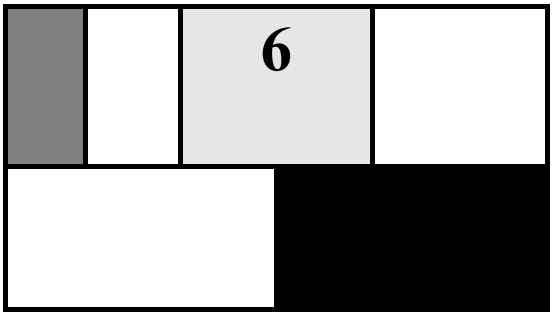
CHAPTER



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

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.

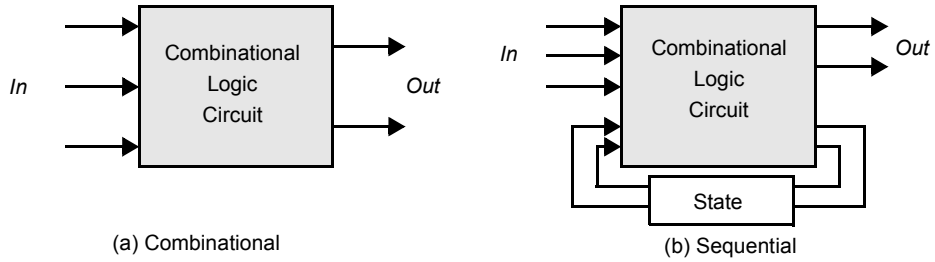


Figure 6.1 High level classification of logic circuits.

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:

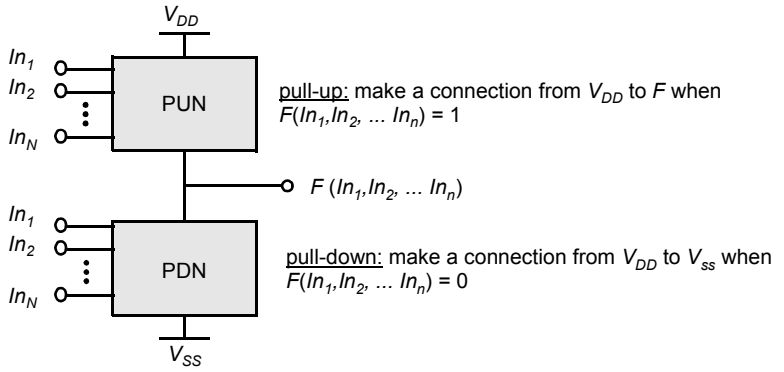
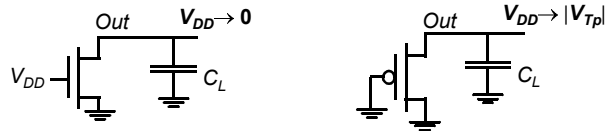
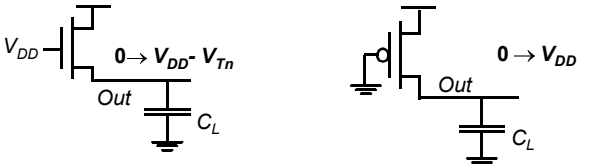


Figure 6.2 Complementary logic gate as a combination of a PUN (pull-up network) and a PDN (pull-down network).

- 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

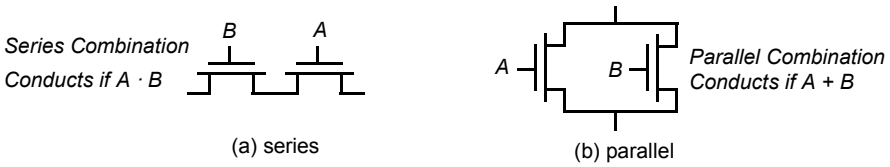


Figure 6.4 NMOS logic rules — series devices implement an AND, and parallel devices implement an OR.

