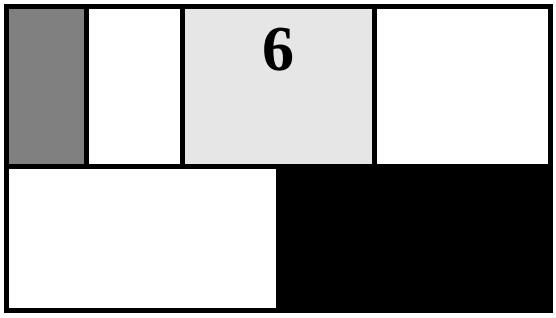
CHAPTER



DESIGNING COMBINATIONAL LOGIC GATES IN CMOS

In-depth discussion of logic families in CMOS—static and dynamic, pass-transistor, nonratioed and ratioed logic

n

Optimizing a logic gate for area, speed, energy, or robustness

n

Low-power and high-performance circuit-design techniques

6.1 Introduction

6.2 Static CMOS Design

- 6.2.1 Complementary CMOS
- 6.5 Leakage in Low Voltage Systems
- 6.2.2 Ratioed Logic
- 6.2.3 Pass-Transistor Logic

6.3 Dynamic CMOS Design

- 6.3.1 Dynamic Logic: Basic Principles
- 6.3.2 Speed and Power Dissipation of Dynamic Logic
- 6.3.3 Issues in Dynamic Design
- 6.3.4 Cascading Dynamic Gates

6.4 Perspective: How to Choose a Logic Style

6.6 Summary

6.7 To Probe Further

6.8 Exercises and Design Problems

6.1 Introduction

The design considerations for a simple inverter circuit were presented in the previous chapter. In this chapter, the design of the inverter will be extended to address the synthesis of arbitrary digital gates such as NOR, NAND and XOR. The focus will be on *combinational logic* (or *non-regenerative*) circuits that have the property that at any point in time, the output of the circuit is related to its current input signals by some Boolean expression (assuming that the transients through the logic gates have settled). No intentional connection between outputs and inputs is present.

In another class of circuits, known as *sequential* or *regenerative* circuits —to be discussed in a later chapter—, the output is not only a function of the current input data, but also of previous values of the input signals (Figure 6.1). This is accomplished by connecting one or more outputs intentionally back to some inputs. Consequently, the circuit "remembers" past events and has a sense of *history*. A sequential circuit includes a combinational logic portion and a module that holds the state. Example circuits are registers, counters, oscillators, and memory.

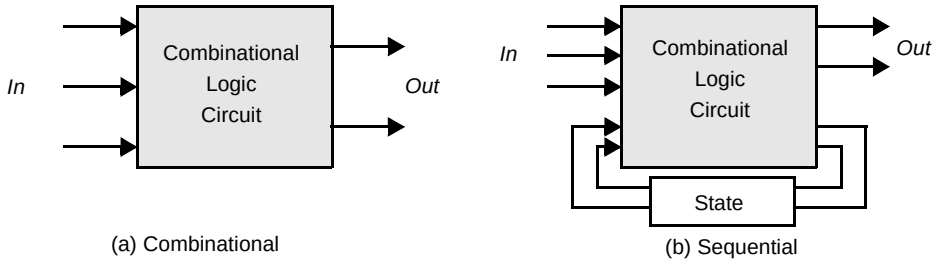


Figure 6.1 High level classification of logic circuits.

There are numerous circuit styles to implement a given logic function. As with the inverter, the common design metrics by which a gate is evaluated include area, speed, energy and power. Depending on the application, the emphasis will be on different metrics (e.g., in high performance processor, the switching speed of digital circuits is the primary metric while in a battery operated circuit it is the energy dissipation). In addition to these metrics, robustness to noise is also a very important consideration. We will see that certain logic styles (e.g., Dynamic logic) can significantly improve performance, but can be more sensitive to noise. Recently, power dissipation has also become a very important requirement and significant emphasis is placed on understanding the sources of power and approaches to deal with power.

6.2 Static CMOS Design

The most widely used logic style is static complementary CMOS. The static CMOS style is really an extension of the static CMOS inverter to multiple inputs. In review, the primary advantage of the CMOS structure is robustness (i.e, low sensitivity to noise), good performance, and low power consumption (with no static power consumption). As we will

see, most of those properties are carried over to large fan-in logic gates implemented using the same circuit topology.

The complementary CMOS circuit style falls under a broad class of logic circuits called *static* circuits in which at every point in time (except during the switching transients), each gate output is connected to either V_{DD} or V_{ss} via a low-resistance path. Also, the outputs of the gates assume at all times the value of the Boolean function implemented by the circuit (ignoring, once again, the transient effects during switching periods). This is in contrast to the *dynamic* circuit class, that relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes. The latter approach has the advantage that the resulting gate is simpler and faster. On the other hand, its design and operation are more involved than those of its static counterpart, due to an increased sensitivity to noise.

In this section, we sequentially address the design of various static circuit flavors including complementary CMOS, ratioed logic (pseudo-NMOS and DCVSL), and pass-transistor logic. The issues of scaling to lower power supply voltages and threshold voltages will also be dealt with.

6.2.1 Complementary CMOS

A static CMOS gate is a combination of two networks, called the *pull-up network* (PUN) and the *pull-down* network (PDN) (Figure 6.2). The figure shows a generic N input logic gate where all inputs are distributed to both the pull-up and pull-down networks. The function of the PUN is to provide a connection between the output and V_{DD} anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to V_{SS} when the output of the logic gate is meant to be 0. The PUN and PDN networks are constructed in a mutually exclusive fashion such that *one and only one* of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between V_{DD} and the output F, realizing a high output ("one"), or, alternatively, between V_{SS} and F for a low output ("zero"). This is equivalent to stating that the output node is always a *low-impedance* node in steady state.

In constructing the PDN and PUN networks, the following observations should be kept in mind:

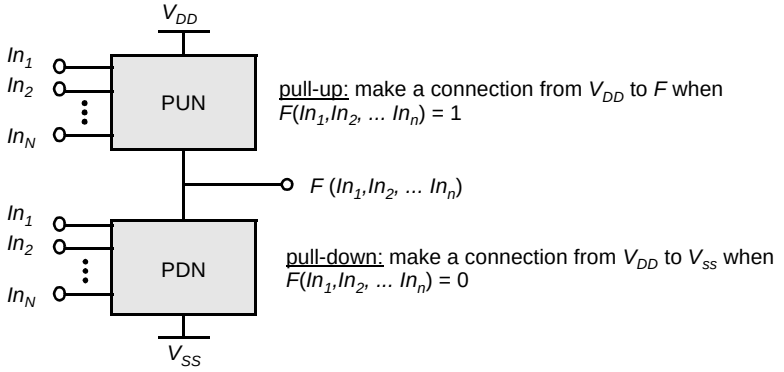
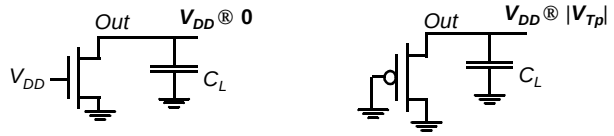
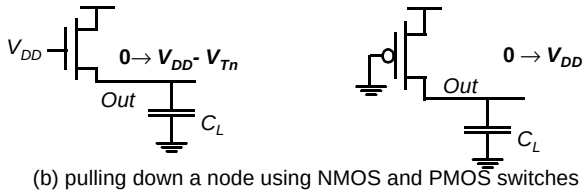


Figure 6.2 Complementary logic gate as a combination of a PUN (pull-up network) and a PDN (pull-down network).

- A transistor can be thought of as a switch controlled by its gate signal. An NMOS switch is *on* when the controlling signal is high and is *off* when the controlling signal is low. A PMOS transistor acts as an inverse switch that is *on* when the controlling signal is low and *off* when the controlling signal is high.
- The PDN is constructed using NMOS devices, while PMOS transistors are used in the PUN. The primary reason for this choice is that NMOS transistors produce "strong zeros," and PMOS devices generate "strong ones". To illustrate this, consider the examples shown in Figure 6.3. In Figure 6.3a, the output capacitance is initially charged to V_{DD}. Two possible discharge scenario's are shown. An NMOS device pulls the output all the way down to GND, while a PMOS lowers the output no further than $|V_{Tp}|$ — the PMOS turns *off* at that point, and stops contributing discharge current. NMOS transistors are hence the preferred devices in the PDN. Similarly, two alternative approaches to charging up a capacitor are shown in Figure 6.3b, with the output load initially at GND. A PMOS switch succeeds in charging the output all the way to V_{DD}, while the NMOS device fails to raise the output above V_{DD}-V_{Tn}. This explains why PMOS transistors are preferentially used in a PUN.



(a) pulling down a node using NMOS and PMOS switches



(b) pulling down a node using NMOS and PMOS switches

Figure 6.3 Simple examples illustrate why an NMOS should be used as a pull-down transistor, while a PMOS should be used as a pull-up device.

- A set of construction rules can be derived to construct logic functions (Figure 6.4). NMOS devices connected in series corresponds to an AND function. With all the inputs high, the series combination conducts and the value at one end of the chain is transfered to the other end. Similarly, NMOS transistors connected in parallel represent an OR function. A conducting path exists between the output and input terminal if at least one of the inpurs is high. Using similar arguments, construction rules for PMOS networks can be formulated. A series connection of PMOS conducts if both

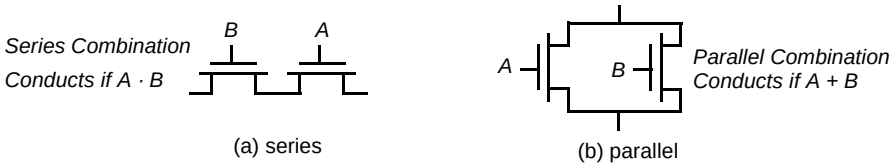


Figure 6.4 NMOS logic rules — series devices implement an AND, and parallel devices implement an OR.

inputs are low, representing a NOR function ($\overline{A}.\overline{B} = \overline{A+B}$), while PMOS transistors in parallel implement a NAND ($\overline{A}+\overline{B} = \overline{A \cdot B}$.

- Using De Morgan's theorems ($(\overline{A + B}) = A \cdot B$ and $\overline{A \cdot B} = \overline{A} + \overline{B}$), it can be shown that the pull-up and pull-down networks of a complementary CMOS structure are *dual* networks. This means that a parallel connection of transistors in the pull-up network corresponds to a series connection of the corresponding devices in the pull-down network, and vice versa. Therefore, to construct a CMOS gate, one of the networks (e.g., PDN) is implemented using combinations of series and parallel devices. The other network (i.e., PUN) is obtained using duality principle by walking the hierarchy, replacing series subnets with parallel subnets, and parallel subnets with series subnets. The complete CMOS gate is constructed by combining the PDN with the PUN.
- The complementary gate is naturally *inverting*, implementing only functions such as NAND, NOR, and XNOR. The realization of a non-inverting Boolean function (such as AND OR, or XOR) in a single stage is not possible, and requires the addition of an extra inverter stage.
- The number of transistors required to implement an N-input logic gate is $2N$.

Example 6.1 Two input NAND Gate

Figure 6.5 shows a two-input NAND gate ($F = \overline{A \cdot B}$). The PDN network consists of two NMOS devices in series that conduct when both A and B are high. The PUN is the dual network, and consists of two parallel PMOS transistors. This means that F is 1 if $A = 0$ or $B = 0$, which is equivalent to $F = \overline{A \cdot B}$. The truth table for the simple two input NAND gate is given in Table 6.1. It can be verified that the output F is always connected to either V_{DD} or GND, but never to both at the same time.

Example 6.2 Synthesis of complex CMOS Gate

Using complementary CMOS logic, consider the synthesis of a complex CMOS gate whose function is $F = \overline{D + A \cdot (B + C)}$. The first step in the synthesis of the logic gate is to derive the pull-down network as shown in Figure 6.6a by using the fact that NMOS devices in series implements the AND function and parallel device implements the OR function. The next step is to use duality to derive the PUN in a hierarchical fashion. The PDN network is broken into smaller networks (i.e., subset of the PDN) called sub-nets that simplify the derivation of the PUN. In Figure 6.6b, the subnets (SN) for the pull-down network are identified At the top level, SN1 and SN2 are in parallel so in the dual network, they will be in series. Since SN1

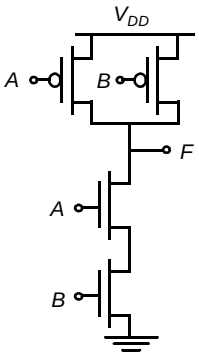
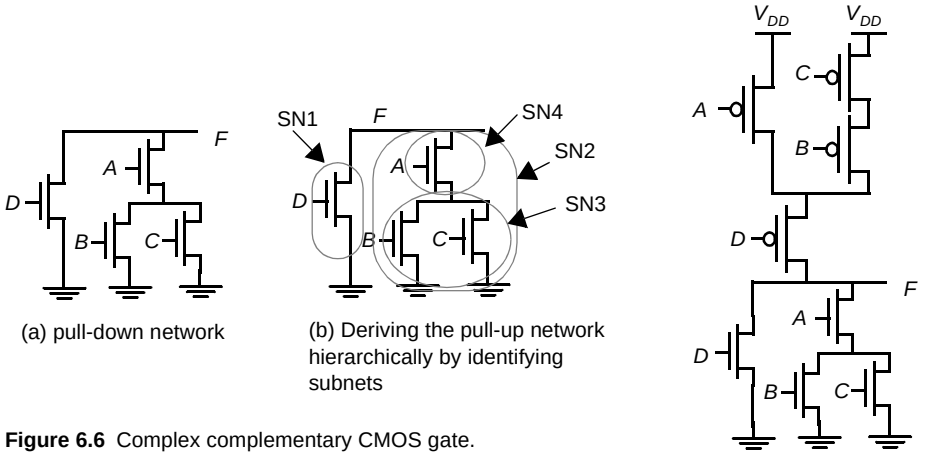


Table 6.1 Truth Table for 2 input NAND

A	*B*	*F*
0	0	1
0	1	1
1	0	1
1	1	0

Figure 6.5 Two-input NAND gate in complementary static CMOS style.

consists of a single transistor, it maps directly to the pull-up network. On the other hand, we need to recursively apply the duality rules to SN2. Inside SN2, we have SN3 and SN4 in series so in the PUN they will appear in parallel. Finally, inside SN3, the devices are in parallel so they will appear in series in the PUN. The complete gate is shown in Figure 6.6c. The reader can verify that for every possible input cobmination, there always exists a path to either V_{DD} or *GND*.



(a) pull-down network

(b) Deriving the pull-up network hierarchically by identifying subnets

(c) complete gate

Figure 6.6 Complex complementary CMOS gate.

Static Properties of Complementary CMOS Gates

Complementary CMOS gates inherit all the nice properties of the basic CMOS inverter, discussed earlier.They exhibit rail to rail swing with $V_{OH} = V_{DD}$ and $V_{OL} = GND$. The circuits also have no static power dissipation, since the circuits are designed such that the pull-down and pull-up networks are mutually exclusive. The analysis of the DC voltage transfer characteristics and the noise margins is more complicated then for the inverter, as these parameters depend upon the data input patterns applied to gate.

Consider the static two-input NAND gate shown in Figure 6.7. Three possible input combinations switch the output of the gate from high-to-low: (a) $A = B = 0 \rightarrow 1$, (b) $A = 1$,

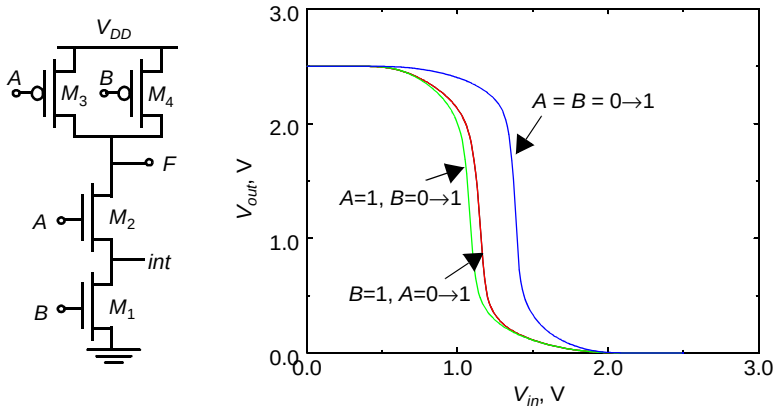


Figure 6.7 The VTC of a two-input NAND is data-dependent. NMOS devices are 0.5µm/0.25µm while the PMOS devices are sized at 0.75µm/0.25µm.

$B = 0 \rightarrow 1$, and (c) $B = 1$, $A = 0 \rightarrow 1$. The resulting voltage transfer curves display significant differences. The large variation between case (a) and the others (b & c) is explained by the fact that in the former case both transistors in the pull-up network are on simultaneously for A=B=0, representing a strong pull-up. In the latter cases, only one of the pull-up devices is on. The VTC is shifted to the left as a result of the weaker PUN.

The difference between (b) and (c) results mainly from the state of the internal node *int* between the two NMOS devices. For the NMOS devices to turn on, both gate-to-source voltages must be above V_{Tn}, with $V_{GS2} = V_A - V_{DS1}$ and $V_{GS1} = V_B$. The threshold voltage of transistor M_2 will be higher than transistor M_1 due to the body effect. The threshold voltages of the two devices are given by:

$$V_{Tn2} = V_{tn0} + \gamma((\sqrt{|2\phi_f| + V_{int}}) - \sqrt{|2\phi_f|}) \tag{6.1}$$

$$V_{Tn1} = V_{tn0} \tag{6.2}$$

For case (b), M_3 is turned *off*, and the gate voltage of M_2 is set to V_{DD}. To a first order, M_2 may be considered as a resistor in series with M_1. Since the drive on M_2 is large, this resistance is small and has only a small effect on the voltage transfer characteristics. In case (c), transistor M1 acts as a resistor, causing body effect in M_2. The overall impact is quite small as seen from the plot.

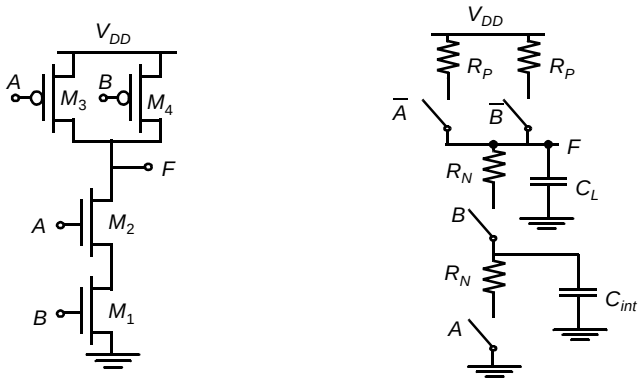
Design Consideration

The important point to take away from the above discussion is that the noise margins are input-pattern dependent. For the above example, a smaller input glitch will cause a transition at the output if only one of the inputs makes a transition. Therefore, this condition has a lower low noise margin. A common practice when characterizing gates such as NAND and NOR is to

connect all the inputs together. This unfortunately does not represent the worst-case static behavior. The data dependencies should be carefully modeled.

Propagation Delay of Complementary CMOS Gates

The computation of propagation delay proceeds in a fashion similar to the static inverter. For the purpose of delay analysis, each transistor is modeled as a resistor in series with an ideal switch. The value of the resistance is dependent on the power supply voltage and an equivalent large signal resistance, scaled by the ratio of device width over length, must be used. The logic is transformed into an equivalent RC network that includes the effect of internal node capacitances. Figure 6.8 shows the two-input NAND gate and its equivalent RC switch level model. Note that the internal node capacitance C_{int} —attributable to the source/drain regions and the gate overlap capacitance of M_2/M_1— is included. While complicating the analysis, the capacitance of the internal nodes can have quite an impact in some networks such as large fan-in gates.



(a) Two-input NAND (b) RC equivalent model

Figure 6.8 Equivalent RC model for a NAND gate.

We will initially ignore the effect of the internal capacitance (for a first pass). The most important observation is that delay is also dependent on the input patterns. Consider for instance the low-to-high transition. Three possible input scenarios can be identified for charging the output to V_{DD}. If both inputs are driven low, the two PMOS devices are on. The delay in this case is $0.69 \times (R_p/2) \times C_L$, since the two resistors are in parallel. This is not the worst-case low-to-high transition, which occurs when only one device turns on, and is given by $0.69 \times R_p \times C_L$. For the pull-down path, the output is discharged only if both A and B are switched high, and the delay is given by $0.69 \times (2R_N) \times C_L$ to a first order. In other words, adding devices in series slows down the circuit, and devices must be made wider to avoid a performance penalty. When sizing the transistors in a gate with multiple fan-in's, we should pick the combination of inputs that triggers the worst-case conditions.

For example, for a NAND gate to have the same pull-down delay delay (t_{phl}) as a minimum sized inverter (NMOS: 0.375μm/0.25μm and PMOS: 1.125μm/0.25μm), the

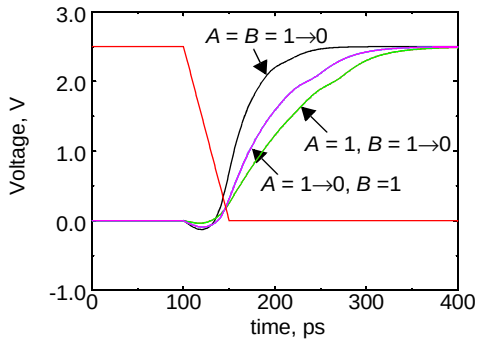
NMOS devices in the NAND stack must be made twice as large (i.e., NMOS of NAND should be 0.75μm/0.25μm) so that the equivalent resistance the NAND pull-down is the same as the inverter. The PMOS device can remain unchanged.

This first-order analysis assumes that the extra capacitance introduced by widening the transistors can be ignored. This is not a good assumption in general, but allows for a reasonable first cut at device sizing.

Example 6.3 Delay dependence on input patterns

Consider the NAND gate of Figure 6.8a. Assume NMOS and PMOS devices of 0.5μm/0.25μm and 0.75μm/0.25μm, respectively. This sizing should result in approximately equal worst-case rise and fall times (since the effective resistance of the pull-down is designed to be equal to the pull-up resistance).

Figure 6.9 shows the simulated low-to-high delay for different input patterns. As expected, the case where both inputs transition go low ($A = B = 1 \rightarrow 0$) results in a smaller delay, compared to the case where only one input is driven low. Notice that the worst-case low-to-high delay depends upon which input (A or B) goes low. The reason for this involves the internal node capacitance of the pull-down stack (i.e., the source of M_2). For the case that $B = 1$ and A transitions from $1 \rightarrow 0$, the pull-up PMOS device only has to charge up the output node capacitance since M_2 is turned off. On the other hand, for the case where A=1 and B transitions from $1 \rightarrow 0$, the pull-up PMOS device has to charge up the sum of the output and the internal node capacitances, which slows down the transition.



