The time for the output to discharge to $V_{DD}/2$ is thus:

$$t_{pd} = \frac{CV_{DD}}{2I_{\text{eff}}} \tag{4.18}$$

Equating this to $t_{pd} = RC$ gives

$$R = \frac{V_{DD}}{2I_{\text{eff}}} = \frac{V_{DD}}{I_H + I_L} \tag{4.19}$$

4.4 Linear Delay Model

The RC delay model showed that delay is a linear function of the fanout of a gate. Based on this observation, designers further simplify delay analysis by characterizing a gate by the slope and *y*-intercept of this function. In general, the normalized delay of a gate can be expressed in units of τ as

$$d = f + p \tag{4.20}$$

p is the *parasitic delay* inherent to the gate when no load is attached. f is the *effort delay* or *stage effort* that depends on the complexity and fanout of the gate:

$$f = gh \tag{4.21}$$

The complexity is represented by the *logical effort*, g [Sutherland99]. An inverter is defined to have a logical effort of 1. More complex gates have greater logical efforts, indicating that they take longer to drive a given fanout. For example, the logical effort of the 3-input NAND gate from the previous example is 5/3. A gate driving h identical copies of itself is said to have a *fanout* or *electrical effort* of h. If the load does not contain identical copies of the gate, the electrical effort can be computed as

$$h = \frac{C_{\text{out}}}{C_{\text{in}}} \tag{4.22}$$

where C_{out} is the capacitance of the external load being driven and C_{in} is the input capacitance of the gate.[4]

Figure 4.21 plots normalized delay vs. electrical effort for an idealized inverter and 3-input NAND gate. The *y*-intercepts indicate the parasitic delay, i.e., the delay when the gate drives no load. The slope of the lines is the logical effort. The inverter has a slope of 1 by definition. The NAND has a slope of 5/3.

The remainder of this section explores how to estimate the logical effort and parasitic delay and how to use the linear delay model.

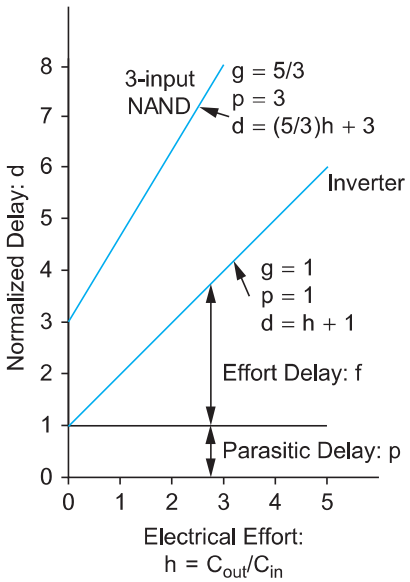

FIGURE 4.21
Normalized delay vs. fanout

[4]Some board-level designers say a device has a fanout of h when it drives h other devices, even if the other devices have different capacitances. This definition would not be useful for calculating delay and is best avoided in VLSI design. The term *electrical effort* avoids this potential confusion and emphasizes the parallels with logical effort.

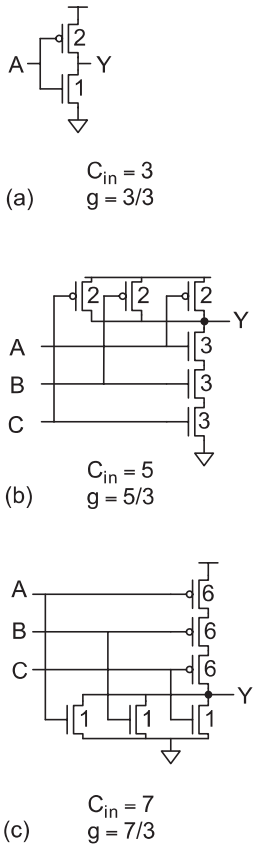

FIGURE 4.22 Logic gates sized for unit resistance

4.4.1 Logical Effort

Logical effort of a gate is defined as *the ratio of the input capacitance of the gate to the input capacitance of an inverter that can deliver the same output current.* Equivalently, logical effort indicates how much worse a gate is at producing output current as compared to an inverter, given that each input of the gate may only present as much input capacitance as the inverter.

Logical effort can be measured in simulation from delay vs. fanout plots as the ratio of the slope of the delay of the gate to the slope of the delay of an inverter, as will be discussed in Section 8.5.3. Alternatively, it can be estimated by sketching gates. Figure 4.22 shows inverter, 3-input NAND, and 3-input NOR gates with transistor widths chosen to achieve unit resistance, assuming pMOS transistors have twice the resistance of nMOS transistors.[5] The inverter presents three units of input capacitance. The NAND presents five units of capacitance on each input, so the logical effort is 5/3. Similarly, the NOR presents seven units of capacitance, so the logical effort is 7/3. This matches our expectation that NANDs are better than NORs because NORs have slow pMOS transistors in series.

