CHAPTER
6

High-Speed CMOS Logic Design

CHAPTER OUTLINE

6.1 Introduction

This chapter addresses the issues of high-speed logic design in CMOS technology. When designing any logic circuit, we seek to find a combination of gates and gate sizes that perform the desired function and satisfy the timing requirements. The correct functionality is obtained by the proper selection of logic gates, while the timing requirements are satisfied by proper gate sizing. Often, we try to design a logic circuit to run as fast as possible so that the clock cycle can be minimized. In the process of optimizing the logic circuit, we will encounter logic paths that have the longest delays from input to output. These are the so-called *critical paths*. If we reduce the delays along the critical paths, the worst-case delay is reduced, and the speed of the circuit is increased. The delay of each gate is controlled by its driving resistance and the load capacitance. In this chapter, we first focus on detailed delay calculation for logic gates and then turn our attention to the optimization of critical paths. Our overall goal is to maximize the speed of a circuit, while minimizing the area and power dissipation.

In the last chapter, we used a simple switching delay model to compute device sizes for inverters. Here we examine detailed calculations for the switching delay of

a logic gate driving a load capacitance, C_L. The currents available to charge and discharge C_L are the drain currents of the driver and load transistors. These drain currents are a function of both V_{in} and V_{out}. Accurate simulation results, in which V_{in} and V_{out} are both changing with time while satisfying the nonlinear dc device equations, can be obtained point by point in the time domain using SPICE or a similar program. However, we would like to be able to analyze circuits quickly using "back-of-the-envelope" hand calculations. With suitable simplifying approximations, a first-order hand analysis is possible. Simplified analyses of this sort are helpful in developing insight into circuit performance and in making the most effective use of subsequent computer simulations.

For hand analysis of a transient circuit response, we first approximate the input waveform by a step function. In this approximation, the input step is assumed to occur at the 50% point of the actual input waveform. Later we will remove this assumption and compute delays when the input is a ramp function. For now, assume an ideal step function as the input.

When we discuss the delay of a path through a logic circuit, we need to define the precise meaning of delay. In Figure 6.1a, a series of three gates are connected to form a logic path from input to output. The switching delay from input to output is referred to as the *propagation delay.* This notion of propagation delay can be defined in a variety of different ways. In all cases, we have to define reference points along the waveform transition from high to low or low to high at which the delay is measured.

One possible reference point is the switching threshold, V_S, from the voltage transfer characteristics, as shown in Figure 6.1b. This is the most sensible definition since the input and output are at the same voltage by the definition of V_S. However, the switching threshold varies from one gate to another and depends on which

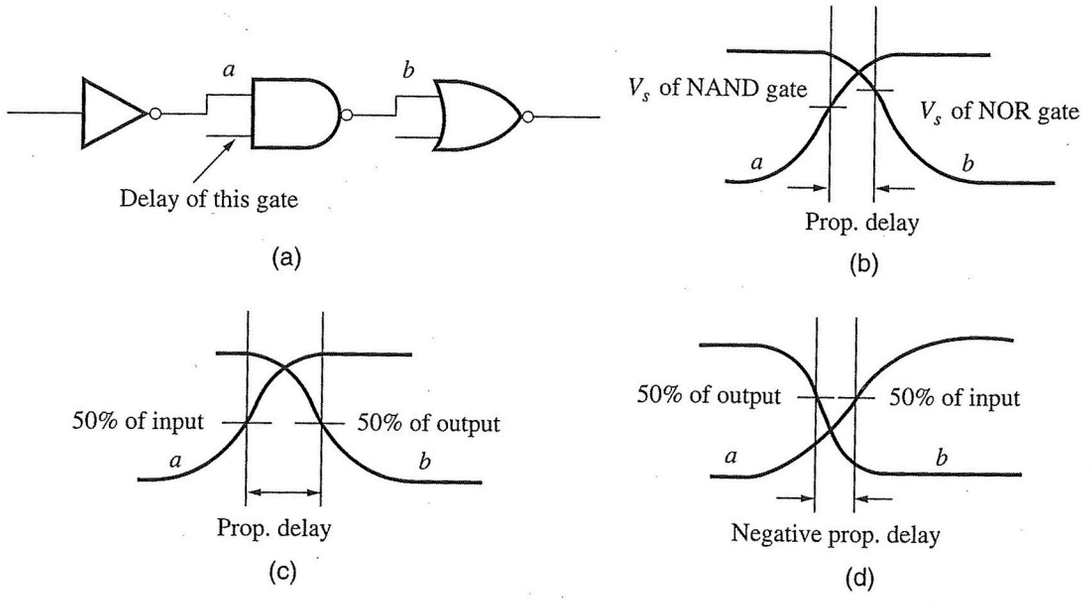


Figure 6.1
Definitions of propagation delay.

input switches. It would take some effort to compute this value for each gate in the logic path before we perform a delay calculation. Therefore, it is not a practical reference point for hand calculation.