Input Data Pattern	Delay (pS)
A = B= 0®1	69
A = 1, B= 0®1	62
A= 0®1, B = 1	50
A=B=1®0	35
A=1, B = 1®0	76
A= 1®0, B = 1	57

Figure 6.9 Example showing the delay dependence on input patterns.

The table in Figure 6.9 shows a compilation of various delays for this circuit. The first-order transistor sizing indeed provides approximately equal rise and fall delays. An important point to note is that the high-to-low propagation delay depends on the state of the internal nodes. For example, when both inputs transition from $0 \rightarrow 1$, it is important to establish the state of the internal node. The worst-case happens when the internal node is charged up to V_{DD}-V_{Tn}. The worst case can be ensured by pulsing the A input from $1 \rightarrow 0 \rightarrow 1$, while input B only makes the $0 \rightarrow 1$. In this way, the internal node is initialized properly.

The important point to take away from this example is that estimation of delay can be fairly complex, and requires a careful consideration of internal node capacitances and data patterns. Care must be taken to model the worst-case scenario in the simulations. A brute force approach that applies all possible input patterns, may not always work as it is important to consider the state of internal nodes.

The CMOS implementation of a NOR gate ($F = \overline{A + B}$) is shown in Figure 6.10. The output of this network is high, if and only if both inputs A and B are low. The worst-case pull-down transition happens when only one of the NMOS devices turns on (i.e., if either A or B is high). Assume that the goal is to size the NOR gate such that it has approximately the same delay as an inverter with the following device sizes: NMOS 0.5μm/0.25μm and PMOS 1.5μm/0.25μm. Since the pull-down path in the worst case is a single device, the NMOS devices (M_1 and M_2) can have the same device widths as the NMOS device in the inverter. For the output to be pulled high, both devices must be turned on. Since the resistances add, the devices must be made two times larger compared to the PMOS in the inverter (i.e., M_3 and M_4 must have a size of 3μm/0.25μm). Since PMOS devices have a lower mobility relative to NMOS devices, stacking devices in series must be avoided as much as possible. A NAND implementation is clearly prefered over a NOR implementation for implementing generic logic.

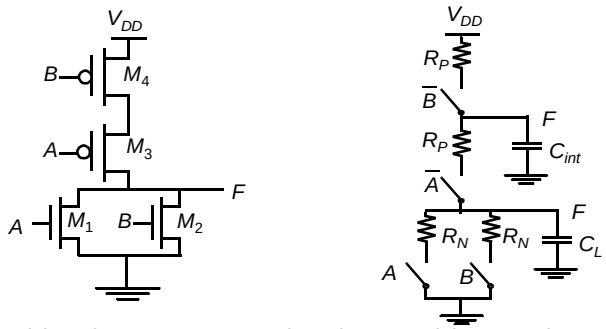


Figure 6.10 Sizing of a NOR gate to produce the same delay as an inverter with size of NMOS: 0.5μm/0.25μm and PMOS: 1.5μm/0.25μm.

Problem 6.1 Transistor Sizing in Complementary CMOS Gates

Determine the transistor sizes of the individual transistors in Figure 6.6c such that it has approximately the same t_{plh} and t_{phl} as a inverter with the following sizes: NMOS: 0.5μm/0.25μm and PMOS: 1.5μm/0.25μm.

So far in the analysis of propagation delay, we have ignored the effect of internal node capacitances. This is often a reasonable assumption for a first-order analysis. However, in more complex logic gates that have large *fan-in*, the internal node capacitances can become significant. Consider a 4-input NAND gate as shown in Figure 6.11, which shows the equivalent RC model of the gate, including the internal node capacitances. The internal capacitances consist of the junction capacitance of the transistors, as well as the gate-to-source and gate-to-drain capacitances. The latter are turned into capacitances to ground using the Miller equivalence. The delay analysis for such a circuit involves solving distributed RC networks, a problem we already encountered when analyzing the delay of interconnect networks. Consider the pull-down delay of the circuit. The output is discharged when all inputs are driven high. The proper initial conditions must be placed on the internal nodes (this is, the internal nodes must be charged to V_{DD}-V_{TN}) before the inputs are driven high.

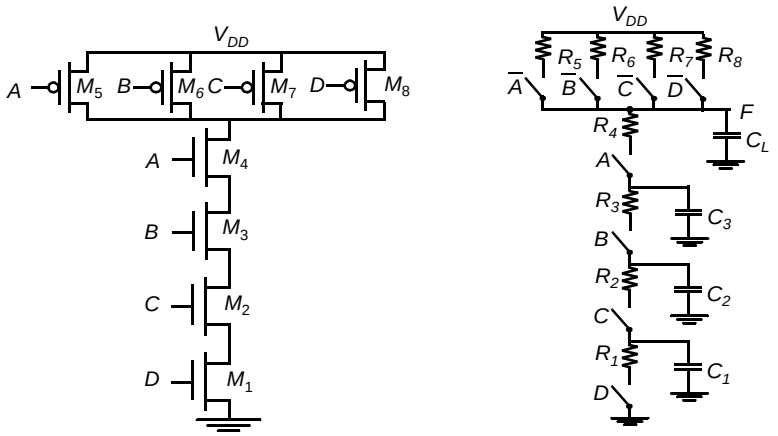


Figure 6.11 Four input NAND gate along with the internal node capacitances.

The propagation delay can be computed using the Elmore delay model and is approximated as:

$$t_{pHL} = 0.69(R_1 \cdot C_1 + (R_1 + R_2) \cdot C_2 + (R_1 + R_2 + R_3) \cdot C_3 + (R_1 + R_2 + R_3 + R_4) \cdot C_L) \tag{6.3}$$

Notice that the resistance of M_1 appears in all the terms, which makes this device especially important when attempting to minimize delay. Assuming that all NMOS devices have an equal size, Eq. (6.3) simplifies to

$$t_{pHL} = 0.69R_N(C_1 + 2 \cdot C_2 + 3 \cdot C_3 + 4 \cdot C_L) \tag{6.4}$$

Example 6.4 A Four-Input Complementary CMOS NAND Gate

In this example, the intrinsic *propagation delay* of the 4 input NAND gate (without any loading) is evaluated using hand analysis and simulation. Assume that all NMOS devices have a *W*/*L* of 0.5μm/0.25μm, and all PMOS devices have a device size of 0.375μm/0.25μm. The layout of a four-input NAND gate is shown in Figure 6.12. The devices are sized such that the worst case rise and fall time are approximately equal (to first order ignoring the internal node capacitances).

Using techniques similar to those employed for the CMOS inverter in Chapter 3, the capacitances values can be computed from the layout. Notice that in the pull-up path, the PMOS devices share the drain terminal in order to reduce the overall parasitic contribution to the the output. Using our standard design rules, the area and perimeter for various devices can be easily computed as shown in Table 6.1

In this example, we will focus on the pull-down delay, and the capacitances will be computed for the high-to-low transition at the output. While the output make a transition from V_{DD} to 0, the internal nodes only transition from V_{DD}-V_{Tn} to GND. We would need to linearlize the internal junction capacitances for this voltage transition, but, to simplify the analysis, we will use the same K_{eff} for the internal nodes as for the output node.

It is assumed that the output connects to a single, minimum-size inverter. The effect of intracell routing, which is small, is ignored. The different contributions are summarized in Table 6.2. For the NMOS and PMOS junctions, we use K_{eq} = 0.57, K_{eqsw} = 0.61, and K_{eq} = 0.79, K_{eqsw} = 0.86, respectively. Notice that the gate-to-drain capacitance is multiplied by a factor of two for all internal nodes and the output node to account for the Miller effect (this ignores the fact that the internal nodes have a slightly smaller swing due to the threshold drop).

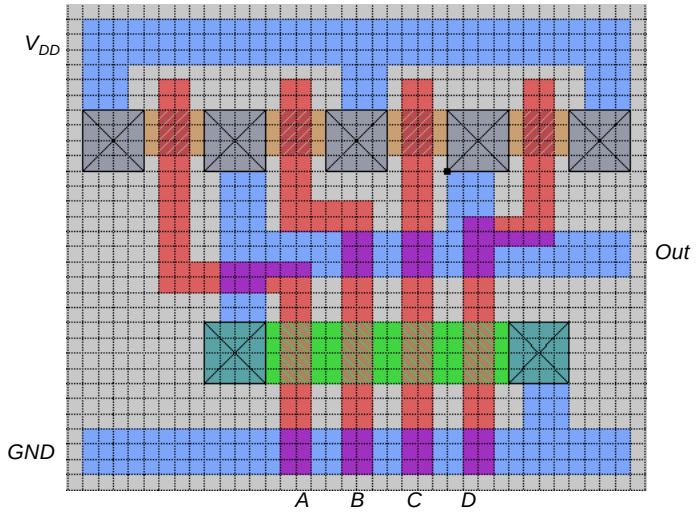


Figure 6.12 Layout a four-input NAND gate in complementary CMOS.

Table 6.1 Area and perimeter of various transistors for 4 input NAND gate.

Transistor	W (mm)	AS (mm^2)	AD (mm^2)	PS (mm)	PD(mm)
1	0.5	0.3125	0.0625	1.75	0.25
2	0.5	0.0625	0.0625	0.25	0.25
3	0.5	0.0625	0.0625	0.25	0.25
4	0.5	0.0625	0.3125	0.25	1.75
5	0.375	0.296875	0.171875	1.875	0.875
6	0.375	0.171875	0.171875	0.875	0.875
7	0.375	0.171875	0.171875	0.875	0.875
8	0.375	0.296875	0.171875	1.875	0.875

Using Eq. (6.4), we can compute the propagation delay as:

$$t_{pHL} = 0.69\left(\frac{13K\Omega}{2}\right)(0.85fF + 2 \cdot 0.85fF + 3 \cdot 0.85fF + 4 \cdot 3.47fF) = 85ps \qquad (6.5)$$

The simulated delay for this particular transition was found to be 86 psec! The hand analysis gives a fairly accurate estimate given all assumptions and linearizations made. For example, we assume that the gate-source (or gate-drain) capacitance only consists of the overlap component. This is not entirely the case, as during the transition some other contributions come in place depending upon the operating region. Once again, the goal of hand analysis is not to provide a totally accurate delay prediction, but rather to give intuition into what factors influence the delay and to aide in initial transistor sizing. Accurate timing analysis and transistor optimization is usually done using SPICE. The simulated worst-case low-to-high delay time for this gate was 106ps.

While complementary CMOS is a very robust and simple approach for implementing logic gates, there are two major problems associated with using this style as the com-

Table 6.2 Computation of capacitances (for high-to-low transition at the output). The circuit shows the intrinsic delay of the gate with no extra loading. Any *fan-out* capacitance would simply be added to the C_L term.

Capacitor	Contributions (H→L)	Value (fF) (H→L)
C_1	$C_{d1} + C_{s2} + 2 * C_{gd1} + 2 * C_{gs2}$	(0.57 * 0.0625 * 2+ 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2+ 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85fF
C_2	$C_{d2} + C_{s3} + 2 * C_{gd2} + 2 * C_{gs3}$	(0.57 * 0.0625 * 2+ 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2+ 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85fF
C_3	$C_{d3} + C_{s4} + 2 * C_{gd3} + 2 * C_{gs4}$	(0.57 * 0.0625 * 2+ 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2+ 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85fF
C_L	$C_{d4} + 2 * C_{gd4} + C_{d5} + C_{d6} + C_{d7} + C_{d8} + 2 * C_{gd5} + 2 * C_{gd6} + 2 * C_{gd7} + 2 * C_{gd8} = C_{d4} + 4 * C_{d5} + 4 * 2 * C_{gd6}$	(0.57 * 0.3125 * 2 + 0.61 * 1.75 *0.28) + 2 * (0.31 * 0.5)+ 4 * (0.79 * 0.171875* 1.9+ 0.86 * 0.875 * 0.22)+ 4 * 2 * (0.27 * 0.375) = 3.47fF

plexity of the gate (i.e., *fan-in*) increases. First, the number of transistors required to implement an *N* fan-in gate is *2N*. This can result in significant implementation area. The second problem is that propagation delay of a complementary CMOS gate deteriorates rapidly as a function of the fan-in. The large number of transistors (*2N*) increases the overall capacitance of the gate. For an *N*-input NAND gate, the output capacitance increases linearly with the fan-in since the number of PMOS devices connected to the output node increases linearly with the fan-in. Also, a series connection of transistors in either the PUN or PDN slows the gate as well, because the effective (dis)charging resistance is increased. For the same *N*-input NAND gate, the effective resistance of the PDN path increases linearly with the fan-in. Since the output capacitance increase linearly and the pull-down resistance increases linearly, the high-to-low delay can increase in a quadratic fashion.

The *fan-out* has a large impact on the delay of complementary CMOS logic as well. Each input to a CMOS gate connects to both an NMOS and a PMOS device, and presents a load to the driving gate equal to the sum of the gate capacitances.

The above observations are summarized by the following formula, which approximates the influence of *fan-in* and *fan-out* on the propagation delay of the complementary CMOS gate

$$t_p = a_1 FI + a_2 FI^2 + a_3 FO \tag{6.6}$$

where *FI* and *FO* are the *fan-in* and *fan-out* of the gate, respectively, and a_1, a_2 and a_3 are weighting factors that are a function of the technology.

At first glance, it would appear that the increase in resistance for larger *fan-in* can be solved by making the devices in the transistor chain wider. Unfortunately, this does not improve the performance as much as expected, since widening a device also increases its gate and diffusion capacitances, and has an adverse affect on the gate performance. For the *N*-input NAND gate, the low-to-high delay only increases linearly since the pull-up resistance remains unchanged and only the capacitance increases linearly.

Figure 6.13 show the propagation delay for both transitions as a function of fan-in assuming a fixed fan-out (NMOS: 0.5μm and PMOS: 1.5μm). As predicted above, the t_{pLH} increases linearly due to the linearly-increasing value of the output capacitance. The simultaneous increase in the pull-down resistance and the load capacitance results in an approximately quadratic relationship for t_{pHL}. Gates with a *fan-in* greater than or equal to 4 become excessively slow and must be avoided.

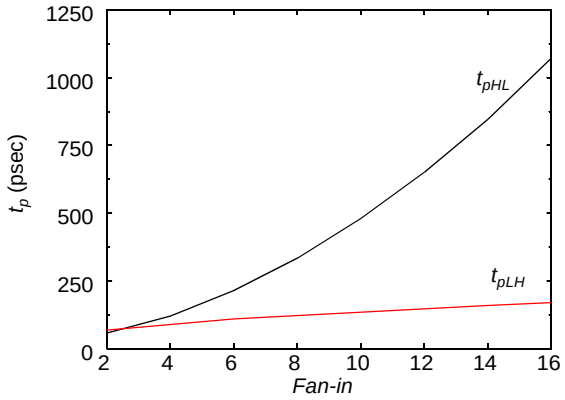


Figure 6.13 Propagation delay of CMOS NAND gate as a function of fan-in. A fan-out of one inverter is assumed, and all pull-down transistors are minimal size.

Design Techniques for Large Fan-in

Several approaches may be used to reduce delays in large fan-in circuits.

1. Transistor Sizing

The most obvious solution is to increase the overall transistor size. This lowers the resistance of devices in series and lowers the time constant. However, increasing the transistor size, results in larger parasitic capacitors, which do not only affect the *propagation delay* of the gate in question, but also present a larger load to the preceding gate. This technique should, therefore, be used with caution. If the load capacitance is dominated by the intrinsic capacitance of the gate, widening the device only creates a "self-loading" effect, and the *propagation delay* is unaffected.

2. Progressive Transistor Sizing

An alternate approach to uniform sizing (in which each transistor is scaled up uniformly), is to use progressive transistor sizing (Figure 6.14). Refering back to Eq. (6.3), we see that the resistance of M_1 (R_1) appears N times in the delay equation, the resistance of M_2 (R_2) appears N-1 times, etc. From the equation, it is clear that R_1 should be made the smallest, R_2 the next smallest, etc. Consequently, a progressive scaling of the transistors is beneficial: $M_1 > M_2 > M_3 > M_N$. Basically, in this approach, the important resistance is reduced while reducing capacitance. For an excellent treatment on the optimal sizing of transistors in a complex network, we refer the interested reader to [Shoji88, pp. 131–143].

3. Input Re-Ordering

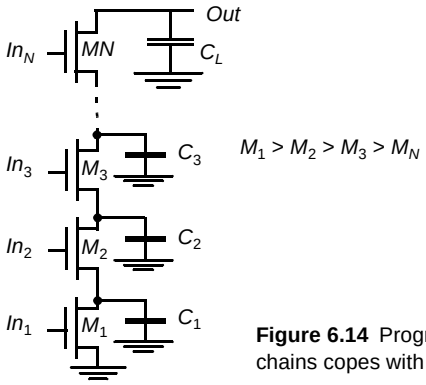


Figure 6.14 Progressive sizing of transistors in large transistor chains copes with the extra load of internal capacitances.

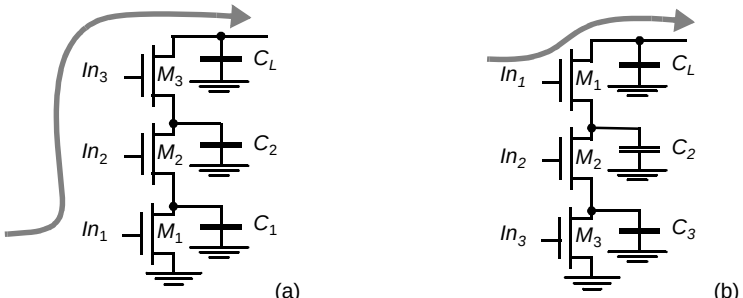


Figure 6.15 Influence of transistor ordering on delay. Signal In_1 is the critical signal.

Some signals in complex combinational logic blocks might be more critical than others. Not all inputs of a gate arrive at the same time (due, for instance, to the propagation delays of the preceding logical gates). An input signal to a gate is called *critical* if it is the last signal of all inputs to assume a stable value. The path through the logic which determines the ultimate speed of the structure is called the *critical path*.

Putting the critical-path transistors closer to the output of the gate can result in a speed-up. This is demonstrated in Figure 6.15. Signal In_1 is assumed to be a critical signal. Suppose further that In_2 and In_3 are high and that In_1 undergoes a $0 \rightarrow 1$ transition. Assume also that C_L is initially charged high. In case (a), no path to *GND* exists until M_1 is turned on, which is unfortunately the last event to happen. The delay between the arrival of In_1 and the output is therefore determined by the time it takes to discharge C_L, C_1 and C_2. In the second case, C_1 and C_2 are already discharged when In_1 changes. Only C_L still has to be discharged, resulting in a smaller delay.

4. Logic Restructuring

Manipulating the logic equations can reduce the fan-in requirements and hence reduce the gate delay, as illustrated in Figure 6.16. The quadratic dependency of the gate delay on *fan-in* makes the six-input NOR gate extremely slow. Partitioning the NOR-gate into two three-input gates results in a significant speed-up, which offsets by far the extra delay incurred by turning the inverter into a two-input NAND gate.

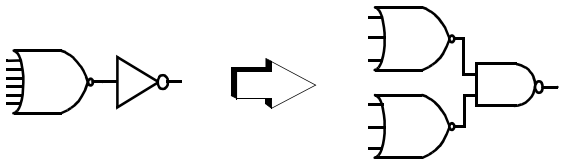


Figure 6.16 Logic restructuring can reduce the gate fan-in.

Power Consumption in CMOS Logic Gates

The sources of power consumption for the complementary CMOS inverter was discussed in detail. Many of issues apply directly to complex CMOS gates. The power dissipation is a strong function of transistor sizing (which affects physical capacitance), input and output rise/fall times (which affects the short-circuit power), device thresholds and temperature (which affect leakage power) and switching activity. The switching power of a CMOS gate is given by $a_{0\rightarrow1}\,C_L\,V_{DD}^2\,f$ and this section will focus of on the *switching activity ($a_{0\rightarrow1}$)* of a logic gate. There are two components to switching activity: a static component (which does not take into account the timing behavior) and a dynamic (or glitching) component (which takes into account the timing behavior of the circuit). The major factors that affect activity is listed below.

Logic Function —The amount of transition activity is a strong function of the logic function being implemented. In static CMOS gates, the static transition probability assuming independent inputs is the probability that the output will be in the zero state in one cycle multiplied by the probability that the output will be in the one state in the next cycle:

$$\alpha_{0\rightarrow1} = p_0 \bullet p_1 = p_0 \bullet (1 - p_0) \tag{6.7}$$

where p_0 is the probability that the output is in the *zero* state and p_1 is the probability that the output will is in the *one* state. Assuming that the inputs are independent and uniformly distributed, any *N*-input static gate will have a transition probability that corresponds to:

$$\alpha_{0\rightarrow1} = \frac{N_0}{2^N} \bullet \frac{N_1}{2^N} = \frac{N_0 \bullet \left(2^N - N_0\right)}{2^{2N}} \tag{6.8}$$

where N_0 is the number of *zero* entries and N_1 is the number of *one* entries in the truth table for the output of the *N*-input function. To illustrate, consider a static 2-input NOR gate whose truth table is shown in Table 6.3. Assume that only one input transition is possible during a clock cycle and that the inputs to the NOR gate have a uniform input distribution (i.e., the four possible states for inputs *A* and *B* (00, 01, 10, 11) are equally likely).

Table 6.3 Truth table of a 2 input NOR gate.

A	*B*	*Out*
0	0	1
0	1	0

Table 6.3 Truth table of a 2 input NOR gate.

A	*B*	*Out*
1	0	0
1	1	0

From Table 6.3 and Eq. (6.8), the output transition probability of a 2-input static CMOS NOR gate is given by:

$$\alpha_{0\rightarrow 1} = \frac{N_0 \bullet \left(2^N - N_0\right)}{2^{2N}} = \frac{3 \bullet \left(2^2 - 3\right)}{2^{2 \bullet 2}} = \frac{3}{16} \tag{6.9}$$

Problem 6.2 *N* input XOR gate

Assuming the inputs to an *N*-input XOR gate are uncorrelated and uniformly distributed, derive the expression for the switching activity factor.

