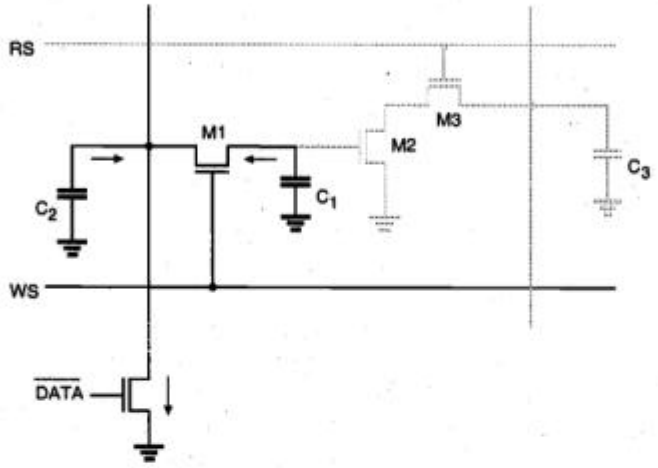


Figure 10.42. Both C_1 and C_2 are discharged via M1 and the data write transistor during the write "0" sequence.

In order to read this stored "0," the "read select" signal RS must be pulled high during the active phase of ϕ_2, following a precharge cycle. The read access transistor M3 turns on, but since M2 is off, there is no conducting path between the column capacitance C_3 and the ground (Fig. 10.43). Consequently, C_3 does not discharge, and the logic-high level on the D_{out} column is interpreted by the data read circuitry as a stored "0" bit.

As we already pointed out in the beginning of this section, the charge stored in C_1 cannot be held indefinitely, even though the "data read" operations do not significantly disturb the stored charge. The drain junction leakage current of the write access transistor M1 is the main reason for the gradual depletion of the stored charge on C_1. In order to *refresh* the data stored in the DRAM cells before they are altered due to leakage, the data must be periodically read, inverted (since the data output level reflects the inverse of the stored data), and then written back into the same cell location. This refresh operation is performed for all storage cells in the DRAM array every 2 to 4 ms. Note that all bits in one row can be refreshed at once, which significantly simplifies the procedure.

It can be seen that the three-transistor dynamic RAM cell examined here does not dissipate any static power for data storage, since there is no continuous current flow in the circuit. Also, the use of periodic precharge cycles instead of static pull-up further reduces the dynamic power dissipation. The additional peripheral circuitry required for

scheduling the non-overlapping control signals and the refresh cycles does not significantly overshadow these advantages of the low-power dynamic memory.

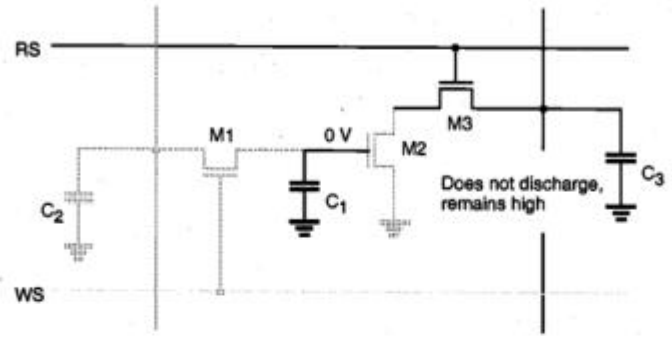


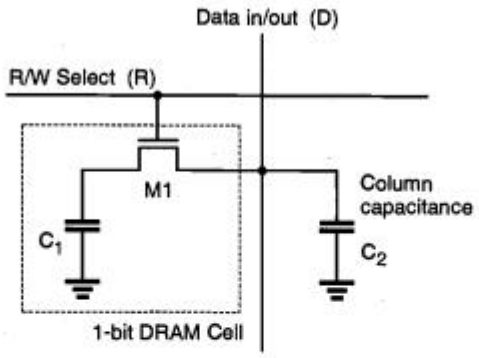
Figure 10.43. The column capacitance C_3 cannot discharge during the read "0" cycle.

One-Transistor DRAM Cell

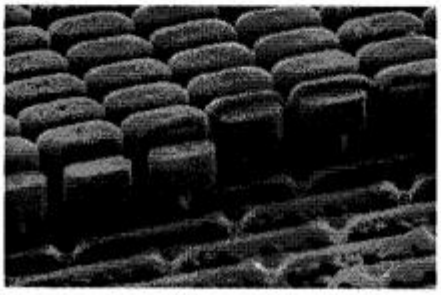
The circuit diagram of the one-transistor (1-T) DRAM cell consisting of one explicit storage capacitor and one access transistor is shown in Fig. 10.44. Here, C_1 represents the storage capacitor which typically has a value of 30 to 100 fF. Similar to the 3-T DRAM cell, binary data are stored as the presence or absence of charge in the storage capacitor. Capacitor C_2 represents the much larger parasitic column capacitance associated with the word line. Charge sharing between this large capacitance and the very small storage capacitance plays a very important role in the operation of the 1-T DRAM cell.

The "data write" operation on the 1-T cell is quite straightforward. For the write "1" operation, the bit line (D) is raised to logic "1" by the write circuitry, while the selected word line is pulled high by the row address decoder. The access transistor M1 turns on, allowing the storage capacitor C_1 to charge up to a logic-high level. For the write "0" operation, the bit line (D) is pulled to logic "0" and the word line is pulled high by the row address decoder. In this case, the storage capacitor C_1 discharges through the access transistor, resulting in a stored "0" bit.

In order to read stored data out of a 1-T DRAM cell, on the other hand, we have to build a fairly elaborate read-refresh circuit. The reason for this is the fact that the "data read" operation on the one-transistor DRAM cell is by necessity a "destructive readout." This means that the stored data must be destroyed or lost during the read operation. Typically, the read operation starts with precharging the column capacitance C_2. Then, the word line is pulled high in order to activate the access transistor M1. Charge sharing between C_1 and C_2 occurs and, depending on the amount of stored charge on C_1, the column voltage either increases or decreases slightly. Note that charge sharing inevitably destroys the stored charge on C_1. Hence, we also have to *refresh* data every time we perform a "data read" operation.



(a)



(b)

Figure 10.44. (a) Typical one-transistor (1-T) DRAM cell with its access lines. (b) SEM photograph of a stacked-capacitor DRAM cell structure.

An example of the 256-cells-per-column DRAM read circuitry is shown in Fig. 10.45, along with typical control signal waveforms. A cross-coupled dynamic latch circuit is used to detect the small voltage differences and to restore the signal levels. The storage array is split in half so that equal capacitances are connected to each side of the cross-coupled latch. This means that half of the cells connected to one bit line (column) are arranged on one side of the latch, and the other half of the cells connected to the same column are arranged on the other side. As shown in Fig. 10.45, each half-column in the array also has a dummy cell which contains a capacitance half of the storage capacitance value. The capacitors C_D and $C_{\overline{D}}$ in Fig. 10.45 represent the relatively large parasitic column capacitances associated with the half-columns.

The "read-refresh" operation occurs in three stages. First, the precharge devices are turned on during the active phase of PC. Both column capacitances C_D and $C_{\overline{D}}$ are charged-up to the same logic-high level, whereas the dummy nodes X and Y are pulled to logic-low level. The devices involved in the precharge operation are highlighted in Fig. 10.46. Note that during this phase, all other signals are inactive.