Table 4.2 lists the logical effort of common gates. The effort tends to increase with the number of inputs. NAND gates are better than NOR gates because the series transistors are nMOS rather than pMOS. Exclusive-OR gates are particularly costly and have different logical efforts for different inputs. An interesting case is that multiplexers built from ganged tristates, as shown in Figure 1.29(b), have a logical effort of 2 independent of the number of inputs. This might at first seem to imply that very large multiplexers are just as fast as small ones. However, the parasitic delay does increase with multiplexer size; hence, it is generally fastest to construct large multiplexers out of trees of 4-input multiplexers [Sutherland99].

TABLE 4.2 Logical effort of common gates

Gate Type	Number of Inputs				
	1	2	3	4	*n*
inverter	1				
NAND		4/3	5/3	6/3	$(n+2)/3$
NOR		5/3	7/3	9/3	$(2n+1)/3$
tristate, multiplexer	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

4.4.2 Parasitic Delay

The parasitic delay of a gate is the delay of the gate when it drives zero load. It can be estimated with RC delay models. A crude method good for hand calculations is to count only diffusion capacitance on the output node. For example, consider the gates in Figure 4.22, assuming each transistor on the output node has its own drain diffusion contact. Transistor widths were chosen to give a resistance of R in each gate. The inverter has three units of diffusion capacitance on the output, so the parasitic delay is $3RC = \tau$. In other words,

[5] This assumption is made throughout the book. Exercises 4.19–4.20 explore the effects of different relative resistances (see also [Sutherland99]). The overall conclusions do not change very much, so the simple model is good enough for most hand estimates. A simulator or static timing analyzer should be used when more accurate results are required.

the normalized parasitic delay is 1. In general, we will call the normalized parasitic delay $p_{inv} \cdot p_{inv}$ is the ratio of diffusion capacitance to gate capacitance in a particular process. It is usually close to 1 and will be considered to be 1 in many examples for simplicity. The 3-input NAND and NOR each have 9 units of diffusion capacitance on the output, so the parasitic delay is three times as great ($3p_{inv}$, or simply 3). Table 4.3 estimates the parasitic delay of common gates. Increasing transistor sizes reduces resistance but increases capacitance correspondingly, so parasitic delay is, on first order, independent of gate size. However, wider transistors can be folded and often see less than linear increases in internal wiring parasitic capacitance, so in practice, larger gates tend to have slightly lower parasitic delay.

TABLE 4.3 Parasitic delay of common gates

Gate Type	Number of Inputs				
	1	2	3	4	*n*
inverter	1				
NAND		2	3	4	*n*
NOR		2	3	4	*n*
tristate, multiplexer	2	4	6	8	2*n*

This method of estimating parasitic delay is obviously crude. More refined estimates use the Elmore delay counting internal parasitics, as in Example 4.7, or extract the delays from simulation. The parasitic delay also depends on the ratio of diffusion capacitance to gate capacitance. For example, in a silicon-on-insulator process in which diffusion capacitance is much less, the parasitic delays will be lower. While knowing the parasitic delay is important for accurately estimating gate delay, we will see in Section 4.5 that the best transistor sizes for a particular circuit are only weakly dependent on parasitic delay. Hence, crude estimates tend to be sufficient to reach a good circuit design.

Nevertheless, it is important to realize that parasitic delay grows more than linearly with the number of inputs in a real NAND or NOR circuit. For example, Figure 4.23 shows a model of an *n*-input NAND gate in which the upper inputs were all 1 and the bottom input rises. The gate must discharge the diffusion capacitances of all of the internal nodes as well as the output. The Elmore delay is

$$t_{pd} = R(3nC) + \sum_{i=1}^{n-1}\left(\frac{iR}{n}\right)(nC) = \left(\frac{n^2}{2} + \frac{5}{2}n\right)RC \qquad \textbf{(4.23)}$$

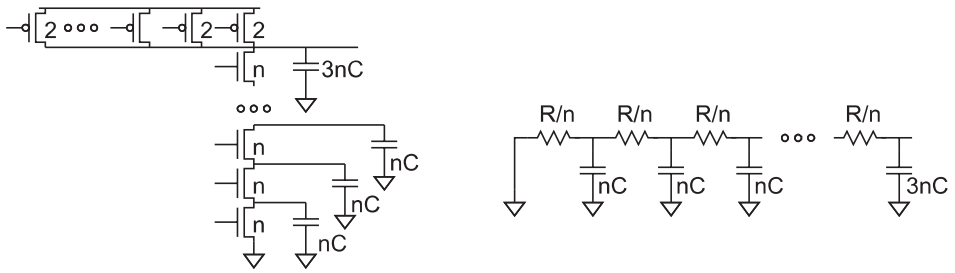

FIGURE 4.23 *n*-input NAND gate parasitic delay

This delay grows quadratically with the number of series transistors n, indicating that beyond a certain point it is faster to split a large gate into a cascade of two smaller gates. We will see in Section 4.4.6.5 that the coefficient of the n^2 term tends to be even larger in real circuits than in this simple model because of gate-source capacitance. In practice, it is rarely advisable to construct a gate with more than four or possibly five series transistors. When building large fan-in gates, trees of NAND gates are better than NOR gates because the NANDs have lower logical effort.