Another approach is to use the 50% point of the input and output, as shown in Figure 6.1c. This is a reasonable definition for the propagation delay and is independent of the gate type. In fact, most signals have a switching threshold that is approximately equal to the 50% point. One problem is that it is possible for the 50% point of an output to occur before the 50% point of the input due to different rise and fall times, as shown in Figure 6.1d. Therefore, we have to be prepared to handle negative propagation delays if we use the 50% point as the switching point. Most signals have similar rise/fall times so we should not encounter negative delays very often. If a negative delay does arise, it indicates that we have a slow gate in the path and we may need to fix the design. Overall, this 50% definition is the most practical and intuitive reference point for propagation delay.

Occasionally, the rise and fall times of signals are of interest in delay calculation. This type of calculation also requires a consistent definition. If we simply use the time to transition from V_{OL} to V_{OH} or vice versa, we have a problem in defining exactly when the signal begins to switch and when it stops switching. For example, if we consider an exponent signal, such as

$$V_{\text{out}}(t) = V_{DD}e^{-t/RC}$$

we might be tempted to compute the value of t with $V_{\text{out}} = V_{OL}$. We would find that the value of t is infinite in this case—a rather impractical solution. Instead, designers use a 10% to 90% delay for rise times and 90% to 10% delay for fall times. This is shown in Figure 6.2a and Figure 6.2b for rise and fall times, respectively.

6.2 Switching Time Analysis

With these definitions in place, we now consider the switching characteristics of the CMOS inverter of Figure 6.3a. When the *input* makes a step change from V_{OH} to V_{OL}, the pull-down transistor turns off while the pull-up transistor turns on. The

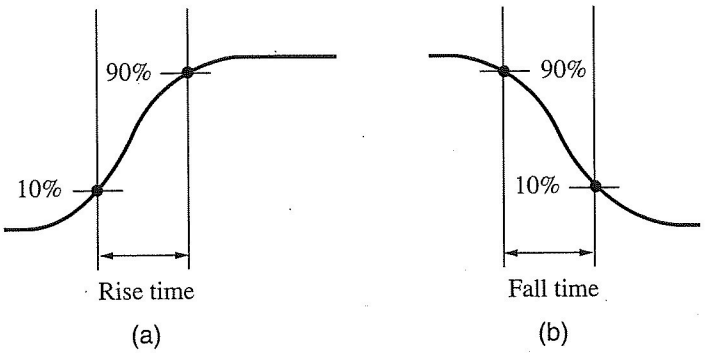


Figure 6.2
Rise and fall time definitions.

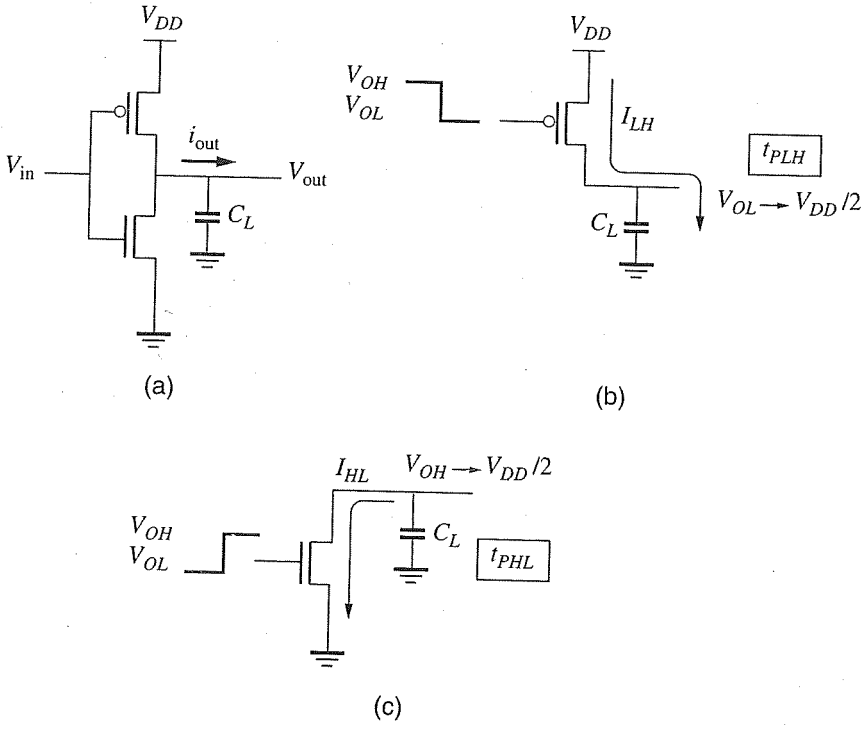


Figure 6.3
CMOS inverter delay calculation.