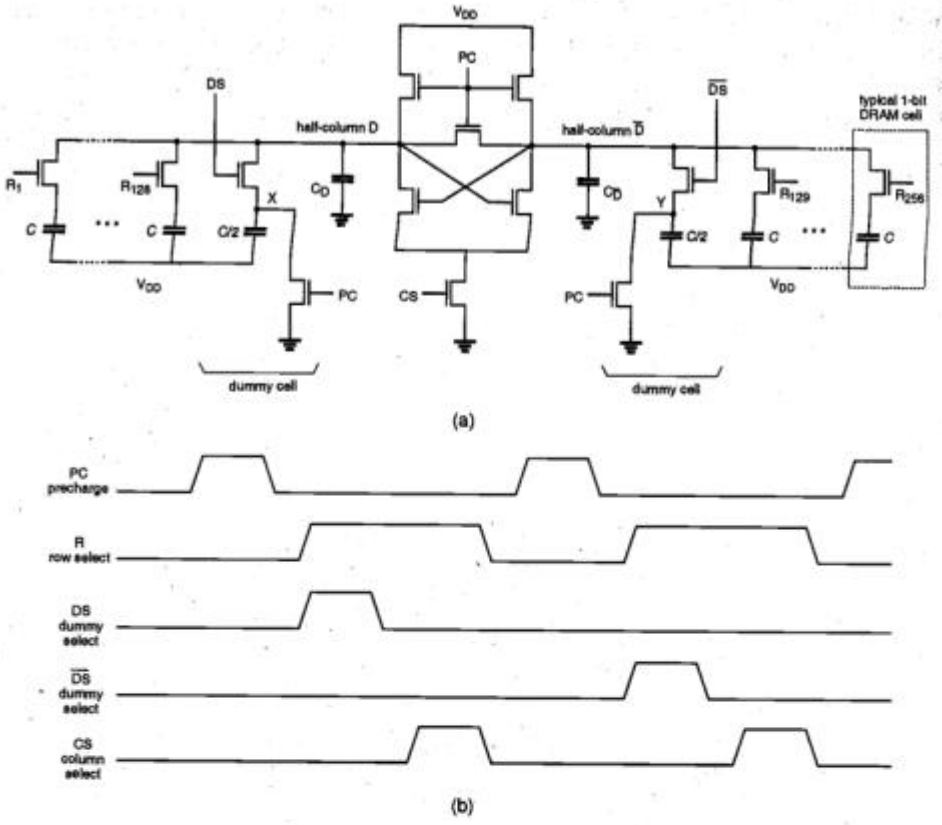


Figure 10.45. (a) Data read-restore circuit example for 256 1-T DRAM cells per column. (b) Typical control signal waveforms for two consecutive data read operations, performed on alternate sides (half-columns) of the array.

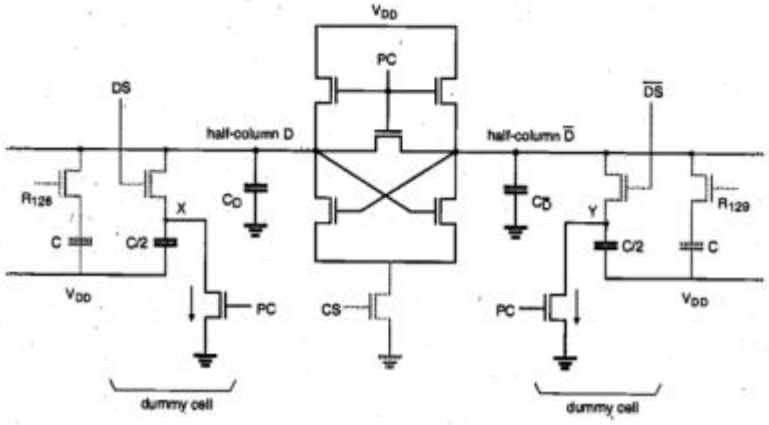


Figure 10.46. The half-columns are being charged-up during the precharge phase.

Next, one of the 256 word lines is raised to logic "1" during the row selection phase. At the same time, the dummy cell on the other side is also selected by raising either *DS* or $\overline{DS}$. This situation is depicted in Fig. 10.47, where only the selected

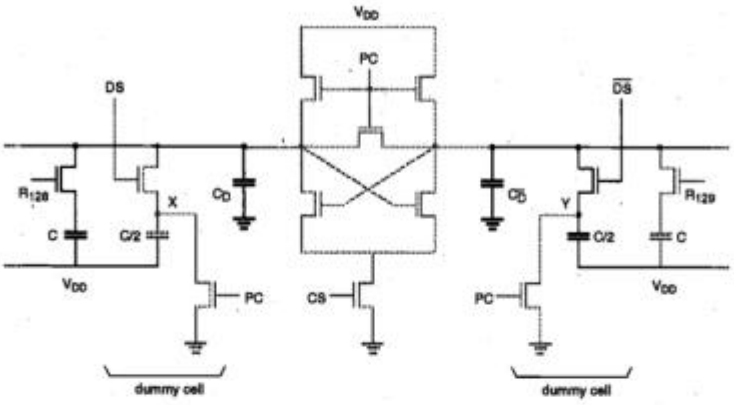


Figure 10.47. Two complementary column voltages are determined through charge sharing, on the one side between the selected storage cell and the half-column capacitance, on the other side between the dummy cell and the other half-column capacitance.

DRAM cell (left) and the corresponding dummy cell (right) are highlighted. If a logic "1" is stored in the selected cell, the voltage on the half-column *D* will rise slightly, while the voltage on half-column $\overline{D}$ drops, because the dummy cell is being charged up. If a logic "0" is stored in the selected cell, however, the voltage on the half-column *D* will also drop, and the drop in V_D will be larger than the drop in $V_{\overline{D}}$. Consequently, there will be a detectable difference between a stored "0" and a stored "1."

The final stage of the "read-refresh" operation is performed during the active phase of CS, the column-select signal. As soon as the cross-coupled latch is activated, the slight voltage difference between the two half-columns is amplified, and the latch forces the two half-columns into opposite states (Fig. 10.48). Thus, the stored data on the selected DRAM cell is refreshed while it is being read by the "read-refresh" circuitry.

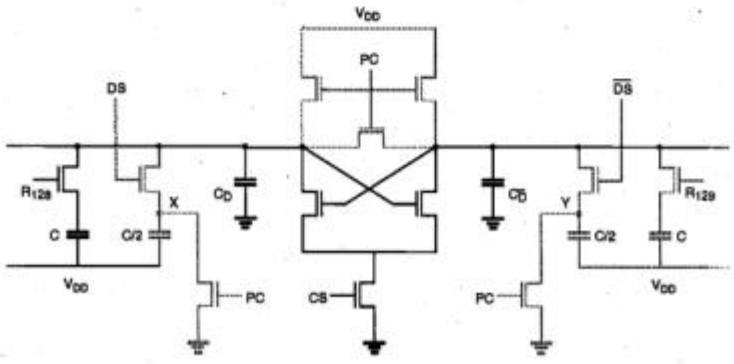


Figure 10.48. The cross-coupled latch circuit is used for detection of the voltage difference between the half-columns and for restoring the voltage level on the accessed cell.

10.3. Static Read-Write Memory (SRAM) Circuits

As already explained in Section 10.1, read-write (R/W) memory circuits are designed to permit the modification (writing) of data bits to be stored in the memory array, as well as their retrieval (reading) on demand. The memory circuit is said to be *static* if the stored data can be retained indefinitely (as long as a sufficient power supply voltage is provided), without any need for a periodic refresh operation. We will examine the circuit structure and the operation of simple SRAM cells, as well as the peripheral circuits designed to read and write the data.

The data storage cell, i.e., the 1-bit memory cell in static RAM arrays, invariably consists of a simple latch circuit with two stable operating points (states). Depending on the preserved state of the two-inverter latch circuit, the data being held in the memory cell will be interpreted either as a logic "0" or as a logic "1." To access (read and write) the data contained in the memory cell via the bit line, we need at least one switch, which is controlled by the corresponding word line, i.e., the row address selection signal (Fig. 10.21(a)). Usually, two complementary access switches consisting of nMOS pass transistors are implemented to connect the 1-bit SRAM cell to the complementary bit lines (columns). This can be likened to turning the car steering wheel with both left and right hands in complementary directions.