4.4.3 Delay in a Logic Gate

Consider two examples of applying the linear delay model to logic gates.

Example 4.10

Use the linear delay model to estimate the delay of the fanout-of-4 (FO4) inverter from Example 4.6. Assume the inverter is constructed in a 65 nm process with $\tau = 3$ ps.

SOLUTION: The logical effort of the inverter is $g = 1$, by definition. The electrical effort is 4 because the load is four gates of equal size. The parasitic delay of an inverter is $p_{inv} \approx 1$. The total delay is $d = gh + p = 1 \times 4 + 1 = 5$ in normalized terms, or $t_{pd} = 15$ ps in absolute terms.

Often path delays are expressed in terms of FO4 inverter delays. While not all designers are familiar with the τ notation, most experienced designers do know the delay of a fanout-of-4 inverter in the process in which they are working. τ can be estimated as 0.2 FO4 inverter delays. Even if the ratio of diffusion capacitance to gate capacitance changes so $p_{inv} = 0.8$ or 1.2 rather than 1, the FO4 inverter delay only varies from 4.8 to 5.2. Hence, the delay of a gate-dominated logic block expressed in terms of FO4 inverters remains relatively constant from one process to another even if the diffusion capacitance does not.

As a rough rule of thumb, the FO4 delay for a process (in picoseconds) is 1/3 to 1/2 of the drawn channel length (in nanometers). For example, a 65 nm process with a 50 nm channel length may have an FO4 delay of 16–25 ps. Delay is highly sensitive to process, voltage, and temperature variations, as will be examined in Section 7.2. The FO4 delay is usually quoted assuming typical process parameters and worst-case environment (low power supply voltage and high temperature).

Example 4.11

A ring oscillator is constructed from an odd number of inverters, as shown in Figure 4.24. Estimate the frequency of an N-stage ring oscillator.

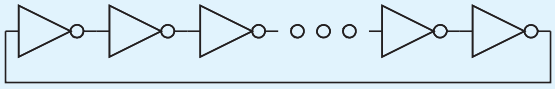

FIGURE 4.24 Ring oscillator

SOLUTION: The logical effort of the inverter is $g = 1$, by definition. The electrical effort of each inverter is also 1 because it drives a single identical load. The parasitic delay is also 1. The delay of each stage is $d = gh + p = 1 \times 1 + 1 = 2$. An N-stage ring oscillator

has a period of $2N$ stage delays because a value must propagate twice around the ring to regain the original polarity. Therefore, the period is $T = 2 \times 2N$. The frequency is the reciprocal of the period, $1/4N$.

A 31-stage ring oscillator in a 65 nm process has a frequency of $1/(4 \times 31 \times 3 \text{ ps}) = 2.7$ GHz.

Note that ring oscillators are often used as process monitors to judge if a particular chip is faster or slower than nominally expected. One of the inverters should be replaced with a NAND gate to turn the ring off when not in use. The output can be routed to an external pad, possibly through a test multiplexer. The oscillation frequency should be low enough (e.g., 100 MHz) that the path to the outside world does not attenuate the signal too badly.

4.4.4 Drive

A good standard cell library contains multiple sizes of each common gate. The sizes are typically labeled with their drive. For example, a unit inverter may be called inv_1x. An inverter of eight times unit size is called inv_8x. A 2-input NAND that delivers the same current as the inverter is called nand2_1x.

It is often more intuitive to characterize gates by their drive, x, rather than their input capacitance. If we redefine a unit inverter to have one unit of input capacitance, then the drive of an arbitrary gate is

$$x = \frac{C_{\text{in}}}{g} \tag{4.24}$$

Delay can be expressed in terms of drive as

$$d = \frac{C_{\text{out}}}{x} + p \tag{4.25}$$

4.4.5 Extracting Logical Effort from Datasheets

When using a standard cell library, you can often extract logical effort of gates directly from the datasheets. For example, Figure 4.25 shows the INV and NAND2 datasheets from the Artisan Components library for the TSMC 180 nm process. The gates in the library come in various drive strengths. INVX1 is the unit inverter; INVX2 has twice the drive. INVXL has the same area as the unit inverter but uses smaller transistors to reduce power consumption on noncritical paths. The X12–X20 inverters are built from three stages of smaller inverters to give high drive strength and low input capacitance at the expense of greater parasitic delay.

From the datasheet, we see the unit inverter has an input capacitance of 3.6 fF. The rising and falling delays are specified separately. We will develop a notation for different delays in Section 9.2.1.5, but will use the average delay for now. The average *intrinsic* or parasitic delay is (25.3 + 14.6)/2 = 20.0 ps. The slope of the delay vs. load capacitance curve is the average of the rising and falling K_{load} values. An inverter driving a fanout of h will thus have a delay of

$$t_{pd} = 20.0 \text{ ps} + \left(3.6 \frac{\text{fF}}{\text{gate}}\right)(h \text{ gates})\left(\frac{4.53 + 2.37}{2} \frac{\text{ns}}{\text{pF}}\right) = (20.0 + 12.4h) \text{ ps} \tag{4.26}$$