***Signal Statistics*—**The switching activity of a logic gate is a strong function of the signal statistics. Using a uniform input distribution to compute activity is not a good one since the propagation through logic gates can significantly modify the signal statistics. For example, consider once again a 2-input static NOR gate, and let p_a and p_b be the probabilities that the inputs *A* and *B* are one. Assume that the inputs are not correlated. The probability that the output node is a one is given by:

$$p_1 = (1\text{-}p_a)\ (1\text{-}p_b) \tag{6.10}$$

Therefore, the probability of a transition from 0 to 1 is:

$$a_{0\text{->}1} = p_0\, p_1 = (1\text{-}(1\text{-}p_a)\ (1\text{-}p_b))\ (1\text{-}p_a)\ (1\text{-}p_b) \tag{6.11}$$

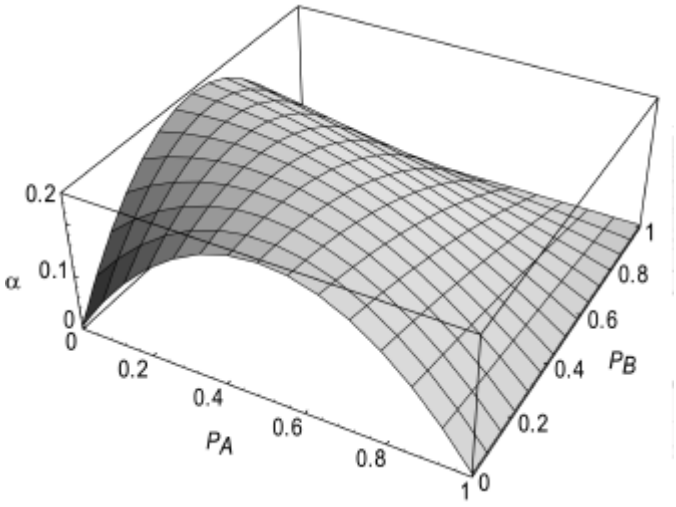


Figure 6.17 Transition activity of a two-input NOR gate as a function of the input probabilities (p_A ,p_B)

Figure 6.17 shows the transition probability as a function of p_a and p_b. Observe how this graph degrades into the simple inverter case when one of the input probabilities is set to 0. From this plot, it is clear that understanding the signal statistics and their impact on switching events can be used to significantly impact the power dissipation.

Problem 6.3 Power Dissipation of Basic Logic Gates

Derive the $0 \rightarrow 1$ output transition probabilities for the basic logic gates (AND, OR, XOR). The results to be obtained are given in Table 6.4.

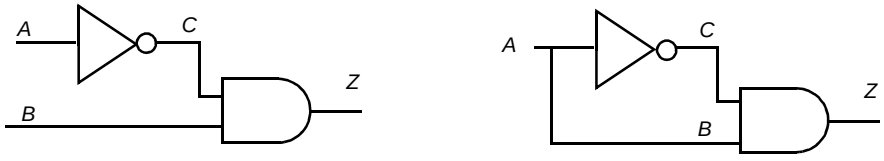
Table 6.4 Output transition probabilities for static logic gates.

	$a_{\mathbf{0\rightarrow 1}}$
AND	$(1 - p_A p_B) p_A p_B$
OR	$(1 - p_A)(1 - p_B)[1 - (1 - p_A)(1 - p_B)]$
XOR	$[1 - (p_A + p_B - 2p_A p_B)](p_A + p_B - 2p_A p_B)$

Inter-signal Correlations—The evaluation of the switching activity is further complicated by the fact that signals exhibit correlation in space and time. Even if the primary inputs to a logic network are uncorrelated, the signals become correlated or 'colored', as they propagate through the logic network. The example of Figure 6.18 provides a simple example.. Consider the circuit shown in Figure 6.18a, and assume that the primary inputs, *A* and *B*, are uncorrelated and uniformly distributed. Node *C* has a 1 (0) probability of 1/2, and a 0->1 transition probability of 1/4. The probability that the node *Z* undergoes a power consuming transition is then determined using the AND-gate expression of Table 6.4.

$$p_{0\text{->}1} = (1 - p_a\, p_b)\, p_a\, p_b = (1 - 1/2 \cdot 1/2)\, 1/2 \cdot 1/2 = 3/16 \tag{6.12}$$

The computation of the probabilities is straightforward: signal and transition probabilities are evaluated in an ordered fashion, progressing from the input to the output node. This approach, however, has two major limitations: (1) it does not deal with circuits with feedback, as found in sequential circuits; (2) it assumes that the signal probabilities at the input of each gate are independent. This is rarely the case in actual circuits, where reconvergent fanout often causes inter-signal dependencies. Fo instance, the inputs to the AND gate in Figure 6.18b (*C* and *B*) are interdependent, as both are a function of *A*. The



(a) Logic circuit without reconvergent fanout

(b) Logic circuit with reconvergent fanout

Figure 6.18 Example illustrating the effect of signal correlations.

approach to compute probabilities, presented previously, fails under these circumstances. Traversing from inputs to outputs yields a transition probability of 3/16 for node Z , similar to the previous analysis. This value for transition probability is clearly false, as logic transformations show that the network can be reduced to $Z = C \bullet B = A \bullet \overline{A} = 0$, and no transition will ever take place.

To get the precise results in the progressive analysis approach, its is essential to take signal inter-dependencies into account. This can be accomplished with the aid of conditional probabilities. For an AND gate, Z equals 1 if and only if B and C are equal to 1.

$$p_Z = p(Z=1) = p(B=1, C=1) \tag{6.13}$$

where $p(B=1,C=1)$ represents the probability that B and C are equal to 1 simultaneously. If B and C are independent, $p(B=1,C=1)$ can be decomposed into $p(B=1) \bullet p(C=1)$, and this yields the expression for the AND-gate, derived earlier: $p_Z = p(B=1) \bullet p(C=1) = p_B\, p_C$. If a dependency between the two exists (as is the case in Figure 6.18b), a conditional probability has to be employed, such as

$$p_Z = p(C=1|B=1) \bullet p(B=1) \tag{6.14}$$

The first factor in Eq. (6.14) represents the probability that C=1 given that B=1. The extra condition is necessary as C is dependent upon B. Inspection of the network shows that this probability is equal to 0, since C and B are logical inversions of each other, resulting in the signal probability for Z, $p_Z = 0$.

Deriving those expressions in a structured way for large networks with reconvergent fanout is complex, especially when the networks are contain feedback loops. Computer support is therefore essential. To be meaningful, the analysis program has to process a typical sequence of input signals, as the power dissipation is a strong function of statistics of those signals.

Dynamic or Glitching Transitions—When analyzing the transition probabilities of complex, multistage logic networks in the preceding section, we ignored the fact that the gates have a non-zero propagation delay. In reality, the finite propagation delay from one logic block to the next can cause spurious transitions, called *glitches, critical races, or dynamic hazards*, to occur: a node can exhibit multiple transitions in a single clock cycle before settling to the correct logic level.

A typical example of the effect of glitching is shown in Figure 6.19, which displays the simulated response of a chain of NAND gates for all inputs going simultaneously from 0 to 1. Initially, all the outputs are 1 since one of the inputs was 0. For this particular transition, all the odd bits must transition to 0 while the even bits remain at the value of 1. However, due to the finite propagatin delay, the higher order even outputs start to discharge and the voltage drops. When the correct input ripples through the network, the output goes high. The glitch on the even bits causes extra power dissipation beyond what is required to strictly implement the logic function. Although the glitches in this example are only partial (i.e., not from rail to rail), they contribute significantly to the power dissipation. Long chains of gates often occur in important structures such as adders and multipliers and the glitching component can easily dominate the overall power consumption.

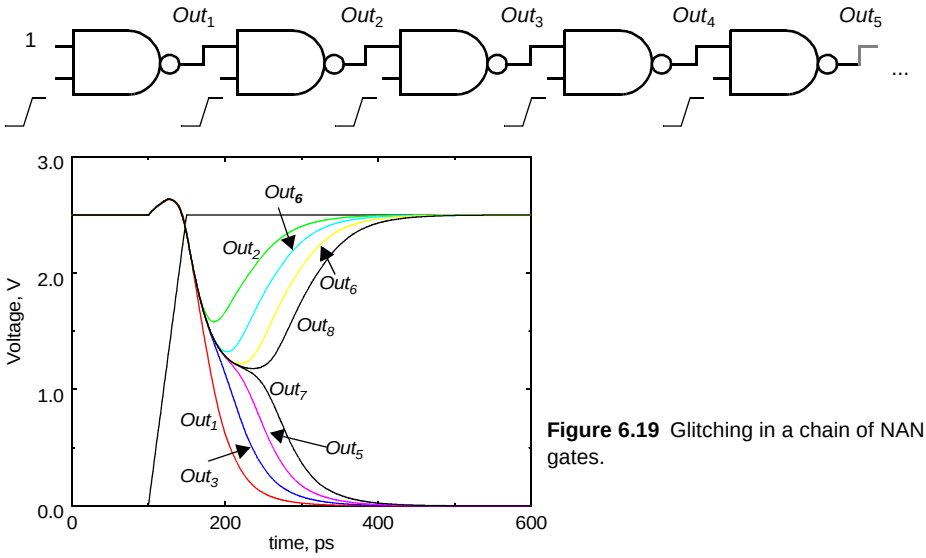


Figure 6.19 Glitching in a chain of NAND gates.

Design Techniques to Reduce Switching Activity

The dynamic power of a logic gate can be reduced by minimizing the physical capacitance and the switching activity. The physical capacitance can be minimized in a number ways, including circuit style selection, transistor sizing, placement and routing, and architectural optimizations. The switching activity, on the other hand, can be minimized at all level of the design abstraction, and is the focus of this section. Logic structures can be optimized to minimize both the fundamental transitions required to implement a given function, and the spurious transitions. This can be accomplished in the following ways:

1. Logic Restructuring

The topology of a logic network can affect the overall power dissipation. To illustrate this point consider two alternate implementations of $F = A \bullet B \bullet C \bullet D$, as shown in Figure 6.20. Ignore glitching and assume that all primary inputs (A,B,C,D) are uncorre-

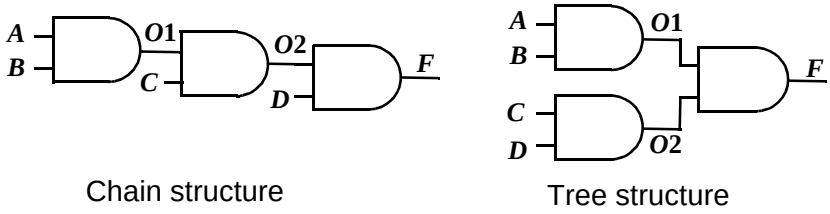


Figure 6.20 Simple example to demonstrate the influence of circuit topology on activity.

lated and uniformly distributed (i.e., $p_{1\ (a,b,c,d)} = 0.5$). For an AND gate, the probability that the output is 1 is $p_1 = p_a\, p_b$ and the transition probability is:

$$a_{0->1} = p_0\, p_1 = p_0\,(1\text{-}p_0) = (1\text{-}p_a\, p_b)\, p_a\, p_b \tag{6.15}$$

Given this, the activity can be computed for the two topologies as shown in Table 6.5. The results indicate that the chain implementation will have an overall lower switching activity than the tree implementation for random inputs. However, as mentioned before, it is also important to consider the timing behavior to accurately make power trade-offs. In this example the tree topology will have lower (no) glitching activity since the signal paths are balanced to all the gates.

Table 6.5 Probabilities for tree and chain topologies.

	*O*1	*O*2	*F*
p_1 (chain)	1/4	1/8	1/16
$p_0 = 1\text{-}p_1$ (chain)	3/4	7/8	15/16
$p_{0->1}$ (chain)	3/16	**7/64**	15/256
p_1 (tree)	1/4	1/4	1/16
$p_0 = 1\text{-}p_1$ (tree)	3/4	3/4	15/16
$p_{0->1}$ (tree)	3/16	**3/16**	15/256

2. Input ordering

Consider the two static logic circuits of Figure 6.21. The probabilities of *A*, *B* and C being 1 is listed in the figure. Since both circuits implement identical logic functionality, it is obvious that the activity at the output node *Z* is equal in both cases. The difference is the activity at the intermediate node. In the first circuit, this activity equals $(1 - 0.5 \times 0.2)\,(0.5 \times 0.2) = 0.09$. In the second case, the probability that a $0 \rightarrow 1$ transition occurs equals $(1 - 0.2 \times 0.1)\,(0.2 \times 0.1) = 0.0196$. This is substantially lower. From this we learn that it is beneficial to postpone the introduction of signals with a high transition rate (i.e., signals with a signal probability close to 0.5). A simple reordering of the input signals is often sufficient to accomplish that goal.

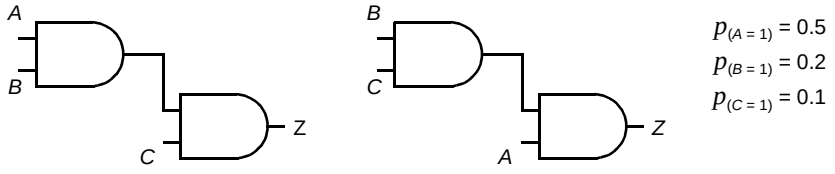
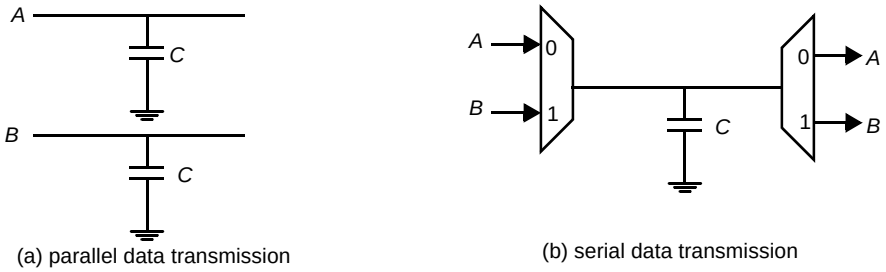


Figure 6.21 Reordering of inputs affects the circuit activity.

3. Time-multiplexing resources

Another important design consideration is the amount of resources required to implement a given function. To conserve area, it is often desirable to minimize the amount of physical hardware (logic units or data busses). Unfortunately, the minimum area solution does not always result in the lowest switching activity. For example, consider the transmission of two input bits (*A* and *B*) using dedicated resources and a time-multiplexed approach as shown in Figure 6.22. To first order, it would seem that the degree of time-multiplexing should not affect the overall switched capacitance since the time multiplexed solution has half the capacitance switched at twice the frequency (for a fixed throughput).

If data being transmitted were random, it will make no difference what architecture is used. However if data is not correlated, the power dissipation of the time-multiplexed solution can be significantly higher. For example, suppose *A* was mostly low and *B* was mostly high. In the parallel solution, the switched capacitance should be very low since there are very few transitions on the data bits. However, in the time-multiplexed solution, the bus is going to toggle between 0 and 1. Care must be taken in digital systems to avoid time-multiplexing data stream that are not correlated.

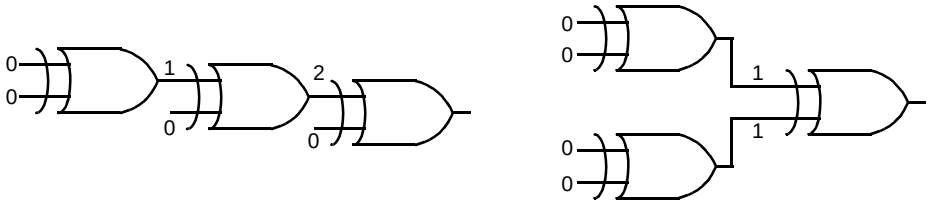


(a) parallel data transmission (b) serial data transmission

Figure 6.22 Parallel vs. time-multiplexed data busses.

4. Glitch Reduction by balancing signal paths

The occurrence of glitching in a circuit is mainly due to a mismatch in the path lengths in the network. If all input signals of a gate change simultaneously, no glitching occurs. On the other hand, if input signals change at different times, a dynamic hazard might develop. Such a mismatch in signal timing is typically the result of different path lengths with respect to the primary inputs of the network. This is illustrated in Figure 4.20.



(a) Network sensitive to glitching (b) Glitch-free network

Figure 6.23 Glitching is influenced by matching of signal path lengths. The annotated numbers indicate the signal arrival times.

Assume that the XOR gate has a unit delay. The first network (a) suffers from glitching as a result of the wide disparity between the arrival times of the input signals for a gate. For example, for gate F_3, one input settles at time 0, while the second one only arrives at time

2. Redesigning the network so that all arrival times are identical can dramatically reduce the number of transitions (network b).

6.2.2 Ratioed Logic

The CMOS logic style described in the previous section is highly robust and scalabe with technology, but requires 2*N* transistors to implement a *N*-input logic gate. Also, the load capacitance is significant since each gate drives two devices (a PMOS and an NMOS) per *fan-out*. Ratioed logic is an attempt to reduce the number of transistors required to implement a given logic function, at the cost of reduced robustness and extra power dissipation. The purpose of the PUN in complementary CMOS is to provide a conditional path between V_{DD} and the output when the PDN is turned *off*. In ratioed logic, the entire PUN is replaced with a single load device that pulls up the output when the PDN is turned off.

Figure 6.24 shows an example of ratioed logic which uses a grounded PMOS load and referred to as a pseudo-NMOS style. Instead of a combination of active pull-down and pull-up networks, such a gate consists of an NMOS pull-down network that realizes the *logic function*, and a simple *load device*.

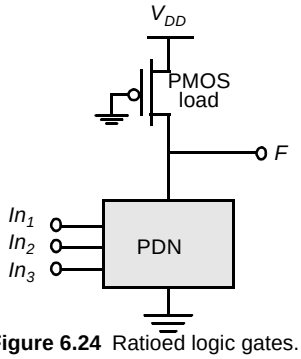


Figure 6.24 Ratioed logic gates.

The clear advantage of pseudo-NMOS is the reduced number of transistors (*N*+1 vs. 2*N* for complementary CMOS). The nominal high output voltage (V_{OH}) for this gate is V_{DD} since the pull-down devices is turned *off* when the output is pulled high (assuming that V_{OL} is below V_{Tn}). On the other hand, the nominal low output voltage is not 0V since there is a fight between the devices in the PDN and the load grounded PMOS device. This results in reduced noise margins and more importantly static power dissipation. The sizing of the load device relative to the pull-down devices can be used to trade-off parameters such a *noise margin*, *propagation delay* and *power dissipation*. Since the voltage swing on the output and overall functionality of the gate is dependent on the device size, the circuit is called *ratioed*. This is in contrast to the *ratioless* logic styles, such as complementary CMOS, where the low and high levels do not depend upon transistor sizes.

Computing the dc transfer characteristic of the pseudo-NMOS proceeds along paths similar to those used for its complementary CMOS counterpart. The value of V_{OL} is obtained by equating the currents through the driver and load devices for $V_{in} = V_{DD}$. At

this operation point, it is reasonable to assume that the NMOS device resides in linear mode (since the output should ideally be close to 0V), while the PMOS load is saturated.

$$k_n\left((V_{DD}-V_{Tn})V_{OL}-\frac{V_{OL}^2}{2}\right) = k_p\left((-V_{DD}-V_{Tp})\cdot V_{DSAT}-\frac{V_{DSAT}^2}{2}\right) \tag{6.16}$$

Assuming that V_{OL} is small relative to the gate drive (V_{DD}-V_T) and that V_{Tn} is equal to V_{Tp} in magniture, V_{OL} can be approximated as:

$$V_{OL} \approx \frac{k_p(-V_{DD}-V_{Tp})\cdot V_{DSAT}}{k_n(V_{DD}-V_{Tn})} \approx \frac{\mu_p \cdot W_p}{\mu_n \cdot W_n}\cdot |V_{DSAT}| \tag{6.17}$$

In order to make V_{OL} as small as possible, the PMOS device should be sized much smaller than the NMOS pull-down devices. Unfortunately, this has a negative impact on the *propagation delay* for charging up the output node since the current provided by the PMOS device is limited.

An important disadvantage of pseudo-NMOS gates is static power that happens when the output is low, because a direct current path exists between V_{DD} and *GND* through the load and driver devices. The static power consumption in the low-output mode is easily derived

$$P_{low} = V_{DD}I_{low} \approx V_{DD}\cdot k_p\left((-V_{DD}-V_{Tp})\cdot V_{DSATp}-\frac{V_{DSATp}^2}{2}\right) \tag{6.18}$$

Example 6.5 Pseudo-NMOS Inverter

Consider a simple pseudo-NMOS inverter (where the PDN network in Figure 6.24 degenerates to a single transistor) with an NMOS size of 0.5µm/0.25µm. The effect of sizing the PMOS device is studied in this example to demonstrate the impact on various parameters. The *W*/*L* ratio of the grounded PMOS is varied for values of 4, 2, 1, 0.5 and 0.25. The devices less than $W/L < 1$ is contructed by making the length longer than the width. The voltage transfer curve for the different sizes is plotted in Figure 6.25.

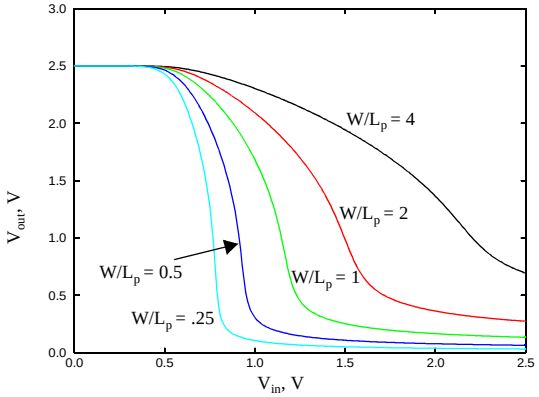


Figure 6.25 Voltage transfer curves for sizes of the pseudo-NMOS devices.

Table 6.6 summarizes the nominal output voltage (V_{OL}), static power dissipation, and the low-to-high propagation delay. The low-to-high delay is measured as the time to

Table 6.6 Performance of a pseudo-NMOS inverter.

Size	V_{OL}	Static Power Dissipation	t_{plh}
4	0.693V	564µW	14ps
2	0.273V	298µW	56ps
1	.133V	160µW	123ps
0.5	0.064V	80µW	268ps
0.25	0.031V	41µW	569ps

reach 1.25V from V_{OL} (which is not 0V for this inverter). This is chosen since the load gate is a CMOS inverter with a switching threshold of 1.25V. The trade-off between the static and dynamic properties is clearly illustrated. A larger pull-up device improves performance, but increases static power dissipation and lower noise margins (i.e., higher V_{OL}).

Notice that the simple first order model to predict V_{OL} is reasonably valid. For a PMOS W/L of 4, V_{OL} is given by (30/115) (4) (0.63V) = 0.66V.

The static power dissipation of pseudo-NMOS has limited its use. However, pseudo-NMOS still finds use in large fan-in circuits. Figure 6.26 shows the implementation of pseudo-NMOS NOR and NAND gates. When area is most important, such an approach is attractive.