situation is illustrated by Figure 6.3b. The time required for V_{out} to charge from V_{OL} to the 50% point can be calculated by assuming that the lumped load capacitance is charged by the current through the pull-up device.

Similarly, when the *input* makes step transition from V_{OL} to V_{OH}, the pull-down transistor turns on while the pull-up transistor shuts off. The situation is illustrated by Figure 6.3c. The time required for V_{out} to discharge from V_{OH} to the 50% point can be calculated by assuming that the lumped load capacitance is discharged by the current through the pull-down device. In both cases, we can use the following formula for the calculation:

$$I = C_L \frac{dV}{dt} \rightarrow \Delta t \approx \frac{C_L \Delta V}{I_{DS}}$$

where C_L is the load capacitance, ΔV is the voltage change, and I_{DS} is the charging or discharging current. If the average charging current is I_{LH} and $\Delta V = V_{DD}/2$, the *low-to-high* propagation delay time is then calculated as follows:

$$t_{PLH} = \frac{C_L(V_{DD}/2)}{I_{LH}} \tag{6.1}$$

The propagation delay from *high to low* is calculated in the same manner as above with an average current of I_{HL}.

$$t_{PHL} = \frac{C_L(V_{DD}/2)}{I_{HL}} \tag{6.2}$$

The average propagation delay is defined as

$$t_P = \frac{t_{PLH} + t_{PHL}}{2} \tag{6.3}$$

We now compute expressions for the propagation delay for the CMOS inverter beginning with a step input from V_{OL} to V_{OH}. Using Figure 6.3 as a guide, we first compute t_{PHL} using Equation (6.2). The only unknown is I_{HL}.

To compute this value, we need to first determine the region of operation of the NMOS device, and this involves a calculation of V_{Dsat}. Since V_{Dsat} is technology-dependent, we will illustrate the steps with 0.13 μm technology parameters. Using $V_{DD} = 1.2$ V, $V_T = 0.4$ V, and $E_CL = 0.6$ V

$$V_{Dsat} = \frac{(V_{GS} - V_T)E_CL}{(V_{GS} - V_T) + E_CL} = \frac{(1.2 - 0.4)(0.6)}{(1.2 - 0.4) + (0.6)} \cong 0.34 \text{ V}$$

Since the output is making a transition from 1.2 V to 0.6 V, which implies that $V_{DS} > V_{Dsat}$ at all times, the NMOS device remains in saturation for the entire duration of interest for t_{PHL}. Therefore, the average current is simply the saturation current for the n-channel device, $(I_{Dsat})_n$.

$$t_{PHL} = \frac{C_L(V_{DD}/2)}{(I_{Dsat})_n} \tag{6.4a}$$

In the previous chapter, we found that $t_{PHL} = 0.7R_NC_L$. If we compare this formula to (6.4a), we see that

$$R_N = \frac{V_{DD}/2}{0.7(I_{Dsat})_n} \tag{6.4b}$$

Next, we compute t_{PLH} using Equation (6.1). The only unknown is I_{LH} due to the PMOS device. For the p-channel pull-up device, the saturation voltage is given by

$$V_{Dsat} = \frac{(1.2 - 0.4)2.4}{(1.2 - 0.4) + 2.4} = 0.6 \text{ V}$$

The PMOS device is also in saturation during the transition. Based on this, we can use the equation

$$t_{PLH} = \frac{C_L(V_{DD}/2)}{(I_{Dsat})_p} \tag{6.5a}$$

If we compare this formula to the one used in the previous chapter, we see that

$$R_P = \frac{V_{DD}/2}{0.7(I_{Dsat})_p} \tag{6.5b}$$

By considering unit-size devices, we can compute R_{eqn} and R_{eqp} from Equations (6.4b) and (6.5b).

Example 6.1 **Hand Calculation of Effective Resistance**

Problem:

Using 0.13 μm technology parameters, compute R_{eqn} and R_{eqp} from the equations above for unit-sized devices.