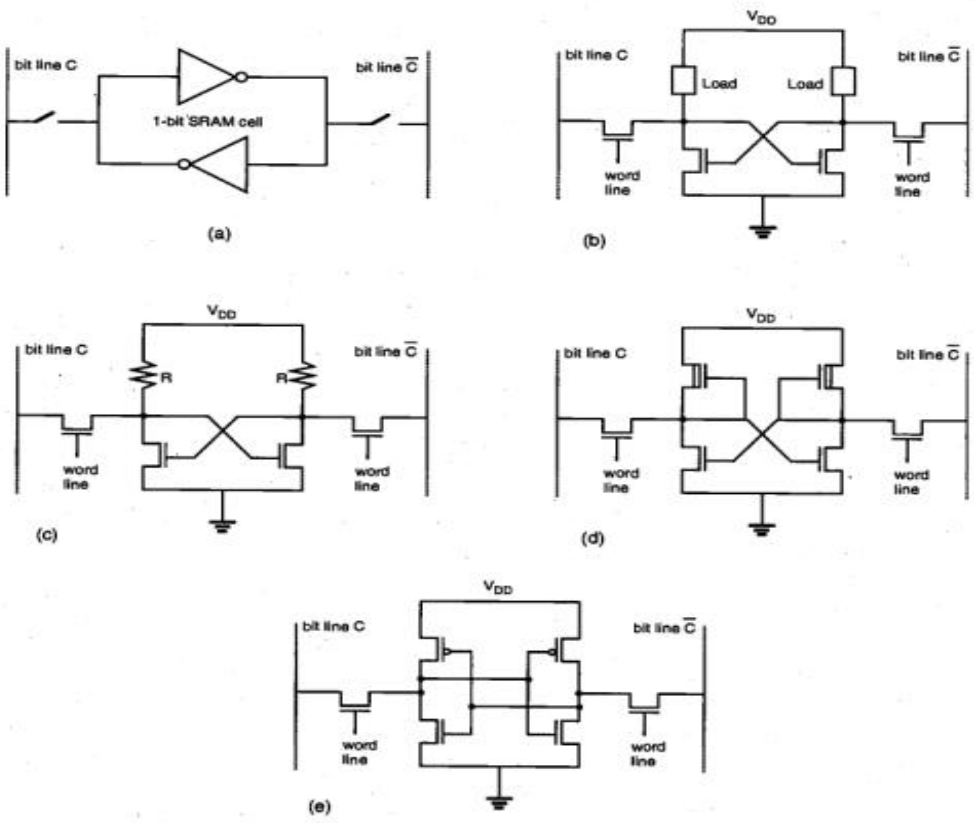


Figure 10.21. Various configurations of the static RAM cell. (a) Symbolic representation of the two-inverter latch circuit with access switches. (b) Generic circuit topology of the MOS static RAM cell. (c) Resistive-load SRAM cell. (d) Depletion-load nMOS SRAM cell. (e) Full CMOS SRAM cell.

Figure 10.21(b) shows the generic structure of the MOS static RAM cell, consisting of two cross-coupled inverters and two access transistors. The load devices may be polysilicon resistors, depletion-type nMOS transistors, or pMOS transistors, depending on the type of the memory cell. The pass gates acting as data access switches are enhancement-type nMOS transistors.

The use of resistive-load inverters with undoped polysilicon resistors in the latch structure typically results in a significantly more compact cell size, compared with the other alternatives (Fig. 10.21(c)). This is true since the resistors can be stacked on top of the cell (using double-polysilicon technology), thereby reducing the cell size to four transistors, as opposed to the six-transistor cell topologies. If multiple polysilicon layers are available, one layer can be used for the gates of the enhancement-type nMOS transistors, while another level is used for load resistors and interconnects.

In order to attain acceptable noise margins and output pull-up times for the resistive-load inverter, the value of the load resistor has to be kept relatively low, as already examined in Section 5.2. On the other hand, a high-valued load resistor is required in order to reduce the amount of standby current being drawn by each memory cell. Thus, there is a trade-off between the high resistance required for low power and the requirement to provide wider noise margins and high speed. The power consumption issue will be addressed later in more detail.

The six-transistor depletion-load nMOS SRAM cell shown in Fig. 10.21(d) can be easily implemented with one polysilicon and one metal layer, and the cell size tends to be relatively small, especially with the use of buried metal-diffusion contacts. The static characteristics and the noise margins of this memory cell are typically better than those of the resistive-load cell. The static power consumption of the depletion-load SRAM cell, however, makes it an unsuitable candidate for high-density SRAM arrays.

The full CMOS SRAM cell shown in Fig. 10.21(e) achieves the lowest static power dissipation among the various circuit configurations presented here. In addition, the CMOS cell offers superior noise margins and switching speed as well. The comparative advantages and disadvantages of the CMOS static RAM cell will be investigated in depth later in this section.

SRAM Operation Principles

Figure 10.22 shows a typical four-transistor resistive-load SRAM cell widely used in high-density memory arrays, consisting of a pair of cross-coupled inverters. The two stable operating points of this basic latch circuit are used to store a one-bit piece of information; hence, this pair of cross-coupled inverters make up the central component of the SRAM cell. To perform read and write operations, we use two nMOS pass transistors, both of which are driven by the row select signal, *RS*. Note that the SRAM cell shown in Fig. 10.22 is accessed via two bit lines or columns, instead of one. This complementary column arrangement allows for a more reliable operation.

When the word line (*RS*) is not selected, i.e., when the voltage level of line *RS* is equal to logic "0," the pass transistors M3 and M4 are turned off. The simple latch circuit consisting of two cross-connected inverters preserves one of its two stable operating points; hence, data is being *held*. At this point, consider the two columns, C and $\overline{C}$. If all word lines in the SRAM array are inactive, the relatively large column capacitances are

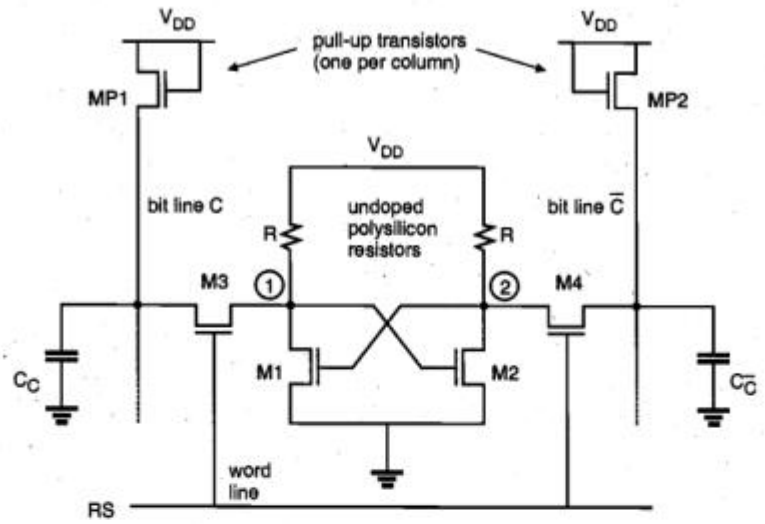


Figure 10.22. Basic structure of the resistive-load SRAM cell, shown with the column pull-up transistors.

charged-up by the column pull-up transistors, MP1 and MP2. Since both transistors operate in saturation, the steady-state voltage level V_C for both columns is determined by the following relationship:

$$V_{DD} - V_C = V_{T0} + \gamma\left(\sqrt{|2\phi_F| + V_C} - \sqrt{|2\phi_F|}\right) \tag{10.1}$$

Assuming $V_{DD} = 5$ V, $V_{T0} = 1$ V, $|2\phi_F| = 0.6$ V, and $\gamma = 0.4$ $V^{1/2}$, this voltage level is found to be approximately equal to 3.5 V. Note that the voltage levels of the two complementary bit lines (columns) are equal during this phase.

Now assume that we select the memory cell by raising its word line voltage to logic "1," hence, the pass transistors M3 and M4 are turned on. Once the memory cell is selected, four basic operations may be performed on this cell.

a) Write "1" operation: The voltage level of column $\overline{C}$ is forced to logic-low by the data-write circuitry. The driver transistor M1 turns off. The voltage V_1 attains a logic-high level, while V_2 goes low.

b) Read "1" operation: The voltage of column C retains its precharge level while the voltage of column $\overline{C}$ is pulled down by M2 and M4. The data-read circuitry detects the small voltage difference ($V_C > V_{\overline{C}}$) and amplifies it as a logic "1" data output.

c) Write "0" operation: The voltage level of column C is forced to logic-low by the data-write circuitry. The driver transistor M2 turns off. The voltage V_2 attains a logic-high level, while V_1 goes low.

d) Read "0" operation: The voltage of column $\overline{C}$ retains its precharge level while the voltage of column C is pulled down by M1 and M3. The data-read circuitry detects the small voltage difference ($V_C < V_{\overline{C}}$) and amplifies it as a logic "0" data output.