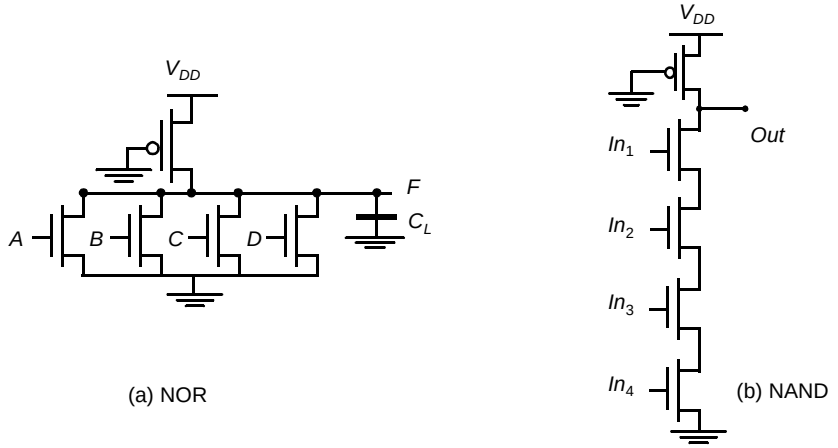


Figure 6.26 Four-input pseudo-NMOS NOR and NANDgates.

Problem 6.4 NAND Versus NOR in Pseudo-NMOS

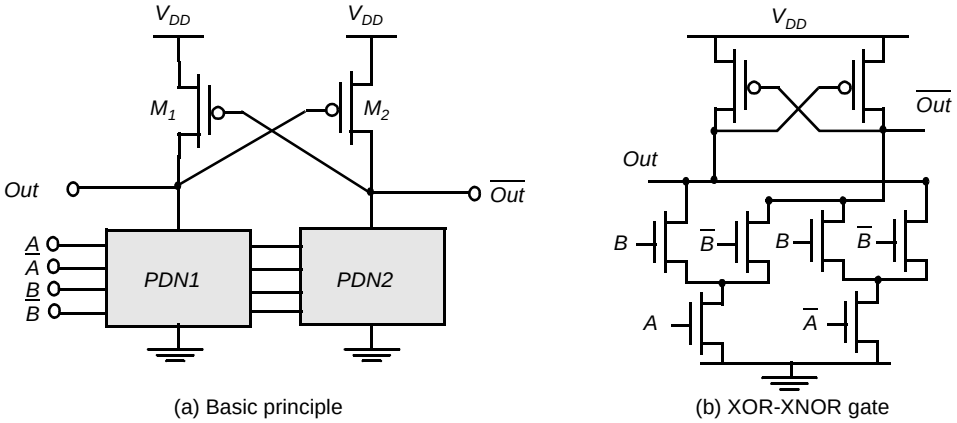
Given the choice between NOR or NAND logic, which one would you prefer for implementation in pseudo-NMOS?

How to Build Even Better Loads

It is possible to create a ratioed logic style that allows us to completely eliminate static currents and provide rail-to-rail swing. This requires the use of feedback concepts. In this particular style of logic, complementary inputs are fed into the gate and the gates

provide complementary outputs. Such a gate, called *Differential Cascade Voltage Switch Logic* (or DCVSL) is presented conceptually in Figure 6.27a.

The pull-down networks PDN1 and PDN2 are designed using NMOS devices and are mutually exclusive (i.e., when PDN1 conducts, PDN2 is off and when PDN1 is off, PDN2 conducts). The mutually exclusive pull-down devices allow the implementation of the required logic function and its inverse. Assume now that, for a given set of inputs, PDN1 conducts while PDN2 does not. Also assume that *Out* was initially high and $\overline{Out}$ intially low. Node *Out* is pulled down and intially there is a fight between M_1 and PDN1 PMOS as the pull-down device is turned *on*. Notice that initially, $\overline{Out}$ is actually in a high impedence state since both M_2 and PDN2 are turned *off*. PDN1 must be strong enough to bring *Out* down to V_{DD}-$|V_{Tp}|$, at which point, M_2 turns *on* and charges *Out* to V_{DD}. This in turn enables *Out* to discharge all the way to *GND*. The circuit is still ratioed since the sizing of the PMOS devices relative to the pull-down devices is critical to functionality, not just performance. Figure 6.27b shown an example of an XOR/XNOR gate. Notice that it is possible to share transistors among the two pull-down networks.



(a) Basic principle (b) XOR-XNOR gate

Figure 6.27 DCVSL logic gate.

In addition to the problem of increase complexity in design, this circuit style has the problem of increased power dissipation due to coss-over current. There is a period of time when the PMOS and PDN is turned on simulatneously, producing a short circuit path. However, notice that the static power dissipation has been eliminated since in steady state, one of the pull-down networks and other PMOS device are turned off.

Example 6.6 DCVSL Transient Response

An example transient response is shown for an AND/NAND gate in DCVSL. Notice that as *Out* is pulled down to V_{DD}-$|V_{Tp}|$, $\overline{Out}$ starts to charge up to V_{DD} quickly. The

delay from the input to *Out* is 197ps and to $\overline{Out}$ is 321 ps. A static CMOS AND gate (NAND followed by an inverter) has a delay of 200ps.

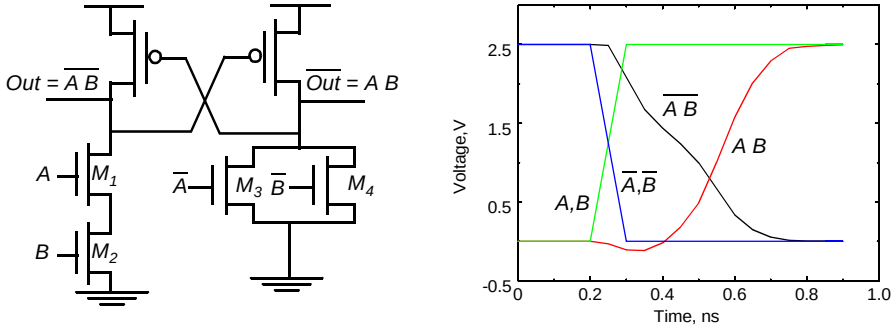


Figure 6.28 Transient response of a simple AND/NAND DCVSL gate. M_1 and M_2 1µm/0.25µm, M_3 and M_4 are 0.5µm/0.25µm and the cross-coupled PMOS device are 1.5µm/0.25µm.

6.2.3 Pass-Transistor Logic

Pass-Transistor Basics

A popular and widely used alternative to complementary CMOS is pass transistor logic. Pass transistor logic attempts to reduce the number of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source/drain terminals [Radhakrishnan85]. This is in contrast to logic families that we have studied so far that only allow primary inputs to drive the gate terminals of MOSFETS. Figure 6.29 shows a transistor level implementation of the AND function constructed using NMOS transistors. In this gate, if the *B* input is high, the top transistor is turned on and copies the input *A* to the output *F*. When input *B* is low, the bottom pass transistor is turned on and passes a 0. The switch driven by $\overline{B}$ seems to be redundant at first glance. Its presence is essential to ensure that a low-impedance path exists to the supply rails under all circumstances, or, in this particular case, when *B* is low.

The potential advantage of pass transistor is that a fewer number of transistors are required to implement a given function. For example, the implementation of the AND gate in Figure 6.29b requires 4 transistors (including the inverter required to invert *B*) while a complementary CMOS implementation would require 6 transistors.

Pass transistor logic uses fewer devices and therefore has lower physical capacitance. Unfortunately, as we have discussed earlier, a NMOS device is effective at passing

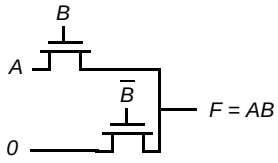


Figure 6.29 Pass-transistor implementation of an AND gate.

a 0 but is poor at pulling a node to V_{DD}. In pass transistor logic, the pass transistors are used to pass high and low voltages. Therefore, when the pass transistor pulls a node high, the output only charges up to V_{DD} -V_{Tn}. In fact, the situation is worsened by the fact that the devices experience body effect since there is a significant source to body voltage when pulling high since the body is tied to *GND* and the source charge up close to V_{DD}.

Consider the case when the pass transistor is charging up a node to V_{DD} where the gate and drain terminals are set at V_{DD}. Let the source the NMOS pass transistor be labeled *x*. Node *x* will charge up to V_{DD}-V_{Tn} where, the threshold must account for body effect as shown in Eq. (6.19). This maximum voltage swing on the output node is given by:

$$V_x = V_{DD} - (V_{tn0} + \gamma((\sqrt{|2\phi_f| + V_x}) - \sqrt{|2\phi_f|})) \tag{6.19}$$

Example 6.7 Voltage swing for pass transistors circuits

Assuming a power supply voltage of 2.5V, the transient response of Figure 6.30 shows the output of a NMOS charging up (where the drain voltage is at V_{DD} and the gate voltage in is ramped from 0V to V_{DD}). Assume that node *x* was initially 0. Also notice that if *IN* is low,

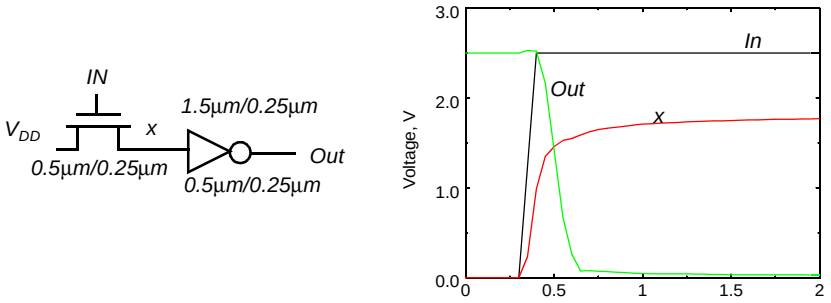


Figure 6.30 Transient response of charging up a node using an *N* device. Notice the slow tail after an initial quick response.

node *x* is in a high impedence state (not driven to one of the rails using a low resistance path). Extra transistors can be added to provide a path to GND, but for this discussion, the simplified circuit is sufficient. Notice that the ouput charges up quickly initially, but has slow tail. This is attributed to the fact that the drive (gate to source voltage) reduces significantly as the output approaches V_{DD}-V_{Tn} and the current available to charge up node *x* reduces drastically. Hand calculation using Eq. (6.19), results in an output voltage of 1.8V, which comes close to the simulated value.

WARNING:

The above example demonstrates that pass transistor gates cannot be cascaded by connecting the output of a pass gate to the gate terminal of another pass transistor. This is illustrated by the simple example of Figure 6.31. In Figure 6.31a, the output of M_1 (node *x*) drives the gate of another MOS device. Node *x* can charge up to V_{DD}-V_{Tn1}. If node *C* has a rail to rail swing, node *Y* only charges up to the voltage on node *x* - V_{Tn2} which works out to V_{DD}-V_{Tn1}-V_{Tn2}. Figure 6.31b on the other hand has the output of M_1 (*x*) driving the junc-

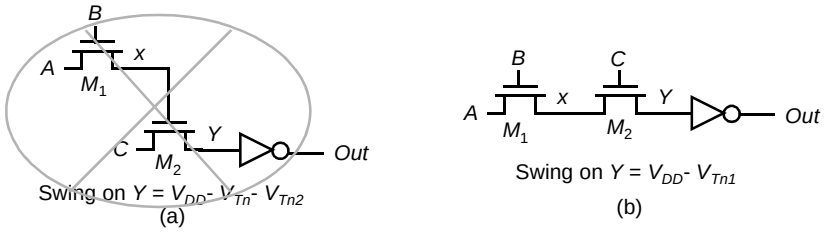


Figure 6.31 Pass transistor output (Drain/Source) terminal should not drive other gate terminals to avoid multiple threshold drops.

tion of M_2 and there is only one threshold drop. In pass transistor logic, the output of pass transistor devices should not drive the gate terminals of other pass transistors.

Example 6.8 VTC of the pass transistor AND gate

The voltage transfer curve of a pass-transistor gate shows little resemblance to complementary CMOS. Consider the AND gate shown in Figure 6.32. Similar to complementary CMOS, the VTC of pass transistor logic is data dependent. For the case when $B = V_{DD}$, the top pass transistor is turned *on* while the bottom one is turned *off*. In this case, the output just follows the input A until the input is high enough to turn *off* the top pass transistor (i.e., reaches V_{DD}-V_{Tn}). Next consider the case when A=V_{DD}, and B makes a transition from $0 \rightarrow 1$. Since the inverter has a threshold of $V_{DD}/2$, the bottom pass transistor is turned *on* till then and the output is close to zero. Once the bottom pass transistor turns *off*, the output follows the input B minus a threshold drop. A similar behavior is observed when both inputs A and B transition from $0 \rightarrow 1$. Note that pass transistor logic gates will need to be restored by placing inverters after every few pass transistors in series. With the inclusion of an inverter in the signal path, the VTC resembles the one of CMOS gates.

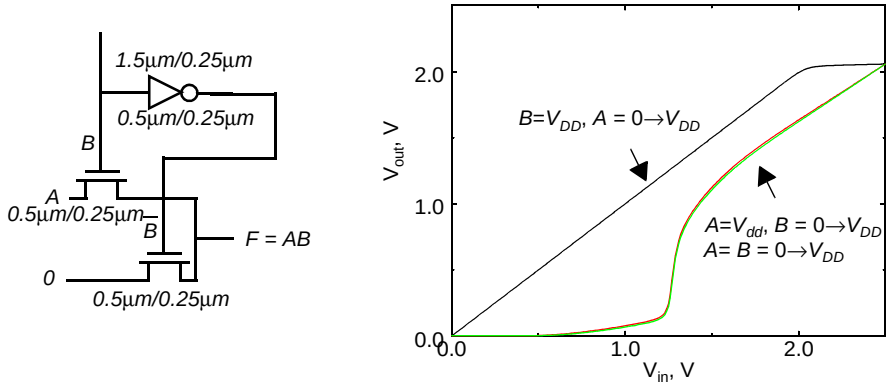


Figure 6.32 Voltage transfer curve for a pass transistor AND gate shown in Figure 6.29.

Pass transistors require lower switching energy to charge up a node due to its reduced voltage swing. For the pass transistor circuit in Figure 6.30 assume that the drain voltage is at V_{DD} and the gate voltage transitions to V_{DD}. The output node charges from 0V

to V_{DD}-V_{Tn} (assuming that node *x* was initially at 0V) and the energy drawn from the power supply for charging the output of a pass transistor is given by:

$$E_{0\rightarrow 1} = \int_0^T P(t)dt = V_{DD}\int_0^T i_{supply}(t)dt = V_{DD}\int_0^{(V_{DD}-V_{Tn})} C_L dV_{out} = C_L \bullet V_{DD} \bullet (V_{DD}-V_{Tn}) \quad (6.20)$$

While the circuit exhibits lower switching power, it consumes static power when the output is high since the PMOS device of the connecting inverter is not fully turned off.

Differential Pass Transistor Logic

For high performance design, a differential pass transistor logic family, called CPL or DPL, is commonly used. The basic idea (similar to DCVSL) is to accept true and complementary inputs and produce true and complementary outputs. A number of CPL gates (AND/NAND, OR/NOR, and XOR/NXOR) are shown in Figure 4.38. These gates possess a number of interesting properties:

- Since the circuits are *differential*, complementary data inputs and outputs are always available. Although generating the differential signals requires extra circuitry, the differential style has the advantage that some complex gates such as XORs and adders can be realized efficiently with a small number of transistors. Furthermore,

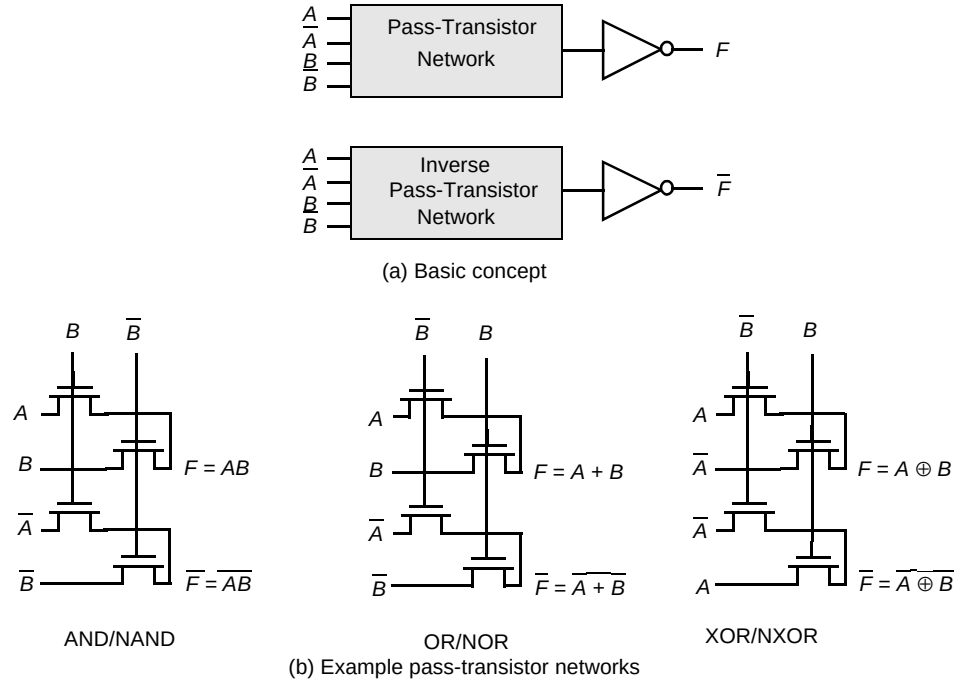


Figure 6.33 Complementary pass-transistor logic (CPL).

the availability of both polarities of every signal eliminates the need for extra inverters, as is often the case in static CMOS or pseudo-NMOS.

- CPL belongs to the class of *static* gates, because the output-defining nodes are always connected to either V_{DD} or *GND* through a low resistance path. This is advantageous for the noise resilience.
- The design is very modular. In effect, all gates use exactly the same topology. Only the inputs are permutated. This makes the design of a library of gates very simple. More complex gates can be built by cascading the standard pass-transistor modules.

Example 6.9 Four-input NAND in CPL

Consider the implementation of a four-input AND/NAND gate using CPL. Based on the associativity of the boolean AND operation [$A{\cdot}B{\cdot}C{\cdot}D = (A{\cdot}B){\cdot}(C{\cdot}D)$], a two-stage approach has been adopted to implement the gate (Figure 6.34). The total number of transistors in the gate

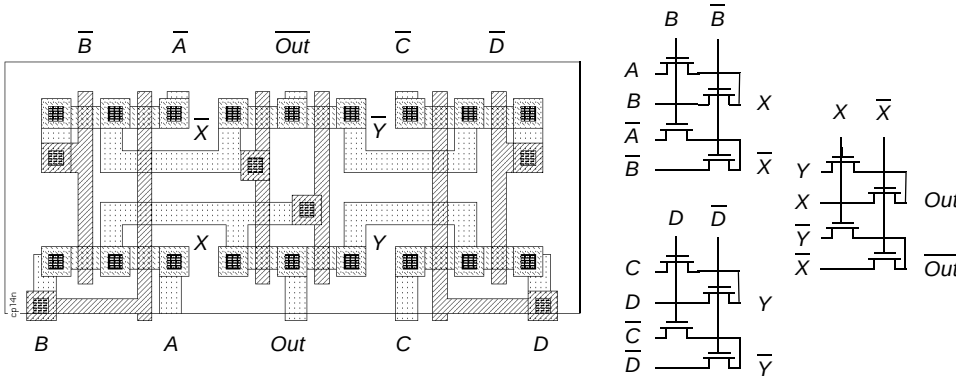


Figure 6.34 Layout and schematics of four-input NAND-gate using CPL (the final inverter stage is omitted). See also Colorplate 9.

(including the final buffer) is 14. This is substantially higher than previously discussed gates. This factor, combined with the complicated routing requirements, makes this circuit style not particularly efficient for this gate. One should, however, be aware of the fact that the structure simultaneously implements the AND and the NAND functions, which might reduce the transistor count of the overall circuit.

In summary, CPL is a conceptually simple and modular logic style. Its applicability depends strongly upon the logic function to be implemented. The availability of a simple XOR as well of the ease of implementing some specific gate structures makes it attractive for structures such as adders and multipliers. Some extremely fast and efficient implementations have been reported in that application domain [Yano90]. When considering CPL, the designer should not ignore the implicit routing overhead of the complementary signals, which is apparent in the layout of Figure 6.34.

Robust and Efficient Pass-Transistor Design

Unfortunately, differential pass transistor logic, like single-ended pass transistor logic suffers from static power dissipation since the high input to the inverter only charges up to V_{DD}-V_{Tn}. Static power is highly undesirable since in many portable electronics, the devices are idle for extended periods of time. Therefore, the voltage drop of pass transistors that causes lower noise margins and static power is not acceptable. There are several solutions proposed to deal with this problem as outlined below.

Solution 1: Level Restoration

A common solution to the voltage drop of pass transistors is the use of a *level restorer*, which is a single PMOS configured in a feedback path (Figure 6.35). The gate of the PMOS device is connected to the output of the inverter, its drain connected to the input of the inverter and the source to V_{DD}. Assume that node X is at 0V (*out* is at V_{DD} and the M_r is turned *off*) with $B = V_{DD}$ and $A = 0$. If input A makes a 0 to V_{DD} transition, M_n only charges up node X to V_{DD}-V_{Tn}. This is, however, enough to switch the output of the inverter low, turning on the feedback device M_r and pulling node X all the way to V_{DD}. This eliminates any static power dissipation in the inverter. Furthermore, no static current path can exist through the level restorer and the pass-transistor, since the restorer is only active when A is high. In summary, this circuit has the advantage that all voltage levels are either at *GND* or V_{DD}, and no static power is consumed.

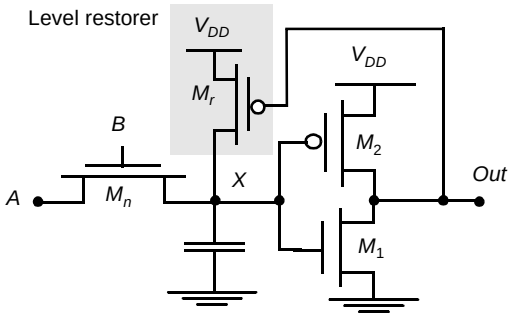


Figure 6.35 Level-restoring circuit.