Solution:

For the NMOS device,

$$I_{Dsat} = \frac{W_N v_{sat} C_{ox}(V_{DD} - V_{TN})^2}{(V_{DD} - V_{TN}) + E_{CN}L_N}$$

$$= \frac{(0.1)(10^{-4})8(10^6)1.6(10^{-6})(1.2 - 0.4)^2}{(1.2 - 0.4) + 0.6} \approx 60\ \mu A$$

$$\therefore R_{eqn} = \frac{1.2/2}{0.7(60\ \mu A)} = 14.5\ k\Omega$$

For the PMOS device,

$$I_{Dsat} = \frac{W_P v_{sat} C_{ox}(V_{DD} - |V_{TP}|)^2}{(V_{DD} - |V_{TP}|) + E_{CP}L_P}$$

$$= \frac{(0.1)(10^{-4})8(10^6)1.6(10^{-6})(1.2 - 0.4)^2}{(1.2 - 0.4) + 2.4} \approx 25\ \mu A$$

$$\therefore R_{eqp} = \frac{1.2/2}{0.7(25\ \mu A)} = 33.5\ k\Omega$$

It is interesting to compare the above results with SPICE simulations. In fact, we have been using the results from SPICE in all timing calculations thus far

$$R_{eqn} \approx 12.5\ k\Omega/\square$$

$$R_{eqp} \approx 30\ k\Omega/\square$$

The hand calculations and SPICE are in close agreement implying that our assumptions are all basically correct. The actual currents in SPICE are somewhat higher than our hand calculations producing smaller values of R_{eq}.

From these values, we can compute R_N and R_P by dividing them by the W/L of each device:

$$R_N = R_{eqn}\left(\frac{L}{W}\right)_n$$
$$R_P = R_{eqp}\left(\frac{L}{W}\right)_p \tag{6.6}$$

We will find that R_P and R_N work well for timing calculation, but these values *should not* be used for any other purpose. These values are obtained from the timing equations and therefore are only expected to work in the timing applications. In reality, the on-resistance is nonlinear and its value depends on the applied voltages.

6.2.1 Gate Sizing Revisited—Velocity Saturation Effects

Since we are already discussing the issues of proper sizing, we should go back and examine the CMOS NAND and NOR sizing operations. Consider the three standard gates shown in Figure 6.4. The inverter has a pull-up device of $2W$ and a pull-down device of W. The NAND gate has two pull-ups that are $2W$ each and two pull-downs that are $2W$ each. The NOR gate has two pull-ups that are $4W$ each and two pull-downs that are W each. These sizes are accurate for the quadratic device model, but do not incorporate the effects of velocity saturation. Some adjustments are now introduced to include velocity saturation.

To understand the effect, we compare a single device with a pair of stacked devices, as shown in Figure 6.5. Ignoring all capacitances except C_L for the moment, we would find that the single device of size W in Figure 6.5a takes longer to discharge the load capacitance than two series stacked $2W$ devices of Figure 6.5b. The reason for this can be explained based on the region of operation of the transistors in the two cases. The single M_0 device is in saturation for the entire transition of the output from V_{DD} to $V_{DD}/2$ and delivers a current of I_0. The two series devices, M_1 and M_2, each operate in different regions during the transition from V_{DD} to $V_{DD}/2$. We know that both conduct the same amount of current. To satisfy this requirement, M_1

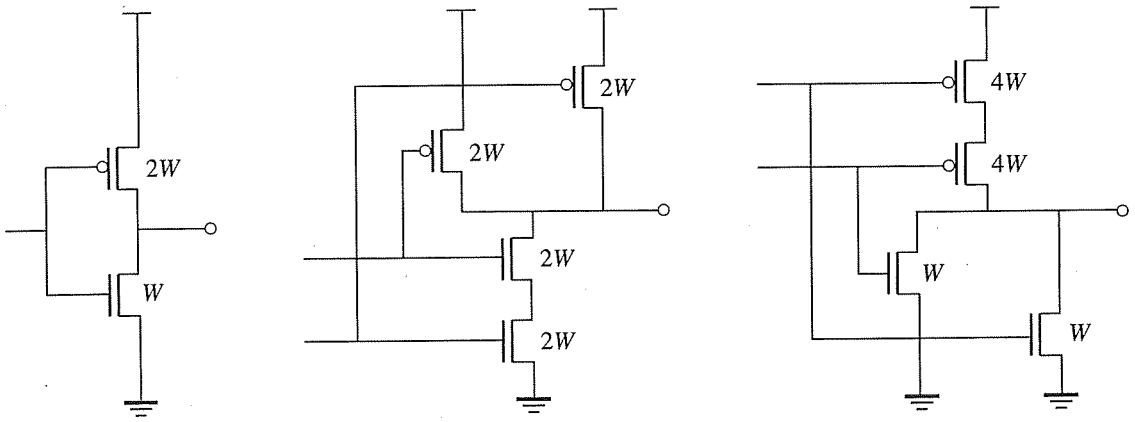


Figure 6.4
CMOS gate sizing without including velocity saturation.