Typical voltage waveforms associated with the word line *RS* and the two pseudo-complementary bit lines are shown qualitatively in Fig. 10.23. Note that the voltage difference between the two columns during a read operation may be only a few hundred millivolts, which must be detected by the data-read circuitry. The reason for this is that the two nMOS transistors in series (e.g., M1 and M3 for read "0") pulling down the column during the read phase cannot discharge the large column capacitance quickly.

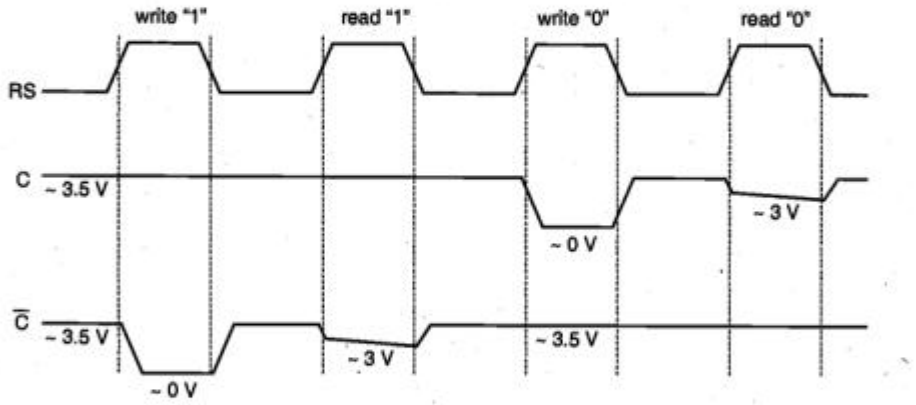


Figure 10.23. Typical row and column voltage waveforms of the resistive-load SRAM shown in Fig. 10.22 during read and write.

Power Consumption

To estimate the standby power consumption of the static read-write memory cell, assume that a "1"-bit is stored in the cell. This means that the driver transistor M1 is turned off, while the driver transistor M2 is conducting, resulting in $V_1 = V_{OH}$ and $V_2 = V_{OL}$. In this circuit, one of the load resistors will always conduct a non-zero current and, consequently, consume steady-state power. The amount of the standby power consumption is ultimately determined by the value of the load resistor.

Large resistance values and a smaller cell size can be achieved by using lightly-doped or undoped polysilicon for the load resistors, which has a typical sheet resistivity of 10 MΩ per square or higher. The added process complexity for implementing resistive-

load SRAM cells using undoped poly resistors is usually worth the advantage of low-power operation, which is manifested by a standby current of less than 10 nA per cell. With a load resistance value of $R = 100\ M\Omega$, for example, the standby power dissipation of the resistive SRAM cell shown in Fig. 10.22 becomes

$$P_{stand-by} = V_{DD} \cdot \frac{\mu_n C_{ox}}{2} \cdot \left(\frac{W}{L}\right)_{load} \cdot \left|V_{T,load}\right|^2 \tag{10.2}$$

If we consider that a typical memory array contains a large number of memory cells, each consuming a non-zero standby power, the significance of the power consumption problem in static memory arrays becomes obvious.

Full CMOS SRAM Cell

A low-power SRAM cell may be designed simply by using cross-coupled CMOS inverters instead of the resistive-load nMOS inverters. In this case, the stand-by power consumption of the memory cell will be limited to the relatively small leakage currents of both CMOS inverters. The possible drawback of using CMOS SRAM cells, on the other hand, is that the cell area tends to increase in order to accommodate the n-well for the pMOS transistors and the polysilicon contacts.

The circuit structure of the full CMOS static RAM cell is shown in Fig. 10.24, along with the pMOS column pull-up transistors on the complementary bit lines. The basic operation principle of the CMOS SRAM cell is identical to that of the resistive-load nMOS cell examined earlier. The most important advantage of this circuit topology is that the static power dissipation is even smaller; essentially, it is limited by the leakage current of the pMOS transistors. A CMOS memory cell thus draws current from the power supply only during a switching transition. The low standby power consumption has certainly been a driving force for the increasing prominence of high- density CMOS SRAMs.

Other advantages of CMOS SRAM cells include high noise immunity due to larger noise margins, and the ability to operate at lower power supply voltages than, for example, the resistive-load SRAM cells. The major disadvantages of CMOS memories historically were larger cell size, the added complexity of the CMOS process, and the tendency to exhibit "latch-up" phenomena. With the widespread use of multi-layer polysilicon and multi-layer metal processes, however, the area disadvantage of the CMOS SRAM cell has been reduced significantly in recent years. Considering the undisputable advantages of CMOS for low-power and low-voltage operation, the added process complexity and the required latch-up prevention measures do not present a substantial barrier against the implementation of CMOS cells in high density SRAM arrays. Figure 10.25 compares typical layouts of the four-transistor resistive-load SRAM cell and the six-transistor full CMOS SRAM cell.

Note that unlike the nMOS column pull-up devices used in resistive-load SRAM, the pMOS column pull-up transistors shown in Fig. 10.24 allow the column voltages to reach full V_{DD} level. To further reduce power consumption, these transistors can also be driven by a periodic precharge signal, which activates the pull-up devices to charge-up column capacitances.

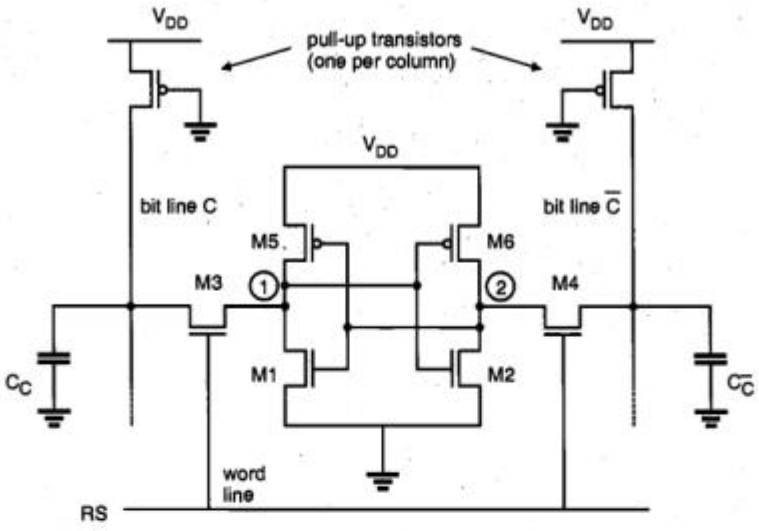
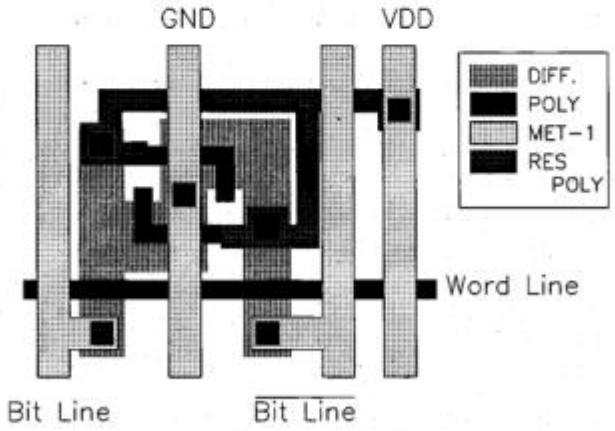


Figure 10.24. Circuit topology of the CMOS SRAM cell.