While this solution is appealing in terms of eliminating static power dissipation, it is more complex since the circuit is now ratioed. The problem arises during the transition of node X from high-to-low. The pass transistor network attempts to pull-down node X while the level restorer pulls now X to V_{DD}. Therefore, the pull-down device must be stronger than the pull-up device to switch node X and the output. We use the notation R_1 to denote the equivalent on-resistance of transistor M_1, R_2 for M_2, and so on. Some careful transistor sizing is necessary to make the circuit function correctly: when R_r is made too small, it is impossible to bring the voltage at node X below the switching threshold of the inverter. Hence, the inverter output never switches to V_{DD}, and the level-restoring transistor stays on. This sizing problem can be reformulated in the following way:

The resistance of M_n and M_r must be such that the voltage at node X drops below the threshold of the inverter, $V_M = f(R_1, R_2)$. This condition is sufficient to guarantee a switching of the output voltage V_{out} to V_{DD} and a turning off of the level-restoring transistor.

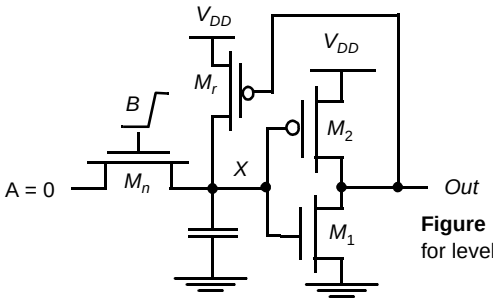


Figure 6.36 Transistor-sizing problem for level-restoring circuit.

Example 6.10 Sizing of a Level Restorer

Analyzing the circuit as a whole is nontrivial, because the restoring transistor acts as a feedback device. One way to simplify the circuit for manual analysis is to open the feedback loop and to ground the gate of the restoring transistor when determining the switching point (this is a reasonable assumption, as the feedback only becomes effective once the inverter starts to switch). Hence, M_r and M_n form a "pseudo-NMOS-like" configuration, with M_r the load transistor and M_n acting as a pull-down device to *GND*. Assume that the inverter M_1, M_2 is sized to have the switching point at $V_{DD}/2$ (NMOS: 0.5μm/0.25μm and PMOS: 1.5μm/0.25μm). Therefore, node *X* must be pulled below $V_{DD}/2$ in order to switch the inverter and shut off M_r.

This is confirmed in Figure 6.38, which shows the transient response as the size of the level restorer is varied while keeping the size of M_n fixed (0.5μm/0.25μm). As the simulation indicates, for sizes above 1.5μm/0.25μm, node *X* can't be brought below the switching threshold of the inverter and can't switch the output. The detailed derivation of sizing requirement will be presented in the sequential design chapter. An important point to observe here is that the sizing of M_r is critical for DC functionality, not just performance!

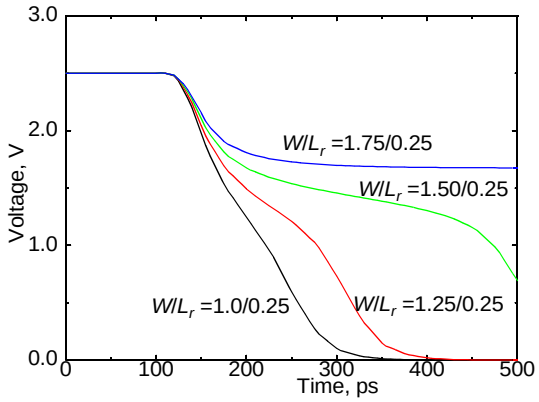


Figure 6.37 Transient response of the circuit in Figure 6.36. A level restorer that is too large can result in incorrect evaluation.

Another concern is the influence of the level restorer on the switching speed of the device. Adding the restoring device increases the capacitance at the internal node *X*, slowing down the gate. The rise time of the gate is further negatively affected, since, the level-restoring transistor M_r fights the decrease in voltage at node *X* before being switched off.

On the other hand, the level restorer reduces the fall time, since the PMOS transistor, once turned on, speeds the pull-up action.

Problem 6.5 Device Sizing in Pass Transistors

For the circuit shown in Figure 6.36, assume that the pull-down device consists of 6 pass transistors in series each with a device size of 0.5μm/0.25μm (replacing transistor M_n). Determine the maximum W/L size for the level restorer transistor for correct functionality.

A modification of the level restorer to differential pass transistors is shown in Figure 6.38, know as swing restored pass transistor logic. Instead of a simple inverter or half latch at the output of the pass transistor network, two back-to-back inverters configured in a cross coupled fashion are used for level restoration and performance improvement. Inputs are fed to both the gate and source/drain terminals as in the case of conventional pass transistor networks. Figure 6.38 shows a simple XOR/XNOR gate of three variables *A*, *B* and *C*. Notice that the complementary network can be optimized by sharing transistors between the true and complementary outputs.

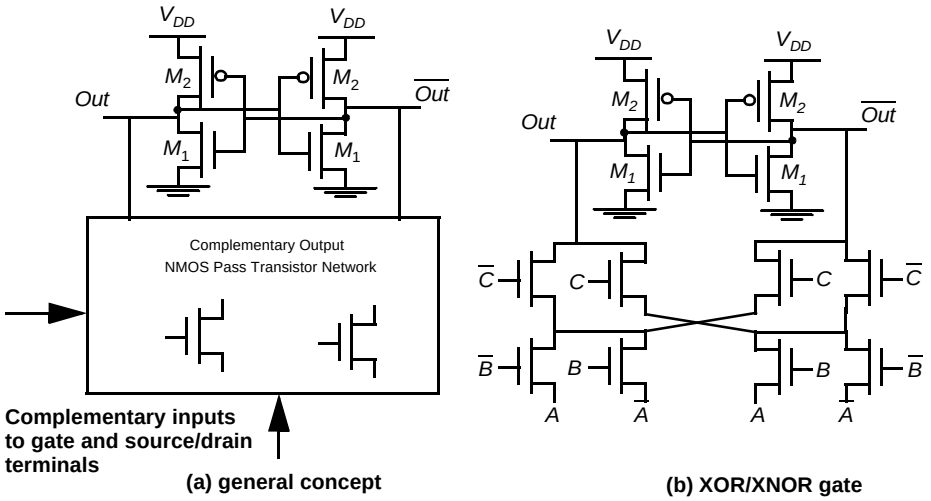


Figure 6.38 Swing restored pass transistor logic [Parameswar94].

Solution 2: Multiple Threshold Transistors

A technology solution to the voltage drop problem associated with pass transistor logic is the use of multiple threshold devices. Pass transistors only pass V_{DD}-V_{Tn}, degrading the high voltage level. One solution is to use *zero threshold* devices for the NMOS pass transistors, enabling passing signal close to V_{DD}. Notice that even if the devices threshold was implanted to be exactly equal to zero, the body effect of the device prevents a swing to V_{DD}. All devices other than the pass transistors (i.e., the inverters) are implemented using standard high threshold devices. The use of multiple threshold transistors is becoming more common and involves simple modifications to existing process flows.

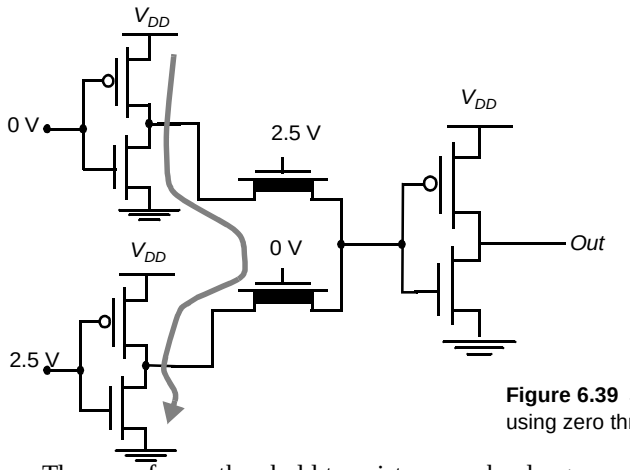
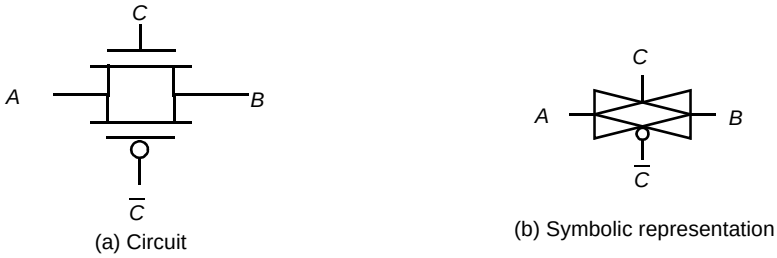


Figure 6.39 Static power consumption when using zero threshold pass-transistors.

The use of zero-threshold transistors can be dangerous due to the subthreshold currents that can flow through the pass-transistors, even if V_{GS} is slightly below V_T. This is demonstrated in Figure 6.39, which points out a potential sneak dc-current path. While these leakage paths are not critical when the device is switching constantly, they do pose a significant energy overhead when the device is in the idle state.

Solution 3: Transmission Gate Logic

The most widely used solution to deal with the voltage drops induced by pass transistors is the use of *transmission gates*. The primary limitation of NMOS or PMOS only pass gate is the threshold drop (NMOS pass device pass a strong 0 while passing a weak 1 and PMOS pass devices pass a strong 1 while passing a weak 0). The ideal approach is to use the NMOS device to pull-down and the PMOS device to pull-up. The transmission gate combines the best of both device flavors by placing a NMOS device in parallel with a PMOS device (Figure 6.40a). The control signals to the transmission gate (C and $\overline{C}$)



(a) Circuit (b) Symbolic representation

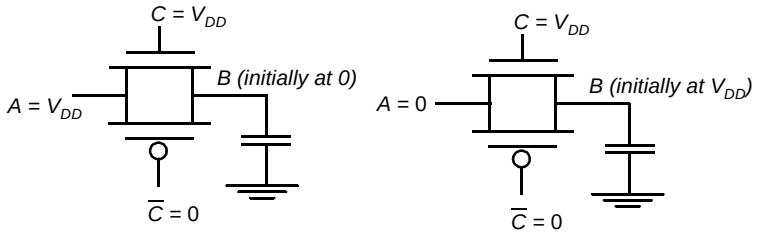
Figure 6.40 CMOS transmission gate.

are complementary. The transmission gate acts as a bidirectional switch controlled by the gate signal C. When $C = 1$, both MOSFETs are on, allowing the signal to pass through the gate. In short,

$$A = B \quad \text{if} \quad C = 1 \tag{6.21}$$

On the other hand, $C = 0$ places both transistors in cutoff, creating an open circuit between nodes *A* and *B*. Figure 6.40b shows a commonly used transmission-gate symbol.

Consider the case of charging node *B* to V_{DD} for the transmission gate circuit in Figure 6.41a. Node *A* is driven to V_{DD} and transmission gate is enabled ($C = 1$ and $\overline{C} = 0$). If only the NMOS pass device were present, node *B* will charge up to V_{DD}-V_{Tn} at which point the NMOS device turns off. However, since the PMOS device is present and turned on ($V_{GSp} = -V_{DD}$), node *B* charge all the way up to V_{DD}. Figure 6.41b shows the case for discharging node *B* to 0. *B* is initially at V_{DD} and node *A* is driven low. The PMOS pass transistor by itself can only pull-down node *B* to V_{Tp} at which point the PMOS device turns *off*. the parallel NMOS device however is turned on (since its $V_{GS} = V_{DD}$) and pulls down node *B* all the way to *GND*. Though the transmission gate requires two transistors and more control signals, it enables rail-to-rail swing.



(a) charing node B (a) discharing node B

Figure 6.41 Transmission gates enable rail-to-rail switching

Transmission gates can be used to build some complex gates very efficiently. Figure 6.42 shows an example of a simple inverting two-input multiplexer. This gate either selects input *A* or *B* based on the value of the control signal *S*, which is equivalent to implementing the following Boolean function:

$$\overline{F} = (A \cdot S + B \cdot \bar{S}) \tag{6.22}$$

A complementary implementation of the gate requires eight transistors instead of six.

Another example of the effective use of transmission gates is the popular XOR circuit shown in Figure 6.43. The complete implementation of this gate requires only six transistors (including the inverter used for the generation of $\overline{B}$), compared to the twelve transistors required for a complementary implementation. To understand the operation of this circuit, we have to analyze the $B = 0$ and $B = 1$ cases separately. For $B = 1$, transistors M_1 and M_2 act as an inverter while the transmission gate M_3/M_4 is off; hence $F = \overline{A}B$. In the opposite case, M_1 and M_2 are disabled, and the transmission gate is operational, or $F = A\overline{B}$. The combination of both results in the XOR function. Notice that, regardless of the values of *A* and *B*, node *F* always has a connection to either V_{DD} or *GND* and is hence a low-impedance node. When designing static-pass transistor networks, it is essential to adhere to the low-impedance rule under all circumstances. Other examples where transmission-gate logic is effectively used are fast adder circuits and registers.

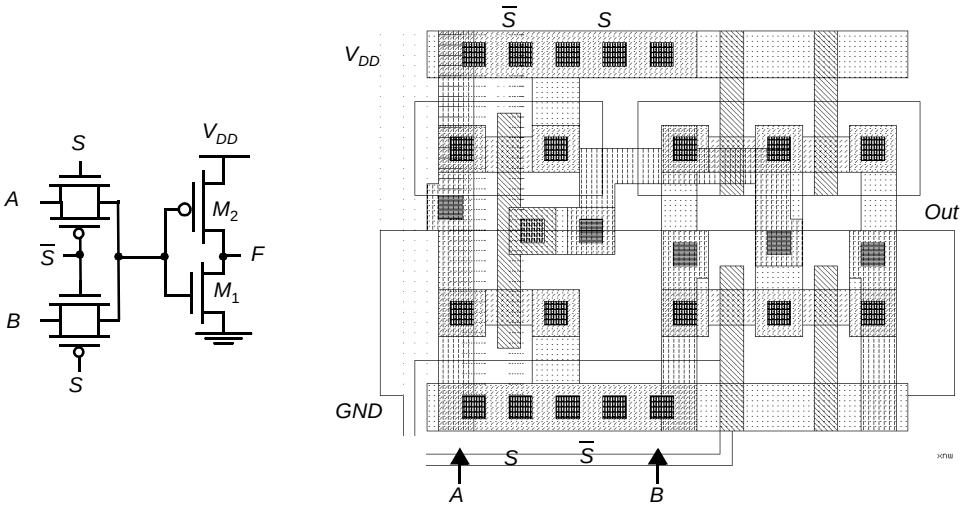


Figure 6.42 Transmission gate multiplexer and its layout.

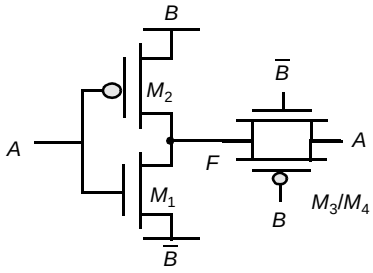


Figure 6.43 Transmission gate XOR.

Resistance and Delay of Transmission Gate Logic

The transmission gate is, unfortunately, not an ideal switch, and has a series resistance associated with it. To quantify the resistance, consider the circuit in Figure 6.44, which involves charging a node from 0 V to V_{DD}. In this discussion, we will us the large signal definition of resistance which involves dividing the voltage across the switch by the drain current. The effective resistance of the switch is modeled as a parallel connection of the resistances R_n and R_p of the NMOS and PMOS devices, defined as $(V_{DD} - V_{out})/I_n$ and $(V_{DD} - V_{out})/I_p$, respectively. The currents through the devices are obviously dependent on the value of V_{out} and the operating mode of the transistors. During the low-to-high transition, the pass-transistors traverse through a number of operation modes. Figure 6.44 shows the individual resistances and the combined parallel resistance. For low values of V_{out}, the the NMOS device is saturated and the resistance is approximated as:

$$R_n = \frac{V_{DD}-V_{out}}{I_N} = \frac{V_{DD}-V_{out}}{k'_n\left(\frac{W}{L}\right)_N\left((V_{DD}-V_{out}-V_{Tn})V_{DSAT}-\frac{V_{DSAT}^2}{2}\right)} \approx \frac{V_{DD}-V_{out}}{k_n(V_{DD}-V_{out}-V_{Tn})V_{DSAT}} \tag{6.23}$$

The resistance goes up for increasing values of V_{out}, and approaches infinity when V_{out} reaches V_{DD}-V_{Tn}, this is when the device shuts off. Similarly, we can analyze the behavior of the PMOS transistor. When V_{out} is small, the PMOS is saturated, but it enters the linear mode of operation for V_{out} approaching V_{DD}. The resistance then approximated by:

$$R_p = \frac{V_{DD}-V_{out}}{I_P} = \frac{V_{DD}-V_{out}}{k_p\cdot\left((-V_{DD}-V_{Tp})(V_{out}-V_{DD})-\frac{(V_{out}-V_{DD})^2}{2}\right)} \approx \frac{1}{k_p(V_{DD}-|V_{Tp}|)} \tag{6.24}$$

The simulated value of $R_{eq} = R_p \parallel R_n$ as a function of V_{out} is plotted in Figure 6.44. It can be observed that R_{eq} is relatively constant (≈ 8kΩ in this particular case). The same is true in other design instances (for instance, when discharging C_L). When analyzing transmission-gate networks, the simplifying assumption that the switch has a constant resistive value is therefore acceptable.

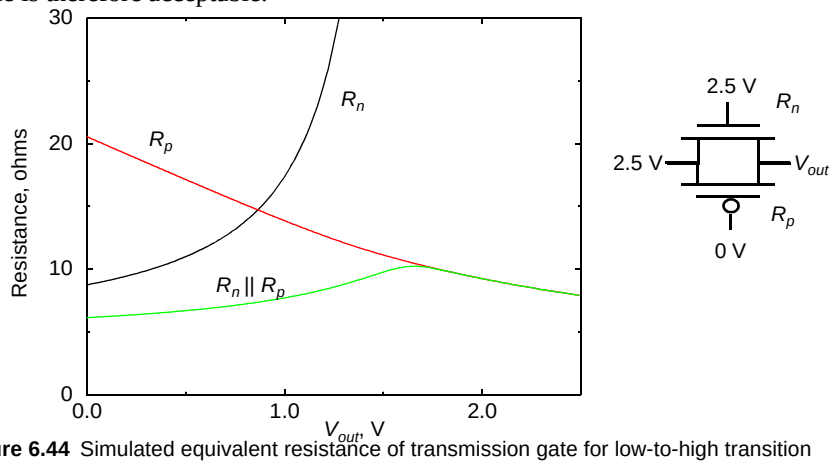
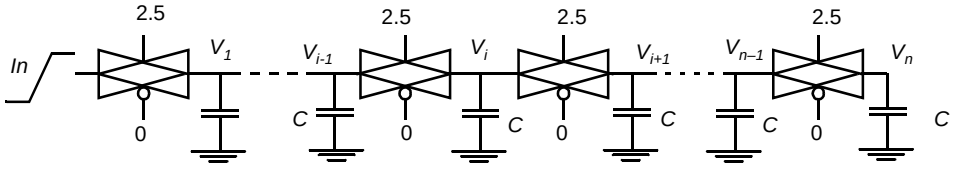


Figure 6.44 Simulated equivalent resistance of transmission gate for low-to-high transition (for $(W/L)_n = (W/L)_p = 0.5\mu m/0.25\mu m$). A similar response for overall resistance is obtained for the high-to-low transition

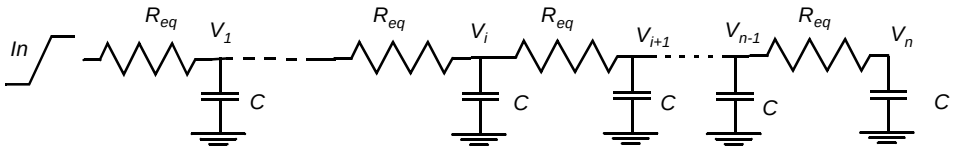
Problem 6.6 Equivalent Resistance During Discharge

Determine the equivalent resistance by simulation for the high-to-low transition of a transmission gate (this is, produce a plot similar to the one presented in Figure 6.44).

An important consideration is the delay associated with a chain of transmission gates. Figure 6.45 shows a chain of *n* transmission gates. Such a configuration often occurs in circuits such as adders or deep multiplexors. Assume that all transmission gates are turned on and a step is applied at the input. To analyze the propagation delay of this network, the transmission gates are replaced by their equivalent resistances R_{eq}. This produces the network of Figure 6.45b.



(a) A chain of transmission gates



(b) Equivalent *RC* network

Figure 6.45 Speed optimization in transmission-gate networks.

The exact analysis of delay is not simple, but as disussed earlier, we can estimate the dominant *time constant* at the output of a chain of *n* transmission gates as follows:

$$\tau(V_n) = \sum_{k=0}^{n} CR_{eq}k = CR_{eq}\frac{n(n+1)}{2} \tag{6.25}$$

This means that the propagation delay is proportional to n^2 and increases rapidly with the number of switches in the chain.

Example 6.11 Delay through 16 transmission gates

Consider 16 minimimum sized transmission gates with an average resistance of 8 kΩ. The node capacitance consists of the capacitance of two NMOS devices (junction and gate) and the capacitance two PMOS devices (junctions and gate). Since the gate inputs are assumed to be fixed, there is no miller multiplication. The capacitance can be calculated to be approximately 3.6fF for the low-to-high transition. The delay is given by:

$$t_p = 0.69 \cdot CR_{eq}\frac{n(n+1)}{2} = 0.69 \cdot (3.6fF)(8K\Omega)\left(\frac{16(16+1)}{2}\right) \approx 2.7ns \tag{6.26}$$

The transient response for this particular example is shown in Figure 6.46. The simulated delay is 2.7ns. It is remarkable that a simple RC model predicts the delay accu-

rately. It is also clear that the use of long pass transistor chains causes significant delay degradation.

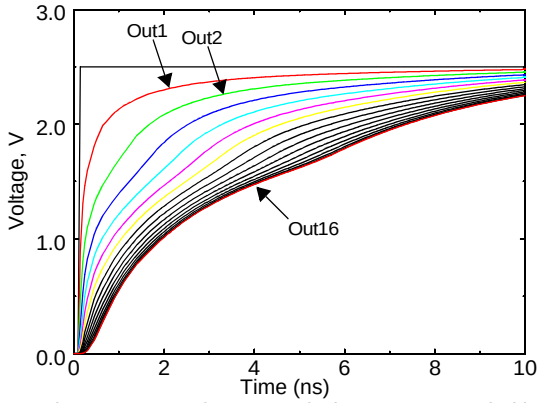


Figure 6.46 Transient response of 16 transmission gates cascaded in series.