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

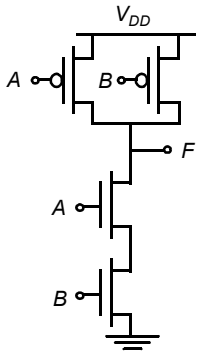
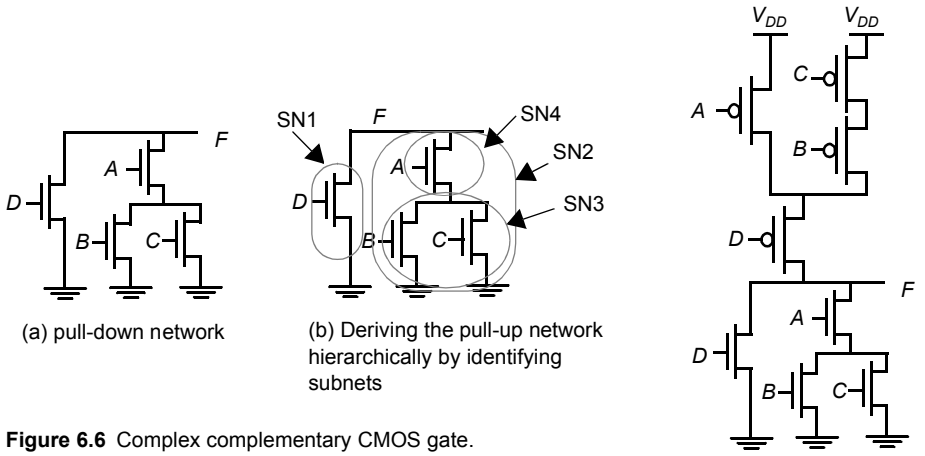


Table 6.1 Truth Table for 2 input NAND

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) $A = 1$,

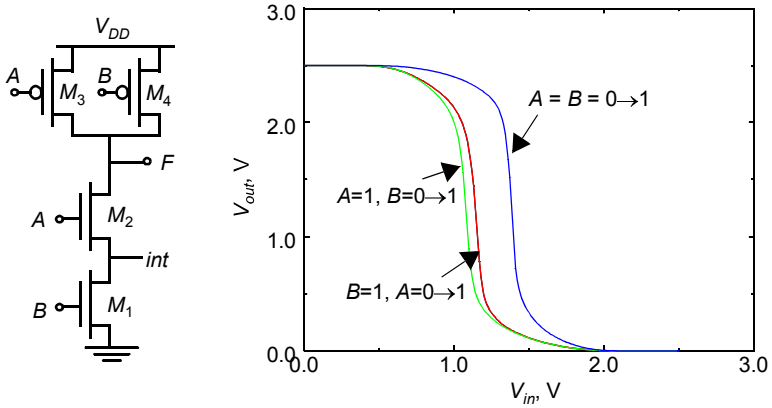


Figure 6.7 The VTC of a two-input NAND is data-dependent. NMOS devices are 0.5μm/0.25μm while the PMOS devices are sized at 0.75μm/0.25μm.

$B = 0 \rightarrow 1$, 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.

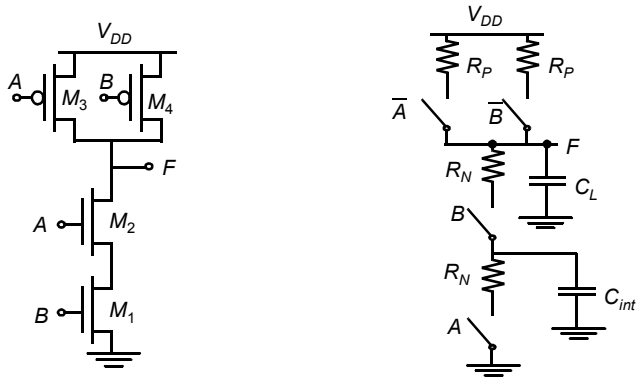
Design Consideration

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

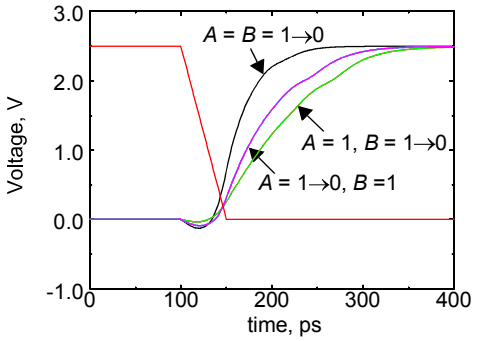
NMOS devices in the NAND stack must be made twice as large (i.e., NMOS of NAND should be 0.75μm/0.25μm) so that the equivalent resistance the NAND pull-down is the same as the inverter. The PMOS device can remain unchanged.

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.