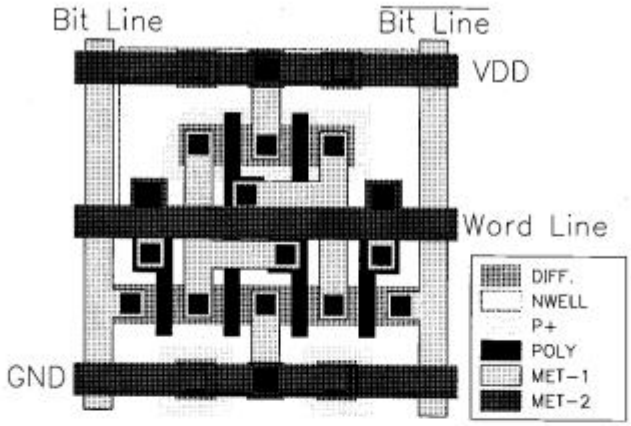
CMOS SRAM Cell Design Strategy

To determine the (*W/L*) ratios of the transistors in a typical CMOS SRAM cell as shown in Fig. 10.24, a number of design criteria must be taken into consideration. The two basic requirements which dictate the (*W/L*) ratios are: (a) the data-read operation should

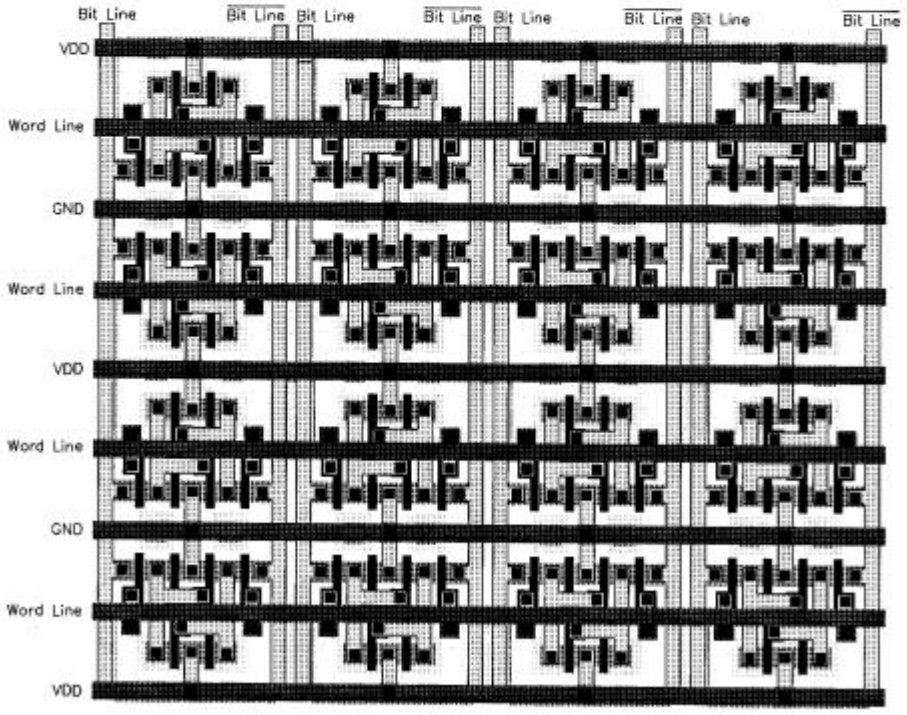


(a)

Figure 10.25. (a) Layout of the resistive-load SRAM cell.



(b)



(c)

Figure 10.25. (continued) (b) Layout of the CMOS SRAM cell. (c) Layout of a 4-bit x 4-bit SRAM array, consisting of 16 CMOS SRAM cells.

not destroy the stored information in the SRAM cell, and (b) the cell should allow modification of the stored information during the data-write phase. Consider the data-read operation first, assuming that a logic "0" is stored in the cell. The voltage levels in the CMOS SRAM cell at the beginning of the "read" operation are depicted in Fig. 10.26. Here, the transistors M2 and M5 are turned off, while the transistors M1and M6 operate

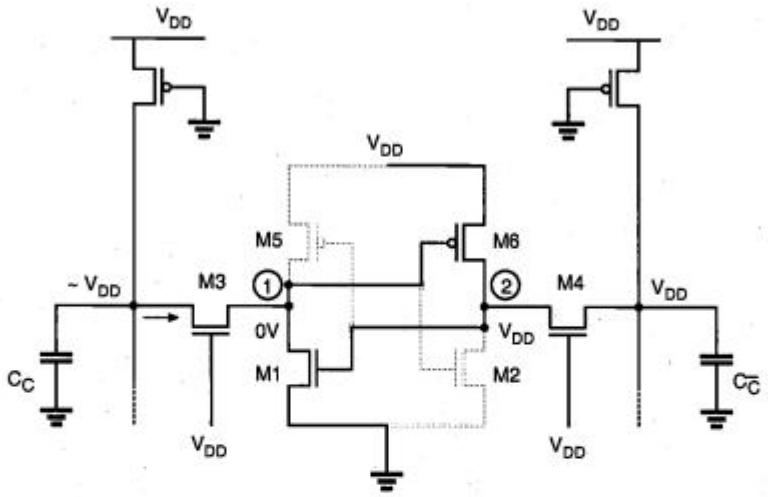


Figure 10.26. Voltage levels in the SRAM cell at the beginning of the "read" operation.

in the linear mode. Thus, the internal node voltages are $V_1 = 0$ and $V_2 = V_{DD}$ *before* the cell access (or pass) transistors M3 and M4 are turned on. The active transistors at the beginning of the data-read operation are highlighted in Fig. 10.26.

After the pass transistors M3 and M4 are turned on by the row selection circuitry, the voltage level of column $\overline{C}$ will not show any significant variation since no current will flow through M4. On the other half of the cell, however, M3 and M1 will conduct a nonzero current and the voltage level of column C will begin to drop slightly. Note that the column capacitance C_C is typically very large; therefore, the amount of decrease in the column voltage is limited to a few hundred millivolts during the read phase. The data-read circuitry to be examined later in this chapter is responsible for detecting this small voltage drop and amplifying it as a stored "0." While M1 and M3 are slowly discharging the column capacitance, the node voltage V_1 will increase from its initial value of 0 V. Especially if the (W/L) ratio of the access transistor M3 is large compared to the (W/L) ratio of M1, the node voltage V_1 may exceed the threshold voltage of M2 during this process, forcing an unintended change of the stored state. The key design issue for the data-read operation is then to guarantee that the voltage V_1 does not exceed the threshold voltage of M2, so that the transistor M2 remains turned off during the read phase, i.e.,

$$V_{1,max} \leq V_{T,2} \tag{10.3}$$

We can assume that after the access transistors are turned on, the column voltage V_C remains approximately equal to V_{DD}. Hence, M3 operates in saturation while M1 operates in the linear region.

$$\frac{k_{n,3}}{2}\left(V_{DD}-V_1-V_{T,n}\right)^2 = \frac{k_{n,1}}{2}\left(2\left(V_{DD}-V_{T,n}\right)V_1-V_1^2\right) \tag{10.4}$$

Combining this equation with (10.3) results in:

$$\frac{k_{n,3}}{k_{n,1}} = \frac{\left(\frac{W}{L}\right)_3}{\left(\frac{W}{L}\right)_1} < \frac{2\left(V_{DD}-1.5\,V_{T,n}\right)V_{T,n}}{\left(V_{DD}-2\,V_{T,n}\right)^2} \tag{10.5}$$

The upper limit of the aspect ratio found above is actually more conservative, since a portion of the drain current of M3 will also be used to charge-up the parasitic node capacitance of node (1). To summarize, the transistor M2 will remain in cut-off during the read "0" operation if condition (10.5) is satisfied. A symmetrical condition also dictates the aspect ratios of M2 and M4.

Now consider the write "0" operation, assuming that a logic "1" is stored in the SRAM cell initially. Figure 10.27 shows the voltage levels in the CMOS SRAM cell at the beginning of the data-write operation. The transistors M1 and M6 are turned off, while the transistors M2 and M5 operate in the linear mode. Thus, the internal node voltages are $V_1 = V_{DD}$ and $V_2 = 0$ V *before* the cell access (or pass) transistors M3 and M4 are turned on.