The most common approach for delaing with the long delay is to break the chain every m switches and to insert buffers (Figure 6.46). Assuming a propagation delay t_{buf} for each buffer, the overall propagation delay of the transmission-gate/buffer network can be computed as follows,

$$\begin{aligned} t_p &= 0.69\left[\frac{n}{m}CR_{eq}\frac{m(m+1)}{2}\right] + \left(\frac{n}{m}-1\right)t_{buf} \\ &= 0.69\left[CR_{eq}\frac{n(m+1)}{2}\right] + \left(\frac{n}{m}-1\right)t_{buf} \end{aligned} \tag{6.27}$$

The resulting delay exhibits only a linear dependence on the number of switches n, in contrast to the unbuffered circuit, which is quadratic in n. The optimal number of switches m_{opt} between buffers can be found by setting the derivative

$$\frac{\partial t_p}{\partial m}$$

to 0, which yields

$$m_{opt} = 1.7\sqrt{\frac{t_{pbuf}}{CR_{eq}}} \tag{6.28}$$

Obviously, the number of switches per segment grows with increasing values of t_{buf}. In current technologies, m_{opt} typically around 3.

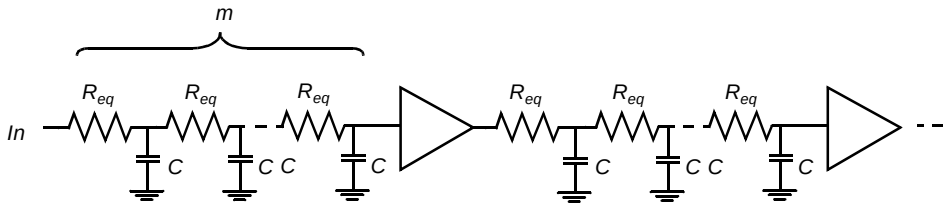


Figure 6.47 Breaking up long transmission gate chains by inserting buffers.

Example 6.12 Transmission Gate Chain

Consider the same 16 transmission gate chain. The buffers shown in Figure 6.47 can be implemented as inverters (instead of two cascaded inverters). In some cases, it might be necessay to add an extra inverter to produce the correct polarity. Assuming that each inverter is sized such that the NMOS is 0.5µm/0.25µm and PMOS is 0.5µm /0.25µm, Eq. (6.28) predicts that an inverter must be inserted every 3 transmission gates. The simulated delay when placing an inverter every two transmission gates equals 154ps, for every three transmission gates is 154ps and for four transmission gates is 164ps. The insertion of buffering inverters reduces the delay with a factor of almost 2.

CAUTION: Although many of the circuit styles discussed in the previous sections sound very exciting, and might be superior to static CMOS in many respects, none of them has the *robustness and ease of design* of complementary CMOS. Therefore, use them sparingly and with caution. For designs that have no extreme area, complexity, or speed constraints, complementary CMOS is the recommended design style.

6.3 Dynamic CMOS Design

It was noted earlier that static CMOS logic with a fan-in of N requires $2N$ devices. A variety of approaches were presented to reduce the number of transistors required to implement a given logic function including pseudo-NMOS, pass transistor logic, etc. The pseudo-NMOS logic style requires only $N + 1$ transistors to implement an N input logic gate, but unfortunately it has static power dissipation. In this section, an alternate logic style called *dynamic logic* is presented that obtains a similar result, while avoiding static power consumption. With the addition of a clock input, it uses a sequence of *precharge* and conditional *evaluation* phases to realize complex logic functions.

6.3.1 Dynamic Logic: Basic Principles

The basic construction of a N-type dynamic logic gate is shown in Figure 6.48a. The PDN (pull-down network) is constructed exactly in the same fashion as a complementary CMOS. The operation of this circuit can be divided into two major phases: *precharge* and *evaluation*, with the mode of operation determined by the *clock signal*.

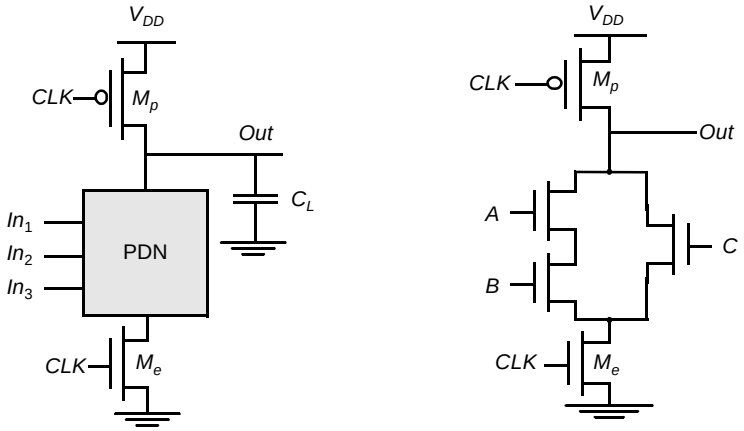


Figure 6.48 Basic concepts of a dynamic gate.

Precharge

When *CLK* = 0, the output node *Out* is precharged to V_{DD} by the PMOS transistor M_p. During that time, the evaluate NMOS transistor M_e is off, so the pull-down path does not fight the pull-up path. The evaluation FETS also eliminate any static power that would be consumed during the precharge period (i.e., if the pull-down path was turned *on* and the precharge device was turned *on*, static current would flow between the supplies).

Evaluation

When *CLK* = 1, the precharge transistor M_p is off, and the evaluation transistor M_e is turned on. The output is conditionally discharged based on the input values and the pull-down topology. If the inputs are such that the PDN conducts, then a low resistance path exists between *Out* and *GND* and the output is discharged to *GND*. If the PDN is turned off, the precharged value remains stored on the output capacitance C_L, which is a combination of junction capacitances, the wiring capacitance, and the input capacitance of the fan-out gates. During the evaluation phase, the only possible path between the output node and a supply rail is to *GND*. Consequently, once *Out* is discharged, it cannot be charged again till then next precharge operation. The inputs to the gate can therefore make *at most one transition during evaluation*. Notice that the output can be in the high impedance state during the evaluation period if the pull-down network is turned off and this behavior is fundamentally different than the static counterpart that always has a low resistance path between the output and one of the power rails.

As as an example of dynamic logic, consider the circuit shown in Figure 6.48b. During the precharge phase (*CLK*=0), the output is precharged to V_{DD} regardless of the input values since the evaluation device is turned off. During evaluation (*CLK*=1), a conducting path is created between *Out* and *GND* if (and only if) $A \cdot B + C$ is TRUE. Otherwise, the output remains at the precharged state of V_{DD}. The following function is thus realized:

$$Out = \overline{A \cdot B + C} \quad \text{(when } CLK = 1\text{)} \tag{6.29}$$

A number of important properties can be derived for the dynamic logic gate:

- The logic function is implemented by the NMOS pull-down network. The construction of the PDN proceeds just as it does for static CMOS.
- The *number of transistors* (for complex gates) is substantially lower than in the static case: $N + 2$ versus $2N$.
- It is *nonratioed*. The sizing of the PMOS precharge device is not important for realizing proper functionality of the gate. The size of the precharge device can be made large to improve the low-to-high transition time (of course, at a cost to the high-to-low transition time). There is however, a trade-off with power dissipation since a larger precharge device directly increases clock power dissipation.
- It only consumes *dynamic power*. Ideally, no static current path ever exists between V_{DD} and *GND*. The overall power dissipation, however, can be significantly higher compared to a static logic gate.
- The logic gates have *faster switching speeds.* There are two main reasons for this. The first (obvious) reason is due to the reduced load capacitance attributed to the number of transistors per gate and the single-transistor load per *fan-in*. Second, the dynamic gate do not have short circuit current, and all the current provided by the pull-down devices go into discharging the load capacitance.

The low and high output levels V_{OL} and V_{OH} are easily identified as *GND* and V_{DD} and are not dependent upon the transistor sizes. The other VTC parameters are dramatically different from static gates. Noise margins and switching thresholds have been defined as static quantities, which are not influenced by time. To be functional, a dynamic gate requires a periodic sequence of precharges and refreshes. Pure static analysis, therefore, does not apply. During the evaluate period, the pull-down network of a dynamic inverter starts to conduct when the input signal exceeds the threshold voltage (V_{Tn}) of the NMOS pull-down transistor. Therefore, it is reasonable to set the switching threshold (V_M) as well as V_{IH} and V_{IL} of the gate equal to V_{Tn}. This translates to a low value for the NM_L.

6.3.2 Speed and Power Dissipation of Dynamic Logic

The main advantage of dynamic logic is speed and potentially smaller implementation area. Fewer devices to implement a given logic function implies that the overall load capacitance is much smaller. The analysis of the switching behavior of the gate has some interesting peculiarities to it. After the precharge phase, the output is high. For a low input signal, no additional switching occurs. As a result, $t_{pLH} = 0$! The high-to-low transition, on

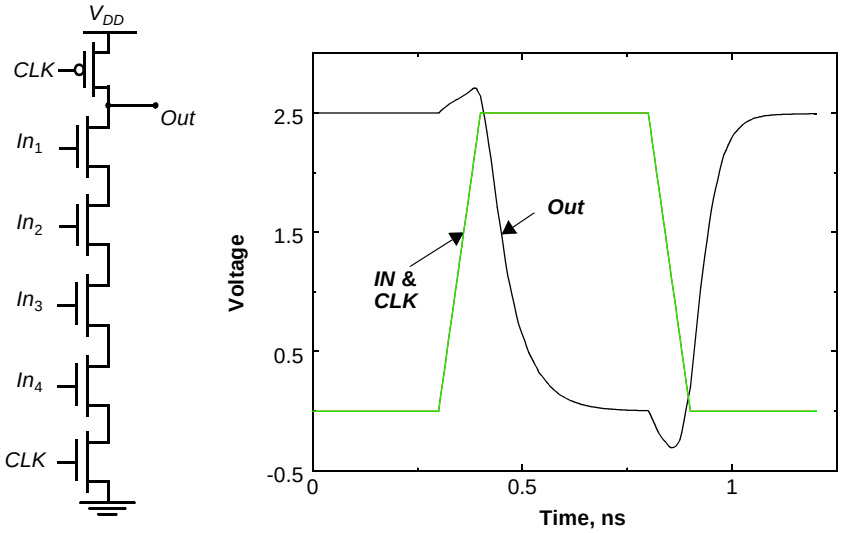


Figure 6.49 Schematic of a dynamic four-input NAND gate.

the other hand, requires the discharging of the output capacitance through the pull-down network. Therefore t_{pHL} is proportional to C_L and the current-sinking capabilities of the PDN. The presence of the evaluation transistor slows the gate somewhat, as it presents an extra series resistance to the pull-down network. Omitting this transistor, while functionally not forbidden, results in significant performance loss and static power dissipation.

The above analysis is somewhat unfair, because it ignores the influence of the precharge time on the switching speed of the gate. The precharge time is determined by the time it takes to charge C_L through the PMOS precharge transistor. During this time, the logic in the gate cannot be utilized. However, very often, the overall digital system can be designed in such a way that the precharge time coincides with other system functions. For instance, the precharge of the arithmetic unit in a microprocessor can coincide with the instruction decode. The designer has to be aware of this "dead zone" in the use of dynamic logic, and should carefully consider the pros and cons of its usage, taking the overall system requirements into account.

Example 6.13 A Four-Input Dynamic NAND Gate

Figure 6.49 shows the design of a four-input NAND example designed using the dynamic-circuit style. Due to the dynamic nature of the gate, the derivation of the voltage-transfer characteristic diverges from the traditional approach. As we had discussed above, we will assume that the switching threshold of the gate equals the threshold of the NMOS pull-down transistor. This results in asymmetrical noise margins, as shown in Table 6.7.

The dynamic behavior of the gate is simulated with SPICE. It is assumed that all inputs are set high as the clock transitions high. On the rising edge of the clock, the output node is discharged. The resulting transient response is plotted in Figure 4.35. The resulting propagation delays are summarized in Table 6.7. The length of the precharge time can be adjusted by changing the size of the PMOS precharge transistor. Making the PMOS too large should be avoided, however, as it both slows down the gate and increases the capacitive load on the clock line. For large designs, the latter factor might become a major design concern because the clock load can become excessive and hard to drive.

Table 6.7 The dc and ac parameters of a four-input dynamic NAND.

Transistors	V_{OH}	V_{OL}	V_M	NM_H	NM_L	t_{pHL}	t_{pLH}	t_{pre}
6	2.5 V	0 V	V_{TN}	2.5-V_{TN}	V_{TN}	110 ps	0 nsec	83pS

As mentioned earlier, the static parameters are time dependent. To illustrate this, consider the four input NAND gate with all inputs tied together. Assume that the inputs make a partial low-to-high transition. Figure 6.50 shows a simulation of the output voltage for three different input voltages (when input transitions to 0.45V, 0.5V and 0.55V). We have previously defined the switching threshold of the dynamic gate as the device threshold. However, notice that the amount by which the output voltage drops is a strong function of the input voltage and the available evaluation time. In this example, a larger input voltage is necessary to corrupt the output. So the switching threshold is really a function of the evaluation time.

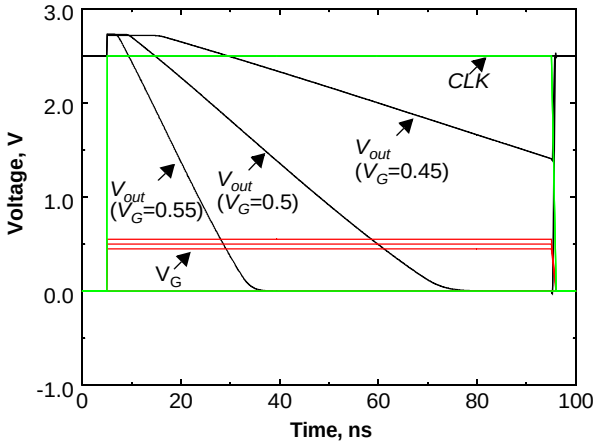


Figure 6.50 Effect of an input glitch on the output. The switching threshold depends on the time for evaluation. A larger glitch is acceptable if the evaluation phase is smaller. In this example, the input glitches high during evaluation and stays high during the whole period.

When evaluating the power dissipation of a dynamic gate, it would appear that dynamic logic presents significant advantage. There are three reasons for this. First, since dynamic logic uses fewer transistors to implement a given function, it should have a lower physical capacitance. Basically the load seen for each fanout is one transistor instead of two. Second, dynamic logic gates *by construction* can at most have one transition per clock cycle. The glitching transitions seen in static gates are not seen in dynamic gates. Finally, dynamic gates do not exhibit short circuit power since the pull-up path is not turned on when the gate is evaluating. While the above statements are generally true, several important second order effects causes the number of transistors to be higher than the minimal set required for implementing the logic and short circuit power does exist if the logic is required to be pseudo-static. Also, the clock power of dynamic logic can be significant particularly since the clock node has a guaranteed transition on every single clock cycle.

Dynamic logic generally has higher activity due to constant *precharge* and *discharge* operations. Earlier, the transition probability for a static gate was shown to be $p_0\, p_1 = p_0\,(1\text{-}p_0)$. For dynamic logic, the output transition probability does not depend on the state (history) of the inputs but rather on just the signal probabilities. For an N-tree dynamic gate, the output will make a 0 to 1 transition during the precharge phase only if the output was discharged by the N-tree logic during the evaluate phase. The zero to one transition probability for an N-tree structure is therefore

$$a_{0\rightarrow 1} = p_0 \tag{6.30}$$

where p_0 is the probability that the output is in the zero state. For uniformly distributed inputs, this means that the transition probability is:

$$a_{0\rightarrow 1} = \frac{N_0}{2^N} \tag{6.31}$$

where N_0 is the number of zero entries in the truth table of the logic function.

Example 6.14Activity estimation in dynamic logic

To illustrate the increased activity for a dynamic gate, once again consider a 2 input NOR gate. An N-tree dynamic NOR gate is shown in Figure 6.51 along with its static counterpart. For the dynamic implementation, power is consumed during the precharge operation for the times when the output capacitor was discharged the previous cycle. For equi-probable input, there is then a 75% probability that the output node will discharge immediately after the precharge phase, implying that the activity for such a gate is 0.75 (i.e $P_{NOR}= 0.75\ C_L V_{dd}^2 f_{clk}$). The corresponding activity is a lot smaller, 3/16, for a static implementation. Note that for the dynamic case, the activity depends only on the signal probability, while for the static case the transition probability depends on previous state. If the inputs to a static CMOS gate do not change from the previous sample period, then the gate does not switch. This is not true in the case of dynamic logic in which gates can switch. For a dynamic NAND gate, the transition probability is 1/4 (since there is a 25% probability the output will be discharged) while it is 3/16 for a static implementation. Though this example illustrates that the switching activity can be higher using dynamic

logic, it should be noted that dynamic logic has lower physical capacitance. Both factors must be accounted for when choosing a logic style.

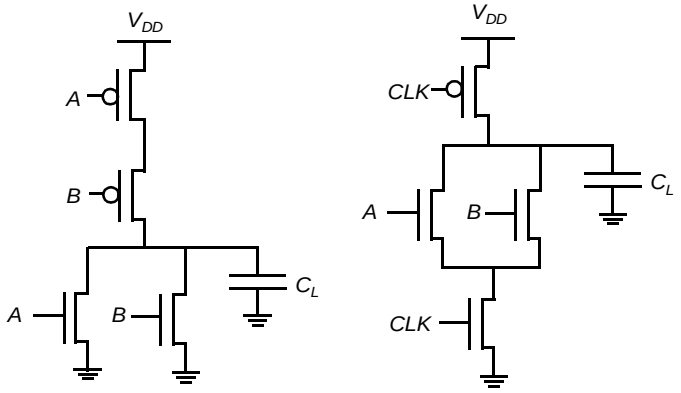


Figure 6.51 Static NOR vs. N-tree based dynamic NOR.

Problem 6.7 Activity Computation

For the 4-input dynamic NAND gate, compute the activity factor with the following assumption for the inputs. Assume that the inputs are independent and $p_{A=1} = 0.2, p_{B=1} = 0.3, p_{C=1} = 0.5$, and $p_{D=1} = 0.4$.

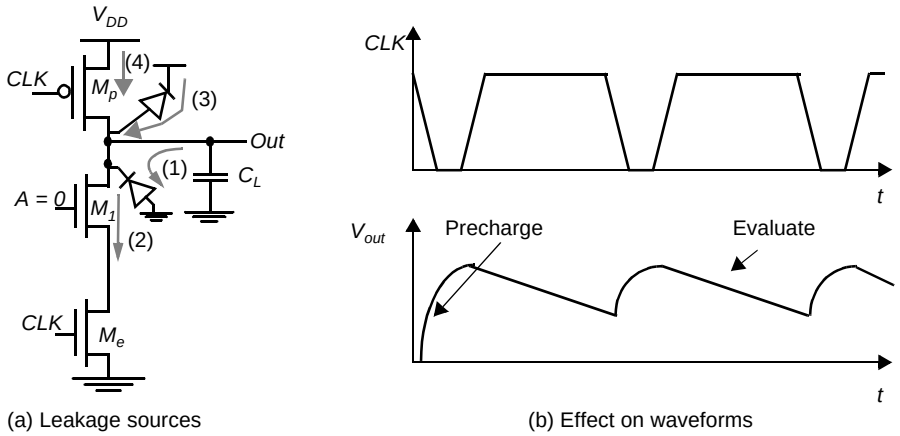
6.3.3 Issues in Dynamic Design

Dynamic logic clearly can result in high performance solutions compared to static circuits. However, there are several important considerations that must be taken into account to make dynamic circuits function properly. This include charge leakage, charge sharing, backgate (and in general capacitive) coupling, and clock feedthrough. Some of these issues are highlighted in this section.

Charge Leakage

The operation of a dynamic gate relies on the dynamic storage of the output value on a capacitor. During the evaluation period, if the pull-down network is *off*, then ideally the output should remain at the precharged state of V_{DD}. However, due to leakage currents, this charge gradually leaks away, resulting eventually in malfunctioning of the gate. Figure 6.52a shows the different sources of leakage for a simple dynamic inverter circuit.

Source 1 and 2 are the *reverse-biased diode* and *sub-threshold leakage* of the NMOS pull-down device M_1 respectively. The charge stored on C_L will slowly leak away due these leakage sources, assuming that the input is in the low state during evaluation. Charge leakage causes a degradation in the high level (Figure 6.52b). Dynamic circuits therefore require a minimal clock rate, which is typically on the order of a few kHz. This makes the usage of dynamic techniques unattractive for certain low performance products such as watches or processors that need to provide conditional clocks (where there are no



(a) Leakage sources (b) Effect on waveforms

Figure 6.52 Leakage issues in dynamic circuits.

guarantees on minimum clock rates). Note that the PMOS precharge device also contributes some leakage due the reverse bias diode (source 3) and subthreshold conduction (source 4). To some extent, the leakage current of the PMOS counteracts the leakage due to the pull-down path. As a result the output voltage is going to be set by the resistive divider composed of the pull-down and pull-up paths.

Example 6.15Example of leakage

Consider the simple inverter with all devices set at 0.5µm/0.25µm. Assume that the input is low during the evaluation period. Ideally, the output should remain at the precharged state of V_{DD}. However, as seen from Figure 6.53 the output voltage drops. Once the output drops below the switching threshold of the fan-out logic gate, the output is interpreted as a low voltage. Notice that the output settles to an intermediate voltage. This is due to the leakage provided by the PMOS pull up device.

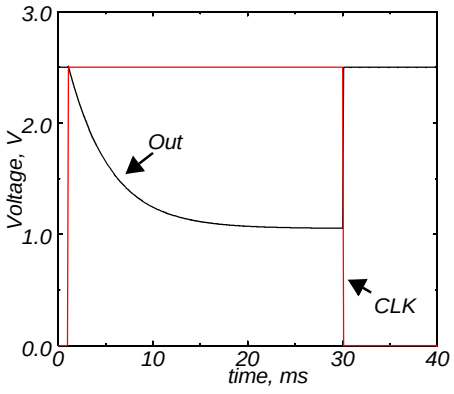


Figure 6.53 Impact of charge leakage. The output settles to an intermediate voltage determined by a resistive divider of the pull-down and pull up devices.

Leakage is caused by the high impedance state of the output node when the pull down path is turned off during the evaluate mode. To deal with the leakage problem, the impedance on the output node must be reduced during evaluate period. This is often done

by adding a bleeder transistor to the output node, as shown in Figure 6.54. The bleeder compensates for the charge lost due to the pull-down leakage paths. When the clock is high and the pull-down network is turned *off*, the output remains high. In order to avoid the ratioed problems associated with this circuit, the bleeder resistance is made high (small device size). This allows the pull-down devices has to be strong enough to pull-down the *Out* node below the switching threshold of the inverter. The circuit does have static power dissipation when *Out* is pulled low during the evaluation period. Often, the bleeder is implemented in a feedback configuration to eliminate static power dissipation.