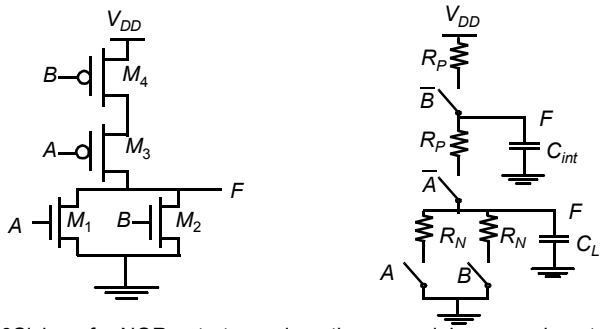


Figure 6.10 Sizing of a NOR gate to produce the same delay as an inverter with size of NMOS: 0.5μm/0.25μm and PMOS: 1.5μm/0.25μm.

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.

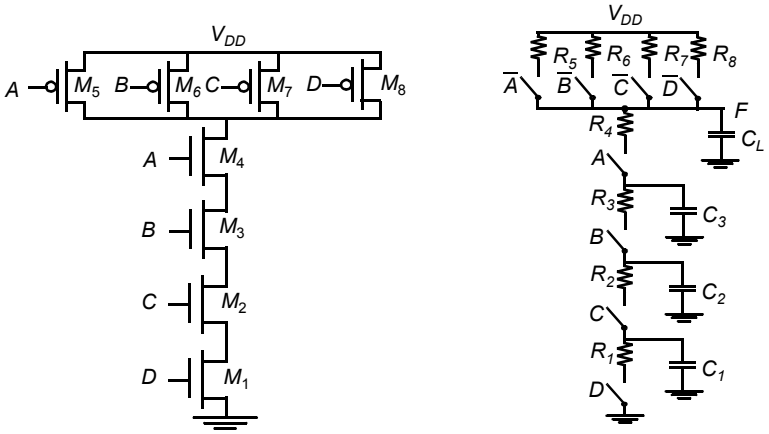


Figure 6.11 Four input NAND gate along with the internal node capacitances.

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).

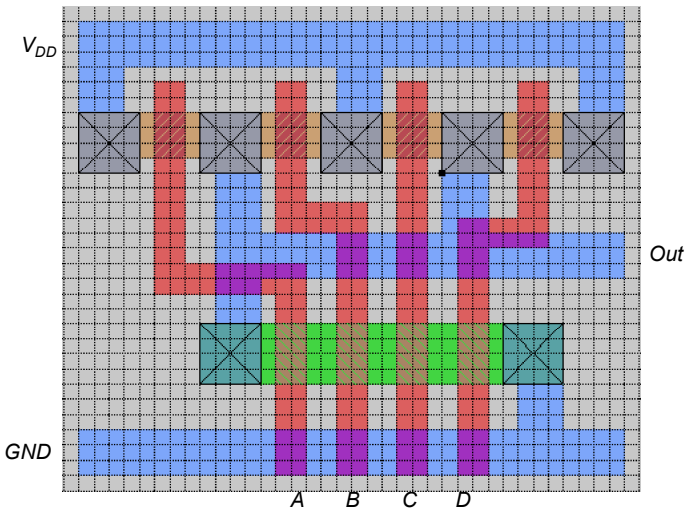


Figure 6.12 Layout a four-input NAND gate in complementary CMOS.

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

Transistor	W (μm)	AS (μm^2)	AD (μm^2)	PS (μm)	PD(μm)
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.

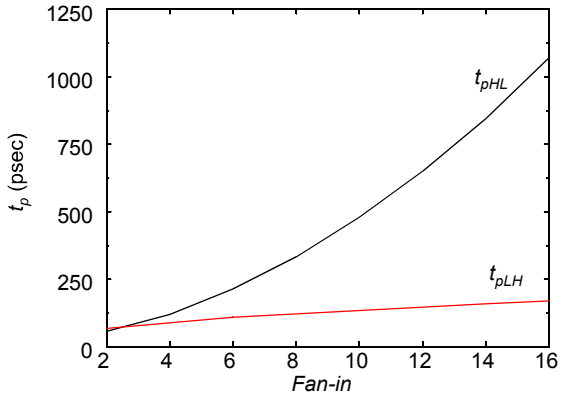


Figure 6.13 Propagation delay of CMOS NAND gate as a function of fan-in. A fan-out of one inverter is assumed, and all pull-down transistors are minimal size.

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