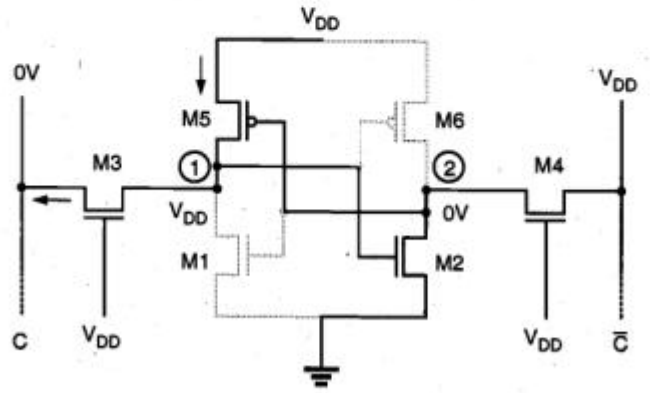


Figure 10.27. Voltage levels in the SRAM cell at the beginning of the "write" operation.

The column voltage V_C is forced to logic "0" level by the data-write circuitry; thus, we may assume that V_C is approximately equal to 0 V. Once the pass transistors M3 and M4 are turned on by the row selection circuitry, we expect that the node voltage V_2 remains *below* the threshold voltage of M1, since M2 and M4 are designed according to condition (10.5). Consequently, the voltage level at node (2) would not be sufficient to turn on M1. To change the stored information, i.e., to force V_1 to 0 V and V_2 to V_{DD}, the node voltage V_1 *must be reduced below* the threshold voltage of M2, so that M2 turns off first. When $V_1 = V_{T,n}$, the transistor M3 operates in the linear region while M5 operates in saturation.

$$\frac{k_{p,5}}{2}\left(0 - V_{DD} - V_{T,p}\right)^2 = \frac{k_{n,3}}{2}\left(2\left(V_{DD} - V_{T,n}\right)V_{T,n} - {V_{T,n}}^2\right) \tag{10.6}$$

Rearranging this condition results in:

$$\frac{k_{p,5}}{k_{n,3}} < \frac{2\left(V_{DD} - 1.5\,V_{T,n}\right)V_{T,n}}{\left(V_{DD} + V_{T,p}\right)^2}$$

$$\frac{\left(\frac{W}{L}\right)_5}{\left(\frac{W}{L}\right)_3} < \frac{\mu_n}{\mu_p} \cdot \frac{2\left(V_{DD} - 1.5\,V_{T,n}\right)V_{T,n}}{\left(V_{DD} + V_{T,p}\right)^2} \tag{10.7}$$

To summarize, the transistor M2 will be *forced* into cut-off mode during the write "0" operation if condition (10.7) is satisfied. This will guarantee that M1 subsequently turns on, changing the stored information. Note that a symmetrical condition also dictates the aspect ratios of M6 and M4.

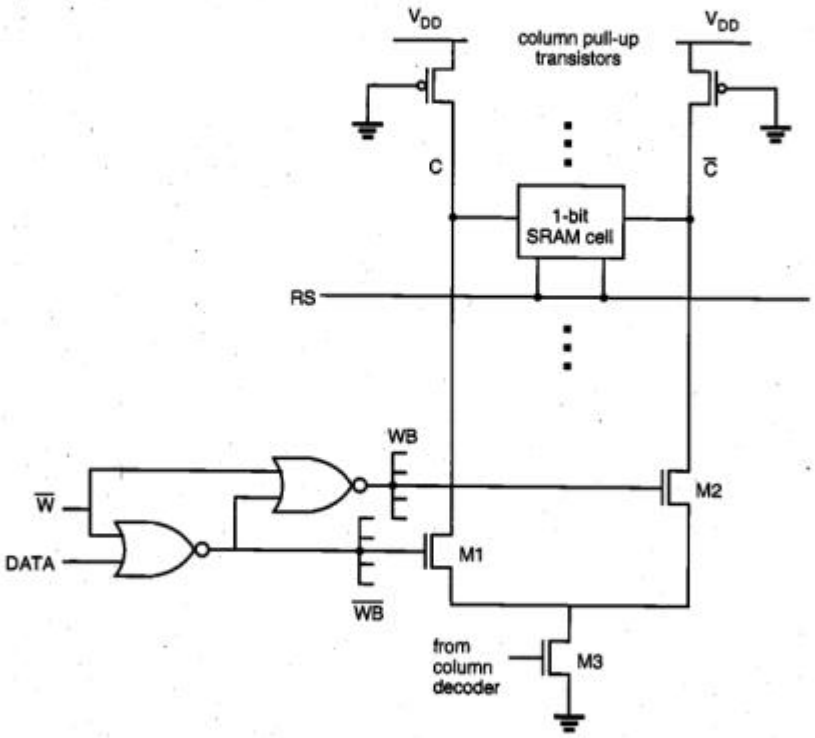
SRAM Write Circuitry

As already discussed in the preceding section, a "write" operation is performed by forcing the voltage level of either column (bit line) to a logic-low level. To accomplish this task, a low-resistance, conducting path must be provided from each column to the ground, which can be selectively activated by the data-write signals. A simplified view of the SRAM "write" circuitry designed for this operation is shown in Fig. 10.28. Here, the nMOS transistors M1 and M2 are used to pull down the two column voltages, while the transistor M3 completes the conducting path to ground. Note that M3 is driven by the column address decoder circuitry, i.e., M3 turns on only when the corresponding column address is selected. The column pull-down transistors, on the other hand, are driven by two pseudo-complementary control signals, *WB* and $\overline{WB}$. The "write-enable" signal *W* (active low) and the data to be written (*DATA*) are used to generate the control signals, as shown in the table in Fig. 10.28.

The nMOS pull-down transistors M1 and M2, as well as the column selection transistor M3 must have sufficiently large (W/L) ratios so that the column voltages can be forced to almost 0 V level during a "write" operation. Also note that the data input circuitry consisting of two NOR2 gates can be shared by several columns, assuming that one column is activated, i.e., selected by the column address decoder, at any given time.

SRAM Read Circuitry

During the "data read" operation in the SRAM array, the voltage level on either one of the columns drops slightly after the pass transistors are turned on by the row address decoder circuit. In order to reduce the read access time, the "read" circuitry must detect



$\overline{W}$	*DATA*	*WB*	$\overline{WB}$	*Operation*
0	1	1	0	M1 is off, M2 is on $\rightarrow$ $V_{\overline{C}}$ low
0	0	0	1	M1 is on, M2 is off $\rightarrow$ V_C low
1	*X*	0	0	M1 and M2 are off $\rightarrow$ both columns remain high

Figure 10.28. Data write circuitry associated with one column-pair in an SRAM array.

a very small voltage difference between the two complementary columns, and amplify this difference to produce a valid logic output level. A simple source-coupled differential amplifier can be used for this task, as shown in Fig. 10.31. Here, the drain currents of the two complementary nMOS transistors are:

$$I_{D,1} = \frac{k_n}{2}\left(V_C - V_x - V_{T,1}\right)^2 \tag{10.8}$$

$$I_{D,2} = \frac{k_n}{2}\left(V_{\overline{C}} - V_x - V_{T,2}\right)^2 \tag{10.9}$$

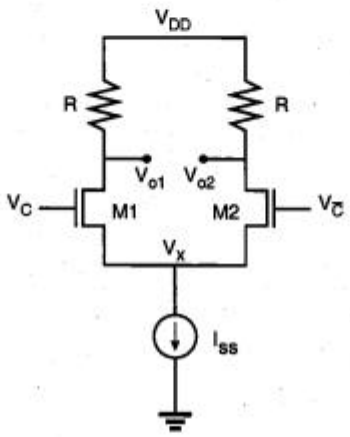


Figure 10.29. Simple source-coupled differential amplifier circuit for "read" operation.