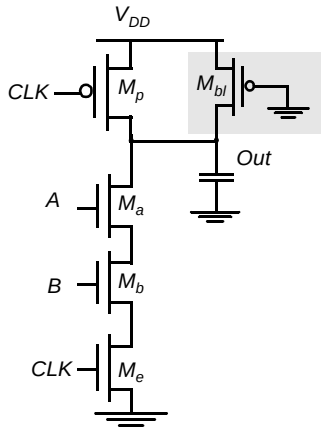


Figure 6.54 Static bleeder to compensate for the charge leakage problem.

Charge Sharing

Another important consideration in dynamic logic is charge sharing. Consider the circuit of Figure 6.55. During the precharge phase, the output node is precharged to V_{DD}. Assume that all inputs are set to 0 during precharge and that the capacitance C_a is discharged. Assume further that input B remains at 0 during evaluation, while input A makes a $0 \rightarrow 1$ transition, turning transistor M_a *on*. The charge stored originally on capacitor C_L is redistributed over C_L and C_a. This causes a drop in the output voltage, which cannot be recovered due to the dynamic nature of the circuit.

The influence on the output voltage is readily calculated. Under the above assumptions, the following initial conditions are valid: $V_{out}(t = 0) = V_{DD}$ and $V_X(t = 0) = 0$. Two cases must be considered:

1. $\Delta V_{out} < V_{Tn}$—In this case, the final value of V_X equals $V_{DD} - V_{Tn}(V_X)$. Charge conservation yields

$$C_L V_{DD} = C_L V_{out}(t) + C_a[V_{DD} - V_{Tn}(V_X)]$$

or

$$\Delta V_{out} = V_{out}(t) - V_{DD} = -\frac{C_a}{C_L}[V_{DD} - V_{Tn}(V_X)] \tag{6.32}$$

2. $\Delta V_{out} > V_{Tn}$—V_{out} and V_X reach the same value:

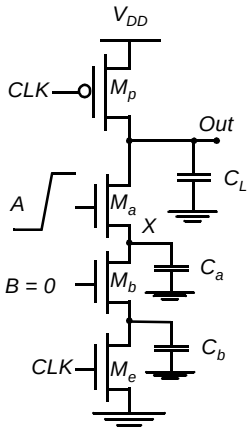


Figure 6.55 Charge sharing in dynamic networks.

$$\Delta V_{out} = -V_{DD}\left(\frac{C_a}{C_a + C_L}\right) \tag{6.33}$$

Overall, it is desirable to keep the value of ΔV_{out} below $|V_{Tp}|$. The output of the dynamic gate might be connected to a static inverter, in which case the low level of V_{out} would cause static power consumption. One major concern is circuit malfunction if the output voltage is brought below the switching threshold of the gate it drives.

Example 6.16 Charge Sharing Example

Consider the dynamic logic gate shown in Figure 6.56. It can be easily verified that the function implemented by the logic gate is $y = A \oplus B \oplus C$. To analyze charge sharing, the question we will ask is for the case when the output is nominally supposed to stay high, what is the worst case change in voltage on node y. For simplicity, ignore the load inverter, and assume that all inputs are low during the precharge operation and that all isolated internal nodes (V_a, V_b, V_c, and V_d) are initially at 0V.

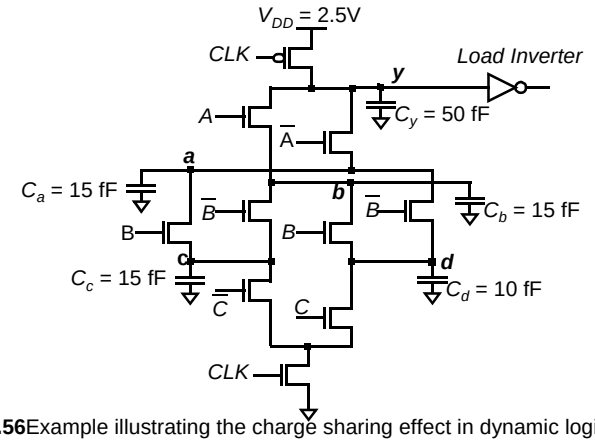


Figure 6.56 Example illustrating the charge sharing effect in dynamic logic.

There four possible cases when the output remains high and the challenge is to find the combination of inputs that results in the maximum change of the output voltage. The worst case change in output is obtained by exposing the maximum amount of internal capacitance to the output node during the evaluation period. This happens when $\overline{A}\ B\ C$ or $A\ \overline{B}\ C$. The voltage change can be easily obtained by equating the initial charge with the final charge as done with equation Eq. (6.33). Doing this results in a worst case change of 30/(30+50) * 2.5V = 0.94V. To ensure the circuit functions correctly, the switching threshold of the inverter should be placed below 2.5- 0.94 = 1.56V.

The most common and effective approach to deal with the charge redistribution is to precharge the (critical) internal nodes as well, as shown in Figure 6.57b. Since the internal nodes are charged to V_{DD} during precharge, there is no problem with charge sharing. However, this solution obviously comes at the cost of increased area and capacitance..

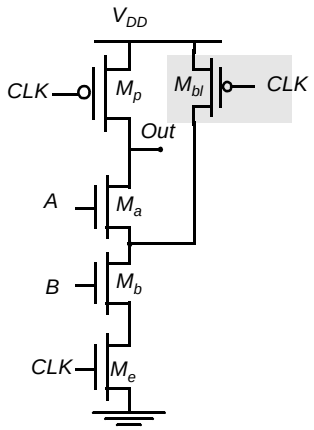


Figure 6.57 Dealing with the charge sharing problem by precharging internal nodes. An NMOS precharge transistor may also be used, however, this requires an inverted clock.

Capacitive Coupling

Capacitive coupling is another major problem in dynamic circuits. There are many forms of capacitive coupling that arise due to floating nodes in dynamic circuits. For example, a wire routed over a dynamic node can capacitively couple and destroy the state of a floating node. Another equally important form of capacitive coupling is backgate coupling. Consider the circuit shown in Figure 6.58 in which a dynamic two input NAND gate drives a static NAND gate. Assume that the input *IN* is initially low during the precharge operation. Also assume that the inputs *A* and *B* are low during the entire precharge and evaluate period. Therefore, Out_1 should remain ideally in the high state (ignoring leakage). If *IN* goes high during the evaluate period, the output of the static gate, Out_2 should be pulled low. In this process, due to capacitive backgate coupling between the internal and output node of the static gate and the output of the dynamic gate, Out_1 node voltages reduces. A simulation of this is shown in Figure 6.59. As seen from this simulation, the output of the dynamic gate can drop significantly. As a result, the output of the

static NAND gate does not drop all the way down to 0V and a small amount of static power is dissipated. If the voltage drop is large enough, the circuit can evaluate incorrectly since the NAND output may not go low. In general, care must be taken to design circuits to minimize noise introduced by capacitive coupling.

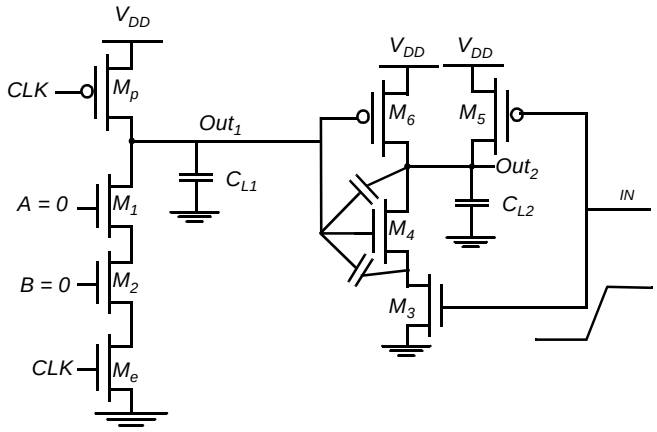


Figure 6.58 Example demonstrating the effect of backgate coupling.

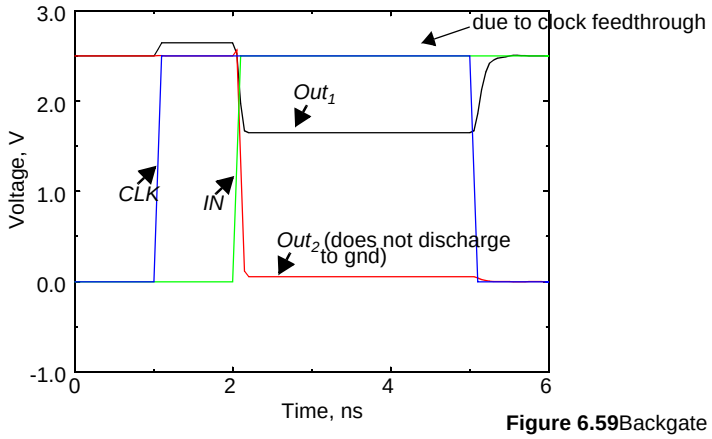


Figure 6.59 Backgate coupling effect.

Clock Feedthrough

A special case of capacitive coupling is clock feedthrough. Clock feedthrough is an effect caused by the coupling between the dynamic output storage node and the gate input of the precharge device due to the gate to drain capacitance (which includes both the overlap and the channel capacitance). During the precharge phase, the output of the dynamic gate pre-charges high. On the low the high transition of the clock, there should be no effect on the output (assuming the pull-down network is turned off). However, due to the capacitive coupling, the voltage on the output node can rise above V_{DD}. The fast rising and falling

edges of the clock couple into the signal node, as is adequately demonstrated in the simulation of Figure 6.59.

The danger of clock feedthrough is that it causes the signal level to rise sufficiently above the supply voltage that the (normally reverse-biased) junction diodes become forward-biased. This causes electron injection into the substrate, which can be collected by a nearby high impedance node in the 1 state, eventually resulting in faulty operation. CMOS latchup might be another result of this injection. For all purposes, high-speed dynamic circuits should be carefully simulated to ensure that clock feedthrough effects stay within bounds.

All the above considerations demonstrate that the design of dynamic circuits is rather tricky and requires extreme care. It should therefore only be attempted when high performance is required.

6.3.4 Cascading Dynamic Gates

So far, we have focused on the basic functionality and constraints of individual dynamic logic gates. Unfortunately, the way the circuit is implemented, dynamic gates cannot be directly cascaded. To illustrate this, consider two simple N-type dynamic inverters cascaded together, as shown in Figure 6.60a. During the precharge phase (i.e., *CLK* =0), the output of both inverters are precharged up to V_{DD}. Assume that the primary input In makes a $0 \rightarrow 1$ transition (Figure 6.60b). On the rising edge of the clock, output Out_1 starts to discharge. The second output should remain in the precharged state of V_{DD} since Out_1 transitions to 0 during evaluation. However, since there is a finite propagation delay for the input to discharge Out_1 to GND, the second output also starts to discharge. As long as Out_1 exceeds the switching threshold of the second gate, which approximately equals V_{Tn}, a conducting path exists between Out_2 and *GND*. Out_2 therefore discharges as well, resulting in incorrect evaluation. This conducting path is only turned off when Out_1 reaches V_{Tn} and shuts off the NMOS pull-down transistor. This leaves Out_2 at an intermediate voltage level. The correct level will not be recovered, since dynamic gates rely on capacitive stor-

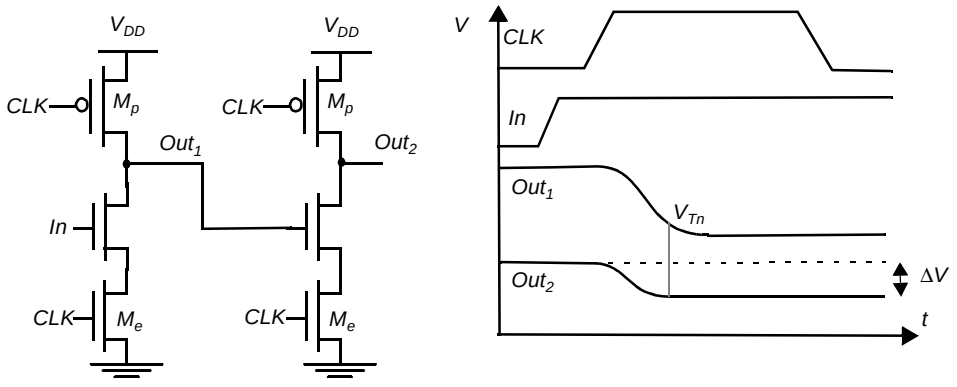


Figure 6.60 Cascade of dynamic N–type blocks.

age, in contrast to static gates, which have dc restoration. The charge loss leads to reduced noise margins and eventual malfunctioning.

It is obvious that the cascading problems arise because the output (and hence the input to the next stage) is precharged to 1. Setting the inputs to 0 during precharge could solve this problem. In doing so, all logic transistors of the next function block are turned off after precharge, and no inadvertent discharging of the storage capacitors can occur during evaluation. In other words, correct operation is guaranteed (ignoring charge redistribution and leakage) as long as **the inputs can only make a single 0 → 1 transition during the evaluation period**. This eliminates the inadvertent discharging since transistors will only be turned on when needed and at most one time per cycle. A number of design styles complying with the above rule have been developed. The two most important ones are discussed below.

Domino Logic

A Domino logic module [Krambeck82] consists of an N-type dynamic logic block followed by a static inverter (Figure 6.61). During precharge, the output of the N-type dynamic gate is charged up to V_{DD} and the output of the inverter is set to 0. During evaluation, based on the inputs, the dynamic gate conditionally discharges and the output of the inverter makes a conditional transition from 0 → 1. The input to a Domino gate always comes from the output of another Domino gate. This ensures that all inputs to the Domino gate are set to 0 at end of the precharge period. Hence, the only possible transition for the input during the evaluation period is the 0 → 1 transition, so that the formulated rule is obeyed. The introduction of the static inverter has the additional advantage that the fan-out of the gate is driven by a static inverter with a low-impedance output, which increases noise immunity. The buffer furthermore reduces the capacitance of the dynamic output node by separating internal and load capacitances.

Consider now the operation of a chain of Domino gates. During precharge, all inputs are set to 0. During evaluation, the output of the first Domino block either stays at 0 or makes a 0 → 1 transition, affecting the second Domino. This effect might ripple through the whole chain, one after the other, as with a line of falling dominoes—hence the name. Domino CMOS has the following properties:

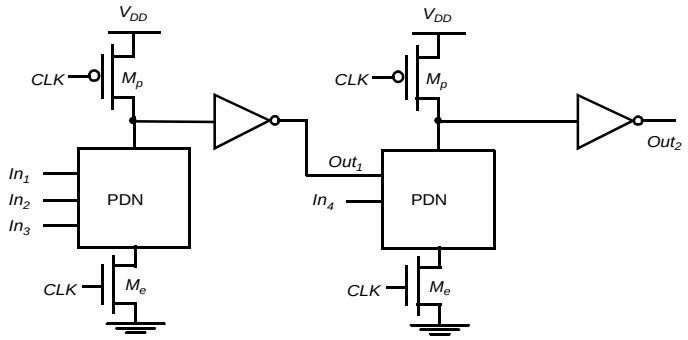


Figure 6.61 DOMINO CMOS logic.

- Since each dynamic gate has a static inverter, only noninverting logic can be implemented. This is major limiting factor, and though there are ways to deal with this (as will be discussed), pure Domino design have become rare.
- Very high speeds can be achieved: only a rising edge delay exists, while t_{pHL} equals zero (as the output node is precharged low). The static inverter can be optimized to match the *fan-out*, which is already much smaller than in the complimentary static CMOS case (only a single gate capacitance per input).

Since the inputs to a Domino gate are low during precharge, it is tempting to eliminate the evaluation transistor as it reduces clock load and increases pull-down drive. However, eliminating the evaluation device results in a performance degradation since the precharge has to ripple through the critical path. Consider the simple logic network shown in Figure 6.62, where the evaluation device has been eliminated. If the primary input In_1 is 1 during evaluation, the output of each dynamic gate is 0 and the output of each static inverter is 1. On the falling edge of the clock, we start the precharge operation assuming In_1 makes a high-to-low transition. Unfortunately, the circuit exhibits *ripple* precharge. The input to the second gate is initially high and it takes two gate delay before In_2 is driven low. During this time, the second gate cannot precharge its output to V_{DD} since the pull-down device is fighting the precharge device. Similarly, the third gate has to wait till the second gate precharges before it can start precharging, etc. Therefore the time taken to precharge the logic circuit is equal to the critical path of the logic circuit. Another important problem here is the static power dissipation due to the fight between the pull-up and pull-down devices. As a result of this, the evaluation device is almost always placed in the circuit.

Dealing with the Non-inverting Property of Domino Logic

A major limitation in Domino logic is that only non-inverting logic can be implemented. This is due to the inclusion of the static inverter at the output of each dynamic gate. This requirement has limited the widespread use of pure Domino logic. There are several ways to deal with the problem of non-inverting logic requirement. Figure 6.63 shows one approach to the problem, which basically involves reorganizing the logic using

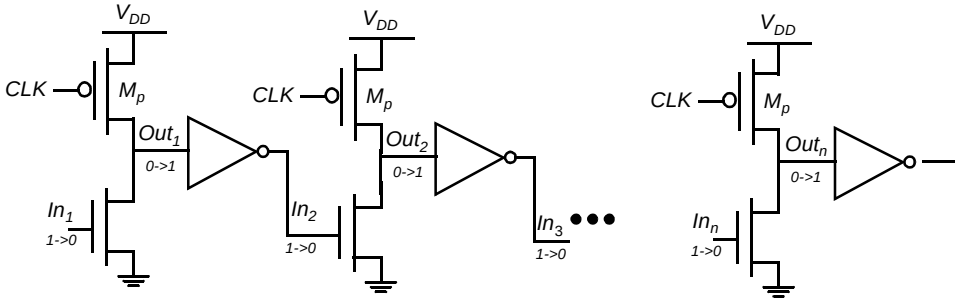


Figure 6.62 Effect of ripple precharge when the evaluation transistor is removed. The circuit also exhibits static power dissipation.

simple boolean transforms, such as De Morgan's Law. Unfortunately, this sort of optimization is not always possible, and more general schemes must be used.

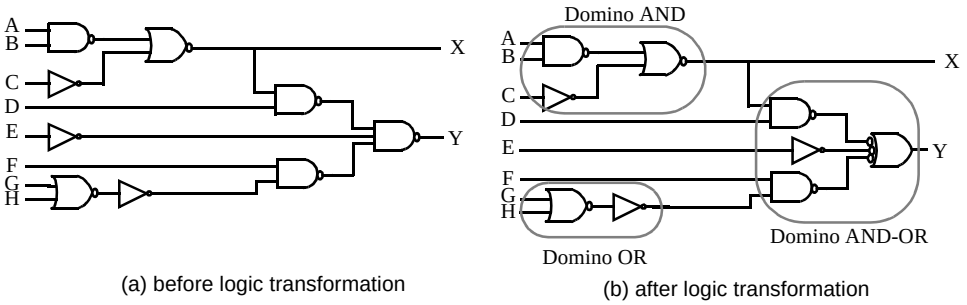


Figure 6.63 Restructuring logic to enable implementation using non-inverting Domino Logic.

A general, but expensive, approach to solving the problem of the non-inverting logic requirement is the use of dual-rail coding. Dual-rail Domino is similar in concept to the DCVS structure discussed earlier, but uses a precharged load instead of a static cross-coupled PMOS load. Figure 6.64 shows the circuit schematic of a simple AND/NAND differential logic gate. Note that all inputs come from other differential Domino gates and therefore, all inputs are low during the precharge phase and make a conditional transition from 0 to 1. Using differential Domino, it is possible to implement any arbitrary function. Differential Domino gates consume significant power since they have a guaranteed transition every single clock cycle, regardless of the input values since either O or $\overline{O}$ will make a 0 to 1 transition. The function of transistors M_{f1} and M_{f2} is to keep the circuit static when the clock is high for extended periods of time. Notice that though there is a cross coupled PMOS pair, this circuit is not ratioed! Such a differential approach is very popular and is used in several commercial microprocessors.

Optimization of Domino Logic Gates

There are a several optimizations that can be performed on Domino logic gates. The most obvious performance optimization involves the sizing of transistors in the static

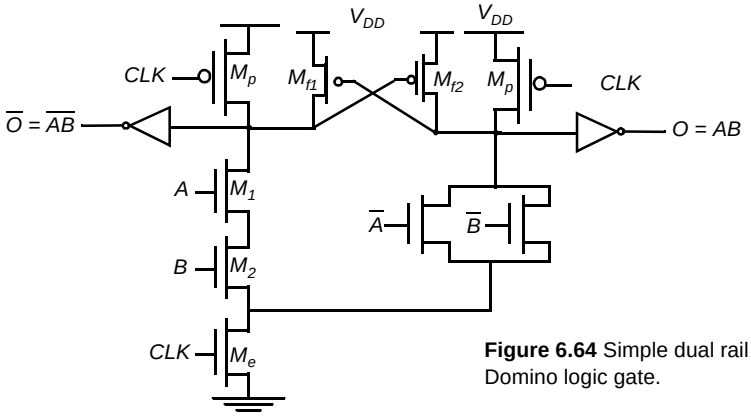


Figure 6.64 Simple dual rail (differential) Domino logic gate.

inverter. With the inclusion of the evaluation devices in Domino circuits, all gates precharge in parallel and the precharge operation is only two gates as the output of the dynamic gate charges to V_{DD} and the inverter output is driven low. The critical path during evaluation happens through the pull-down path of the dynamic gate and the PMOS pull-up path of the static inverter. Therefore, to speed up the circuit, the beta ratio of the static inverter should be made high so that the switching threshold is close to V_{DD}. This can be accomplished by using a small (minimum) sized NMOS and a large PMOS device. The minimum sized NMOS does not affect the performance since the precharge happens in parallel. The only disadvantage of using a large beta ratio is a reduction in noise margin is reduced. Issues such as charge sharing and backgate coupling can cause the dynamic gate voltage to drop below V_{DD} and hence a high switching threshold can cause an incorrect evaluation. The device sizing of the inverter should simultaneously consider the reduced noise margin and performance.

Numerous circuit variations of Domino circuits have been proposed [Bernstein98]. One optimization that reduces area is Multiple Output Domino Logic. The basic concept is illustrated is Figure 6.65. The idea is to exploit the partial trees in the pull-down network and the fact that certain outputs are subsets of other outputs. In this example, *O3 = C+D* is used in all three outputs, and hence it is implemented in the bottom of the pull-down network. Since *O2* is simply *B ·O3*, it is connected in series with the logic for *O3*. Notice that the internal nodes have to be precharged to V_{DD} since the outputs are based on internal nodes. Given that the internal node precharge to V_{DD}, the number of devices driving precharge devices is not reduced. However, the number of devices driving the evaluation switch is reduced since the overhead of evaluation device is amortized over multiple outputs. Also the number of transistors required in the circuit is clearly reduced since the logic for *O3* and *O2* must be duplicated in implementing *O1* if the three logic functions were implemented independently.