Small-signal analysis of the circuit yields the differential gain of this circuit as

$$\frac{\partial\left(V_{o1} - V_{o2}\right)}{\partial\left(V_C - V_{\overline{C}}\right)} = -R \cdot g_m \tag{10.10}$$

where

$$g_m = \frac{\partial I_D}{\partial V_{GS}} = \sqrt{2\,k_n\,I_D}$$

The differential gain of the amplifier can be increased significantly by using active loads instead of resistors and by using *cascode* configuration, i.e., an intermediary common-gate stage between the common-source transistors and the load transistors. Finally, the differential output of the cascode stage must be converted into single-ended output, by using a level-shifter and buffer stage. Although the "data read" circuitry described above

is capable of detecting small voltage differences between the two complementary bit lines (columns), other types of efficient sense amplifiers are also implemented with CMOS technology, as will be examined in the following.

The architecture of the output "read" circuitry is driven primarily by the constant demand for high access speed and high integration density. We must recognize first that the full CMOS SRAM cell has a natural speed advantage which is derived from using both active pull-up and active pull-down devices in the latch circuits. The CMOS rise and fall times are short and symmetrical, whereas the nMOS latch circuit has a short output fall time but a larger rise time because of its high-resistance load. Both the depletion-load nMOS SRAM cell and the resistive-load nMOS SRAM cell, hence, have slower average switching speed compared to that for the full CMOS cell.

Apart from the type of the SRAM cell, the *precharging* of bit lines also plays a significant role in the access time. In an unclocked SRAM array, data from the accessed cell develops a voltage difference on the bit lines. This voltage difference is then detected and amplified to drive the output buffer. When another cell on the same column is accessed next, one that contains data opposite to the data contained in the previously accessed cell, the output has to switch first to an equalized state and then to the opposite logic state. Since the capacitance on the bit lines is quite large, the time required for switching the differential from one state to the other becomes a significant portion of the overall access time. The access time penalty associated with this procedure can be substantially reduced by the *equalization* of bit lines prior to each new access. Equalization can be done when the memory array is deselected, i.e., between two access cycles.

Fast Sense Amplifiers

The availability of the CMOS technology for the manufacturing of high-density SRAM arrays, either with full-CMOS or resistive-load nMOS memory cells, also allows us to implement efficient sense amplifier structures with pMOS current mirrors. Figure 10.30 shows a single-stage differential current-mirror amplifier, which is typically used as a front-end (input stage) in operational amplifiers.

In this circuit, the gates of the two nMOS transistors M1 and M2 are connected to the bit lines. Their substrate terminals are tied to their respective source terminals in order to remove the substrate-bias effect. Notice that each bit line is represented by a large parasitic capacitance. The nMOS transistor M3 is a long-channel device which acts as a current source for both branches, and is controlled by a clock signal. The output inverter is not a part of the differential amplifier, but it is used to drive the output node.

Before the beginning of a "read" operation, the two bit lines (columns) are pulled up for equalization, as discussed earlier. The CLK signal is low during this phase, so that the nMOS transistor M3 remains off. Since both M1 and M2 conduct, the common source node is pulled up, and the output node of the amplifier also goes high. Therefore, the output of the inverter is at a logic-low level initially.

Once a memory cell is selected for the "read" operation, the voltage on one of the complementary bit lines will start to drop slightly. At the same time as the row selection signal, the CLK signal driving M3 is also turned on. If the stored data on the selected SRAM cell forces the bit line C to decrease slightly, transistor M1 turns off, and the output voltage of the differential amplifier drops immediately. Consequently, the output voltage

of the inverter goes high. Otherwise, if the stored data on the selected memory cell forces the bit line $\overline{C}$ to drop slightly, M2 turns off. Thus, the voltage level at the output node of the differential amplifier remains high in this case, and the inverter also preserves its logic-low output level.

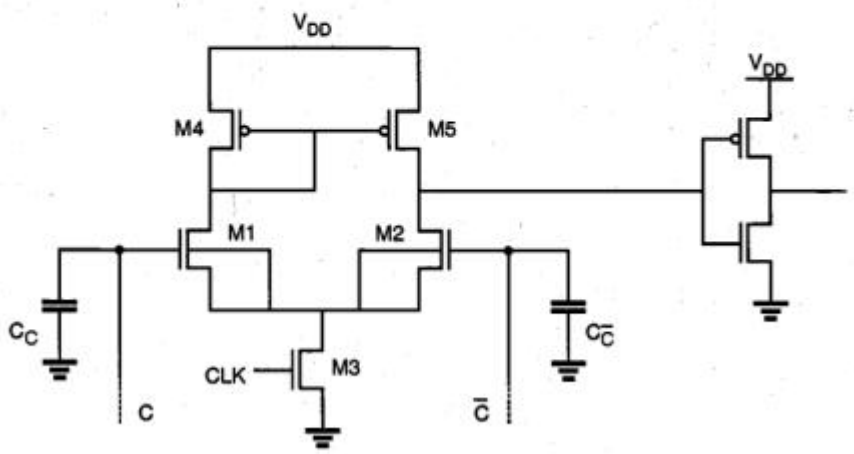


Figure 10.30. Differential current-mirror amplifier to speed up the memory read access time.

In most high-density SRAM chips, two- or three-stage current-mirror differential sense amplifiers are implemented to further improve the "read" access speed. An example of a two-stage CMOS sense amplifier is shown in Fig. 10.31. Here, the first stage consists of two *complementary* differential sense amplifiers, both of which receive the column (bit line) voltages at their inputs. Since they are operated in an anti-symmetrical configuration, the output voltages of the two sense amplifiers must change into opposite directions. The complementary outputs of the first-stage amplifiers are applied to the input terminals of a second-stage differential amplifier, which generates the output signal.

The dynamic behavior of the one-stage sense amplifier is compared with the dynamic behavior of the two-stage sense amplifier via SPICE simulation, in Fig. 10.32. Here, one of the column voltages (V_C) starts to drop slightly, while the other column voltage remains constant (not shown in the figure). It can be seen that the output of the one-stage sense amplifier responds to this change with a delay of about 10 ns, whereas the output delay of the two-stage sense amplifier is only about 1 ns.

As mentioned earlier, in the sense amplifier circuit examined here (Fig. 10.30), the substrate terminals of M1 and M2 are connected to the common source node instead of the ground, in order to avoid threshold voltage variations due to the substrate-bias effect. This is possible using twin-tub, silicon-on-sapphire (SOS), or p-well fabrication processes. In an n-well CMOS process, on the other hand, where all nMOS transistors are formed over the common substrate, the circuit cannot be implemented.

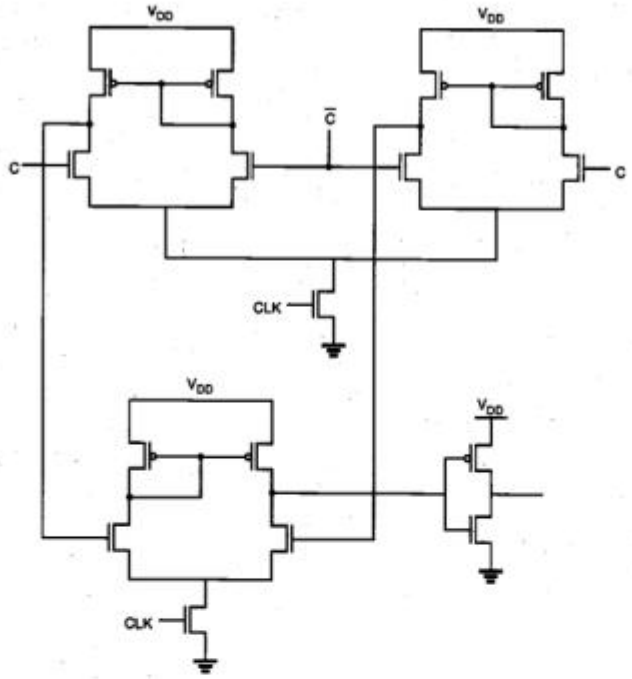


Figure 10.31. Two-stage differential current-mirror sense amplifier circuit.