Another optimization of the generic Domino logic gate is Compound Domino (Figure 6.66). The basic goal of this style is to minimize the number of devices in a dynamic logic gate. Instead of each dynamic gate driving a static inverter, it is possible to combine the output of multiple dynamic gates using complex static CMOS gates as shown in Figure 6.66. In this example, we have three dynamic gates which include $o1 = \overline{A\,B\,C}$, $o2 = \overline{D\,E\,F}$ and $o3 = \overline{G\,H}$. The output of three dynamic structures are combined using a complex

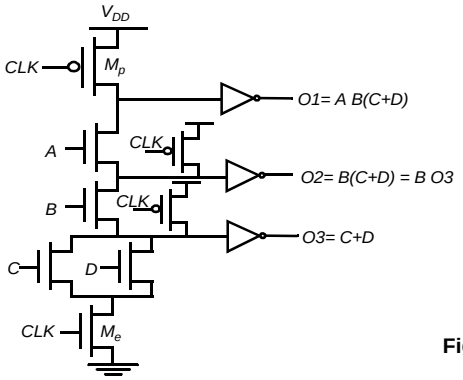


Figure 6.65 Multiple output Domino

CMOS static gate whose function is $O = \overline{(o1+o2)\ o3}$. For this example, this equates to $O = A\ B\ C\ D\ E\ F + GH$.

Compound Domino is a useful tool for constructing complex dynamic logic gates. Large dynamic stacks are replaced using using parallel small fan-in structures and complex CMOS gates. For example, a large *fan-in* Domino AND can be implemented as parallel dynamic NAND structures with lower *fan-in* that are combined using a static NOR gate. One important consideration in Compound Domino is the problem associated with backgate coupling. Care must be taken to ensure that the dynamic nodes are not affected by the coupling between the output of the static gates and the output of dynamic nodes.

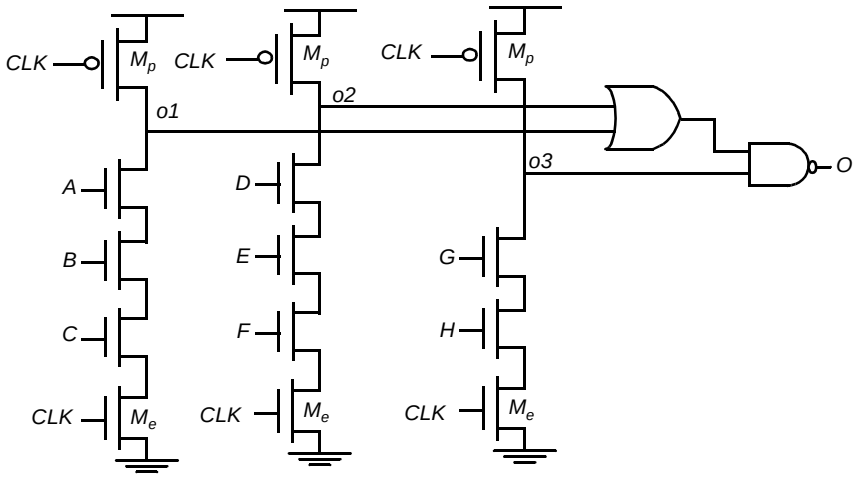


Figure 6.66 Compound Domino logic where complex static gates can be placed at the output of dynamic gates.

np-CMOS

The Domino logic presented in the previous section has the disadvantage that each dynamic gate requires an extra static inverter in the critical path to make the circuit functional. *np*-CMOS, provides an alternate race-free approach to cascading dynamic logic by using two flavors (N-tree and P-tree) of dynamic logic. In a P-tree logic gate, PMOS devices are used to build a pull-up network that has a series evaluation device (Figure 6.67) ([Goncalvez83, Friedman84, Lee86]). A predischarge device drives the output low during precharge and the output conditionaly makes a $0 \rightarrow 1$ transition based on its inputs.

np-CMOS logic exploits the duality between N-tree and P-tree logic gates to eliminate races. The N-tree gates are controlled by CLK and P-tree gates are controlled using $\overline{CLK}$. N-tree gates can directly drive P-tree gates, but similar to Domino, N-tree outputs must go through an inverter if another N-tree gate is to be driven. During the precharge phase ($CLK = 0$), the output of N-tree logic, *Out1*, is charge up to V_{DD} and the output of P-tree, *Out2*, is predischarged to 0V. Since N-tree logic drives PMOS pull-up devices, the PUN of the P-tree is turned off. If some of the P-tree inputs (i.e., the N-tree outputs) are discharged during the evaluation period ($CLK = 1$), and the PUN turns *on*, the output of P-type gates make a 0 to 1 transition. In a similar way, N-tree blocks can follow P-tree

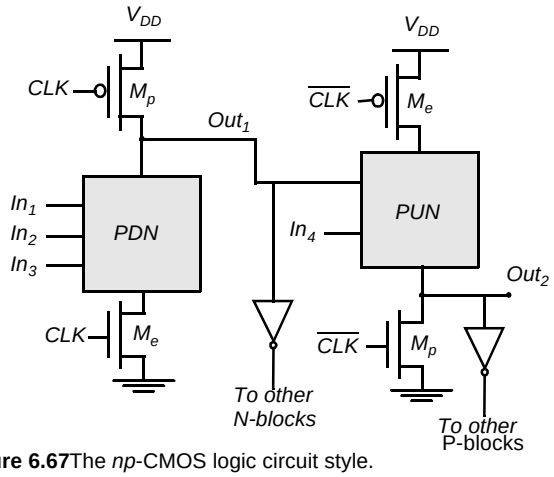


Figure 6.67 The *np*-CMOS logic circuit style.

blocks without any problems, as the precharge value of inputs equals 0. A disadvantage of the *np*-CMOS logic style is that the P-tree blocks are slower than the N-tree modules, due to the lower mobility of the PMOS transistors in the logic network. Equalizing the propagation delays requires extra area.

6.4 Perspective: How to Choose a Logic Style

In the preceding sections, we have discussed several gate-implementation approaches using the CMOS technology. Each of the circuit styles has its advantages and disadvantages. Which one to select depends upon the primary requirement: ease of design, robustness, area, speed, or power dissipation. No single style optimizes all these measures at the same time. Even more, the approach of choice can vary from logic function to logic function.

The static approach has the advantage of being robust in the presence of noise. This makes the design process rather trouble-free and amenable to a high degree of automation. This ease-of-design does not come for free: for complex gates with a large fan-in, complementary CMOS becomes expensive in terms of area and performance. Alternative static logic styles have therefore been devised. Pseudo-NMOS is simple and fast at the expense of a reduced noise margin and static power dissipation. Pass-transistor logic is attractive for the implementation of a number of specific circuits, such as multiplexers and XOR-dominated logic such as adders.

Dynamic logic, on the other hand, makes it possible to implement fast and small complex gates. This comes at a price. Parasitic effects such as charge sharing make the design process a precarious job. Charge leakage forces a periodic refresh, which puts a lower bound on the operating frequency of the circuit.

The current trend is towards an increased use of complementary static CMOS. This tendency is inspired by the increased use of design-automation tools at the logic design level. These tools emphasize optimization at the logic rather than the circuit level and put

a premium on robustness. Another argument is that static CMOS is more amenable to voltage scaling than some of the other approaches discussed in this chapter.

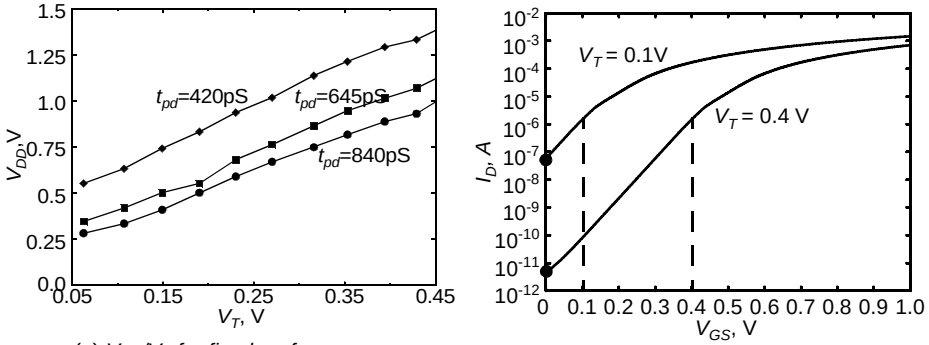
6.5 Leakage in Low Voltage Systems

As power supply voltage scale, the device thresholds must scale to maintain performance. Figure 6.68a shows a plot of power supply and V_T required in order to maintain a fixed performance level. This tradeoff is not without penalty however, as subthreshold leakage currents increase exponentially as V_T is reduced. The leakage can be approximated as follows:

$$I_{leakage} = I_0 e^{\frac{V_{GS}-V_T}{nV_{th}}} = I_0 10^{\frac{V_{GS}-V_T}{S}} \tag{6.34}$$

The subthreshold $S = nV_{th}\ ln(10)$. For a typical technology with a subthreshold slope of 100 mV/decade, each 100mV decrease in V_T will cause an order magnitude change in leakage currents. The leakage of an inverter is current of the NMOS when V_{in} = 0V and the output is at V_{DD}. The exponential increase in leakage when the threshold is decreased is demonstrated in Figure 4.26b.

Leakage currents are particularly a concern for event driven computation in which intermittent computation activity triggered by external events is separated by long periods of inactivity (e.g., the processor in a cellular phone or PDA remains in the idle mode for majority of the time). While the processor is shutdown, the system should ideally consume near zero power. This is only possible if the devices consume low levels of leakage power - i.e., the devices have a high threshold voltage. However for low voltage high-performance operation, reduced threshold devices are required. To satisfy the contradicting requirements of high-performance during active periods and low-standby leakage, several device technologies have recently been introduced. This includes the control of threshold voltages in triple-well CMOS using backgate effect and the use of multiple threshold devices.



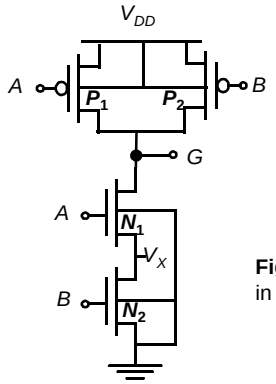
(a) *V_{DD}/V_T for fixed performance* (b) *Leakage as a function of V_T*

Figure 6.68 Voltage Scaling (V_{DD}/V_T on delay and leakage)

As with propagation delay, static properties, and switching activity, leakage of logic gates has a strong dependence on the input patterns. The leakage is a function of the circuit topology and the value of inputs. This is due to the fact that V_T depends on body bias (V_{BS}) we observe that sub-threshold leakage of a long channel MOS transistor depends on gate drive (V_{GS}) and body bias (V_{BS}). So, in an inverter with *IN = 0* the sub-threshold leakage of the inverter will be set by the NMOS transistor with its $V_{GS} = V_{BS} = 0$ *V*. In more complex CMOS gates the leakage current will depend on the input vector, for example, one can show (Eq. (6.35)) that the sub-threshold leakage current of a two-input NAND gate will be the least when *A = B = 0*. Under this condition the intermediate node in will settle to,

$$V_X \approx V_{th} \ln(1 + n) \tag{6.35}$$

The NAND gate sub-threshold leakage then will be set by the NMOS transistor with its $V_{GS} = V_{BS} = -V_X$. Clearly, the sub-threshold leakage under this condition will be slightly smaller than that of the inverter or a stand-alone NMOS transistor's I_{OFF}. This reduction in sub-threshold leakage due to stacked transistors is called the *stack effect*. Figure 6.69 shows the leakage components for a simple two input NAND gate.



A	B	V_X	I_{SUB}
0	0	V_{th} ln (1+n)	I^N_{SUB} ($V_{GS} = V_{BS} = -V_X$)
0	1	0	I^N_{SUB} ($V_{GS} = V_{BS} = 0$)
1	0	$V_{dd}-V_T$	I^N_{SUB} ($V_{GS} = V_{BS} = 0$)
1	1	0	$2\,I^P_{SUB}$ ($V_{SG} = V_{SB} = 0$)

Figure 6.69 Sub-threshold leakage reduction due to stack effect in a two-input NAND gate using long channel transistors.

In short channel MOS transistors the sub-threshold leakage current depends not only on the gate drive (V_{GS}) and body bias (V_{BS}), but also depends strongly on the drain voltage (V_{DS}). Threshold voltage of short channel MOS transistors decrease with increase in V_{DS} due to *drain induced barrier lowering* (DIBL). Typical value for DIBL can range from 20-150 mV change in V_T per volt change in V_{DS}.

Figure 6.70 shows the decrease in sub-threshold leakage due to (i) decrease in gate drive - point *A* to *B* and (ii) increase in body bias - point *A* to *C*, similar to long channel MOS transistors. It also illustrates the increase in sub-threshold leakage due to increase in drain voltage - point *A* to *D*. Because of this reason the impact of stack effect for leakage reduction will be more significant in short channel MOS transistors. Consider the two-input NAND gate in Figure 6.69 when both M_1 and M_2 are *off*. From the load line in Figure 6.71 we can see that in steady state V_X will settle to ~100 mV. So the steady state sub-threshold leakage in the NAND gate will be due to $V_{GS} = V_{BS} = -100$ mV and $V_{DS} = V_{DD} - 100$ mV which is 20X smaller than that leakage of a stand-alone NMOS transistor with $V_{GS} = V_{BS} = 0$ mV and $V_{DS} = V_{DD}$ [Ye98]. Because of enhanced stack effect in short channel MOS transistors the sub-threshold leakage in circuits with stacks will be significantly smaller than individual devices. Note that maximum leakage reduction due to stack effect

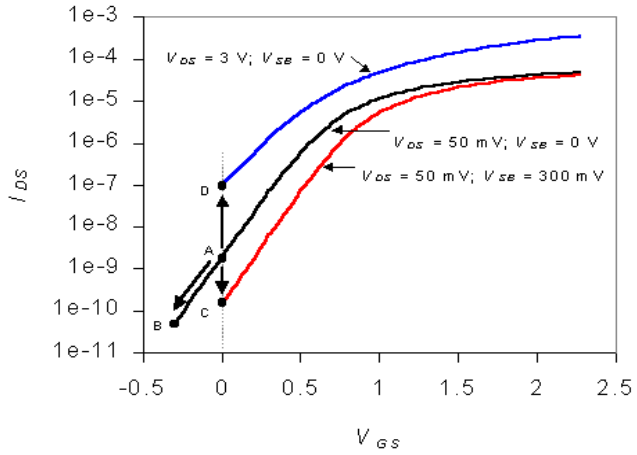


Figure 6.70 Dependence of sub-threshold leakage current on terminal voltages for a typical 0.25 μm NMOS transistor.

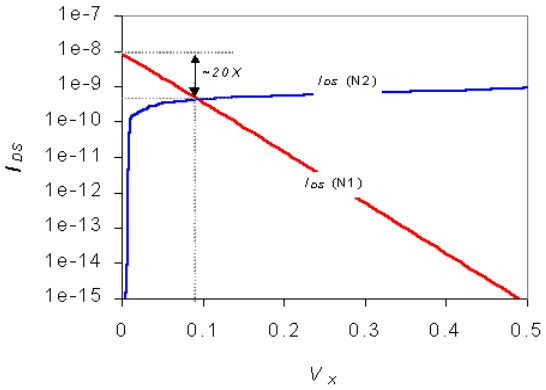


Figure 6.71 Load line indicating the steady state solution for the intermediate node voltage.

happens when all the transistors in the stack are *off* and the intermediate node voltage reaches its steady state value. So one can reduce standby leakage in a VLSI system by forcing stack effect in as many gates as possible. Since the intermediate node gets charged or discharged to its steady state value through sub-threshold currents time constant to realize maximum stack effect can be long, depending on the initial node value. A key challenge is to determine the primary input vector that minimizes the leakage effect.

Problem 6.8 Computing V_X

Eq. (6.35) represents intermediate node voltage for a two-input NAND with less than 10% error, when $A = B = 0$. Derive Eq. (6.35) assuming (i) V_T and I_o of M_1 and M_2 are approximately equal, (ii) NMOS transistors are identically sized, and (iii) $n < 1.5$. Explain the tem-

perature dependence of stack effect and leakage reduction due to stack effect using equation (4.5).

6.6 Summary

In this chapter, we have extensively analyzed the behavior and performance of combinational CMOS digital circuits with regard to area, speed, and power.

- Static complementary CMOS combines dual pull-down and pull-up networks, only one of which is enabled at any time.
- The performance of a CMOS gate is a strong function of fan-in. Techniques to deal with fan-in include transistor sizing, input reordering, and partitioning. The speed is also a linear function of the fan-out. Extra buffering is needed for large fan-outs.
- The ratioed logic style consists of an active pull-down (up) network connected to a load device. This results in a substantial reduction in gate complexity at the expense of static power consumption and an asymmetrical response. Careful transistor sizing is necessary to maintain sufficient noise margins. The most popular approaches in this class are the pseudo-NMOS techniques and the differential DCVSL, which requires complementary signals.
- Pass-transistor logic implements a logic gate as a simple switch network. This results in very simple implementations for some logic functions. Long cascades of switches are to be avoided due to a quadratic increase in delay with respect to the number of elements in the chain. NMOS-only pass-transistor logic produces even simpler structures, but might suffer from static power consumption and reduced noise margins. This problem can be addressed by adding a level-restoring transistor.
- The operation of dynamic logic is based on the storage of charge on a capacitive node and the conditional discharging of that node as a function of the inputs. This calls for a two-phase scheme, consisting of a precharge followed by an evaluation step. Dynamic logic trades off noise margin for performance. It is sensitive to parasitic effects such as leakage, charge redistribution, and clock feedthrough. Cascading dynamic gates can cause hazards and should be addressed carefully.
- The power consumption of a logic network is strongly related to the switching activity of the network. This activity is a function of the input statistics, the network topology, and the logic style.
- Sources of power consumption such as glitches and short-circuit currents can be minimized by careful circuit design and transistor sizing.
- Power consumption is minimized by reducing the supply voltage, which increases the delay. Trading off area for power is a way to compensate for that performance loss.

- Threshold voltage scaling is required for low-voltage operation. Leakage control is critical for low-voltage operation

6.7 To Probe Further

The topic of (C)MOS logic styles is treated extensively in the literature. Numerous texts have been devoted to the issue. Some of the most comprehensive treatments can be found in [Glasser85], [Annaratone86], [Elmasry91], [Uyemura92], and [Weste93]. Regarding the intricacies of high-performance design, [Shoji88] offers the most in-depth discussion of the optimization and analysis of digital MOS circuits. The topic of power minimization is relatively new. Excellent reference works are [Chandrakasan95] and [Rabaey95].

Innovations in the MOS logic area are typically published in the proceedings of the ISSCC Conference and the VLSI circuits symposium, as well as the *IEEE Journal of Solid State Circuits* (especially the November issue).

REFERENCES

[Annaratone86] M. Annaratone, *Digital CMOS Circuit Design*, Kluwer, 1986.

[Burd94] T. Burd, *Low Power CMOS Library Design Methodology*, M.S. thesis, University of California—Berkeley, December 1994.

[Chandrakasan92] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low Power CMOS Digital Design," *IEEE Journal of Solid State Circuits*, vol. SC-27, no. 4, pp. 1082–1087, April 1992.

[Chandrakasan94] A. Chandrakasan, *Low Power Digital CMOS Design*, Ph.D. thesis, University of California—Berkeley, Memorandum No. UCB/ERL M94/65, August 1994.

[Chandrakasan95] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*, Kluwer, 1995.

[Chu86] K. Chu and D. Pulfrey, "Design Procedures for Differential Cascade Logic," *IEEE Journal of Solid State Circuits*, vol. SC-21, no. 6 (Dec. 1986), pp. 1082–1087.

[Dopperpuhl92] D. Dopperpuhl et al., "A 200-MHz 64-b Dual-Issue CMOS Microprocessor," *IEEE Journal of Solid State Circuits*, vol. 27, no. 11, pp. 1555–1567, Nov. 1992.

[Elmasry91] M. Elmasry, Ed., *Digital MOS Integrated Circuits II*, IEEE Press, 1991.

[Friedman84] V. Friedman and S. Liu, "Dynamic Logic CMOS Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-19, no. 2, pp. 263–266, April 1984.

[Glasser85] L. Glasser and D. Dopperpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, 1985.

[Goncalvez83] N. Goncalvez and H. De Man, "NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures," *IEEE Journal of Solid State Circuits*, vol. SC-18, no. 3, pp. 261–266, June 1983.

[Heller84] L. Heller et al., "Cascade Voltage Switch Logic: A Differential CMOS Logic Family," *Proc. IEEE ISSCC Conference*, pp. 16–17, February 1984.

[Hodges88] D. Hodges and H. Jackson, *Analysis and Design of Digital Integrated Circuits*, McGraw-Hill, 1988.

[Krambeck82] R. Krambeck et al., "High-Speed Compact Circuits with CMOS," *IEEE Journal of Solid State Circuits*, vol. SC-17, no. 3, pp. 614–619, June 1982.

[Lee86] C. M. Lee and E. Szeto, "Zipper CMOS," *IEEE Circuits and Systems Magazine*, pp. 10–16, May 1986.

[Liu93] D. Liu and C. Svensson, "Trading Speed for Low Power by Choice of Supply and Threshold Voltages," *IEEE Journal of Solid State Circuits*, vol. SC-28, no 1, pp. 10–17, January 1993.

[Murphy81] B. Murphy and R. Edwards, "A CMOS 32b Single Chip Microprocessor," *Proc. ISCC 81*, pp. 230–231, 1981.

[Rabaey95] J. Rabaey and M. Pedram, *Low Power Design Methodolgies*, Kluwer, 1995.

[Radhakrishnan85] D. Radhakrishnan, S. Whittaker, and G. Maki, "Formal Design Procedures for Pass-Transistor Switching Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-20, no. 2, pp. 531–536, April 1985.

[Shimohigashi93] K. Shimohigashi and K. Seki, "Low-Voltage ULSI Design," *IEEE Journal of Solid State Circuits*, vol. 28, no. 4, April 1993.

[Shoji88] M. Shoji, *CMOS Digital Circuit Technology*, Prentice Hall, 1988.

[Uyemura88] J. Uyemura, *Fundamentals of MOS Digital Integrated Circuits*, Addison-Wesley, 1988.

[Uyemura92] J. Uyemura, *Circuit Design for CMOS VLSI*, Kluwer, 1992.

[Veendrick84] H. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-19, no 4, pp. 468–473, August 1984.

[Weste93] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, 1993.

[Yano90] K. Yano et al., "A 3.8 ns CMOS 16 $\times$ 16 b Multiplier Using Complimentary Pass-Transistor Logic," *IEEE Journal of Solid State Circuits*, vol. SC-25, no 2, pp. 388–395, April 1990.