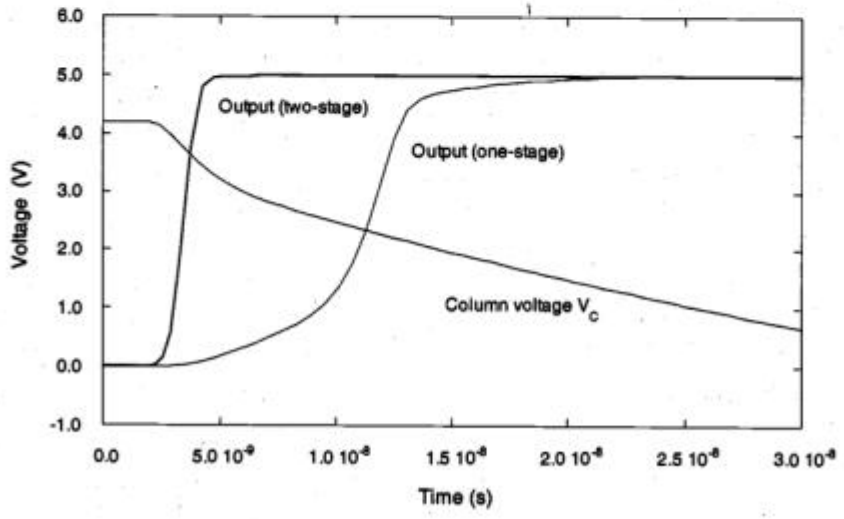


Figure 10.32. Typical dynamic response characteristics of the one-stage sense amplifier and the two-stage sense amplifier circuits.

Another simple circuit topology for sense amplifiers is the cross-coupled latch, shown in Fig. 10.33. Assume that both columns (bit lines) are being pulled up during the precharge cycle and that the voltage on the bit line C starts to drop slightly when the SRAM cell access transistors are activated. Consequently, when the transistor M3 is turned on, the voltage at node $\overline{C}$ is higher than the voltage at node C. Therefore, M1 turns on first and further pulls down the potential at node C. This makes it more difficult for M2 to conduct. Eventually, M2 turns off completely, and M1 keeps conducting, so that the bit line C is discharged through M1 and M3.

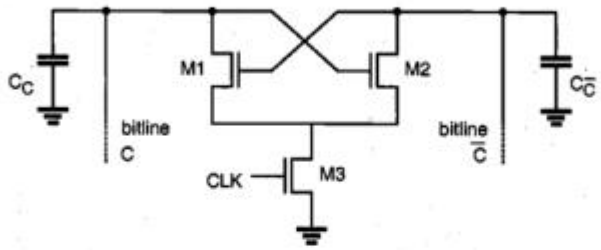


Figure 10.33. Cross-coupled nMOS sense amplifier circuit.

Note that the operation of this circuit is distinctly different from that of the differential amplifier circuit. The cross-coupled sense amplifier does not generate an output voltage level which corresponds to the polarity of the voltage difference between the two bit lines, but it rather *amplifies* the small voltage difference already existing between the bit lines. This voltage difference must still be translated into a logic level, by using a buffer stage. We will see the implementation of a very similar structure also in the dynamic RAM (DRAM) sense amplifiers.

In most SRAM arrays, the cross-coupled sense amplifier circuit is used in conjunction with the differential sense amplifier (Fig. 10.30) examined earlier. In this case, the cross-coupled amplifier serves as a front-end structure to amplify the small voltage difference between the two bit lines, whereas the current-mirror amplifier detects the voltage difference and generates the output level. Figure 10.34 shows the complete circuit diagram of a CMOS SRAM column with one representative CMOS memory cell, the "data write" circuitry, the cross-coupled sense amplifier, and the differential (main) sense amplifier circuit. Note that the main amplifier is connected to the two complementary bit lines via pass transistors, which are driven by the "read select" signal. This configuration enables the use of one main sense amplifier to read the data out of several columns, one at a time.

Dual-Port Static RAM Arrays

In some cases, the memory array may have to be accessed simultaneously by multiple processors, or by one processor and another peripheral device. This could result in a timing conflict called "contention," which can be resolved only by having one of the

processors wait until the SRAM is free. The added wait state, however, significantly reduces the advantages of the high-speed processor. The dual-port RAM architecture is implemented in systems in which a main memory array must serve multiple high-speed processors and peripheral devices with minimum delay.

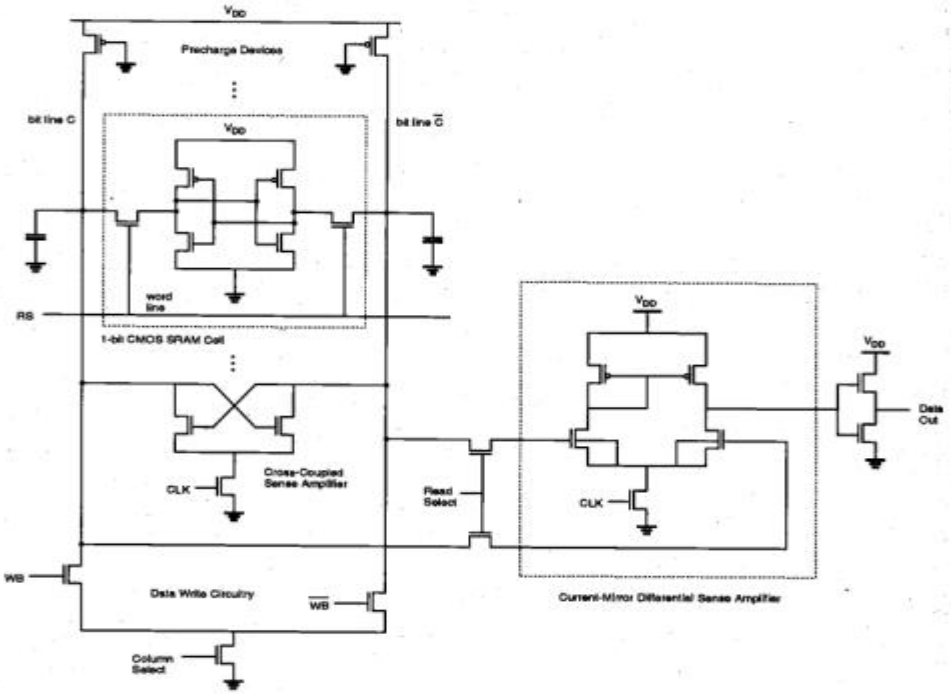


Figure 10.34. Complete circuit diagram of a CMOS static RAM column with data write and data read circuitry.

The ideal dual-port SRAM allows simultaneous access to the same location in the memory array, by using two independent sets of bit lines and associated access switches for each memory cell. The circuit structure of a typical CMOS dual-port SRAM cell is shown in Fig. 10.35. Here, "word line 1" is used to access one set of complementary bit lines (bit line 1), while "word line 2" allows access to the other set of bit lines (bit line 2). The capability of simultaneous access eliminates wait states for the processors during "data read" operations. However, contention may still occur if both external processors accessing the same memory location simultaneously attempt to write data onto the accessed cell, or if one of the processors attempts to read data while the other processor

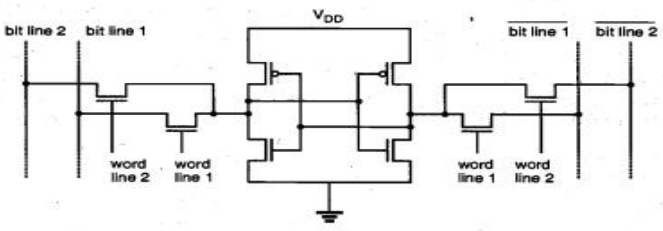


Figure 10.35. Circuit diagram of the CMOS dual-port SRAM cell.

writes data onto the same cell. In most cases, overlapping operations to the same memory location can be eliminated by a contention arbitration logic. It can either allow contention to be ignored and both operations to proceed, or it can arbitrate and delay one port until the operation on the other port is completed.