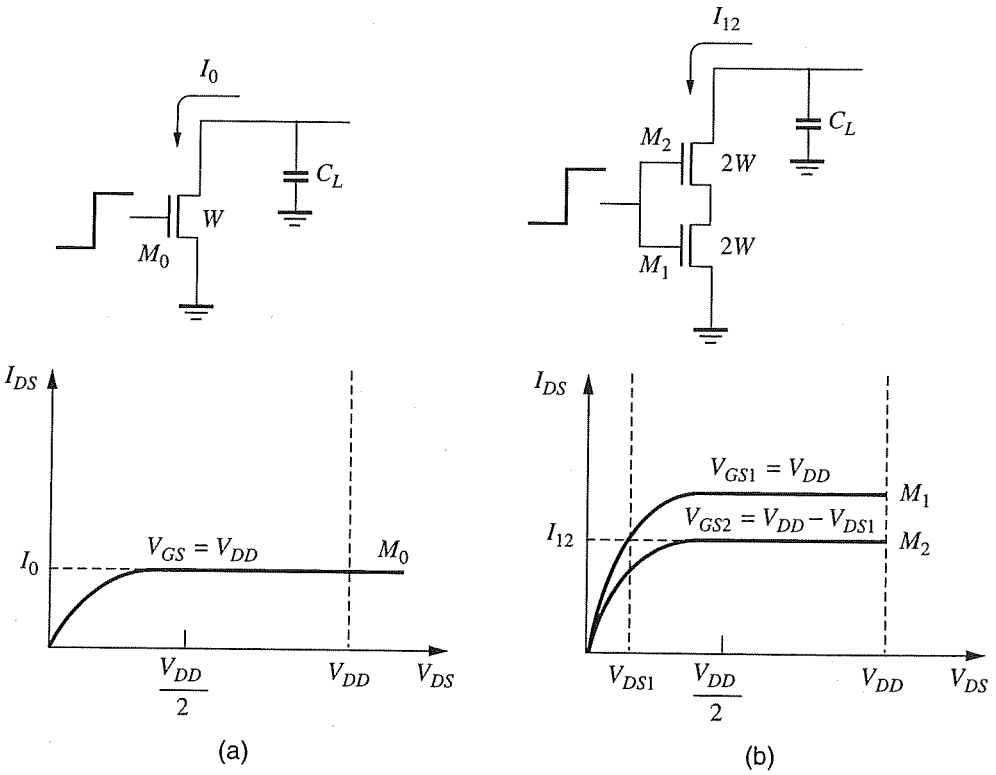


Figure 6.5
Stacked devices with velocity saturation.

is forced into the linear region with a smaller V_{DS1} while M_2 operates in the saturation region with a larger V_{DS2}.

When these operating conditions are established, the resulting current through both devices is equal to I_{12}, the saturation region current of M_2. This is illustrated in Figure 6.5b. The upper curve is for M_1 with $V_{GS1} = V_{DD}$, and the lower curve is for M_2 with $V_{GS2} = V_{DD} - V_{DS1}$. M_2 sets the current value based on its values of V_{GS} and V_{DS}. The current is maintained at this level until the switching point. If we compare the currents in the two cases, we would find that $I_{12} > I_0$. Even though M_2 has a smaller V_{GS} and V_{DS} than M_0, its width of $2W$ makes I_{12} about 20–25% larger than I_0. Therefore, the discharging time is smaller for the stacked short-channel devices. To equalize the delays the stacked devices can be reduced by roughly 20–25%.[1]

If we had a pair of long channel devices in the series stack, the discharge current would actually be lower than the single transistor case. The reason for this is due to the quadratic dependence of current on $V_{GS} - V_T$. That is, as V_{GS} is reduced for M_2, the current drops off quadratically. For two series stacked quadratic devices, the devices must be made at least $2W$ to deliver the same current as a single W size device.[2] The current does not drop off as rapidly for velocity saturated devices due

[1] SPICE simulations can be used to determine the exact scale factor.
[2] When body-effect is included, the size must be slightly larger than $2W$.

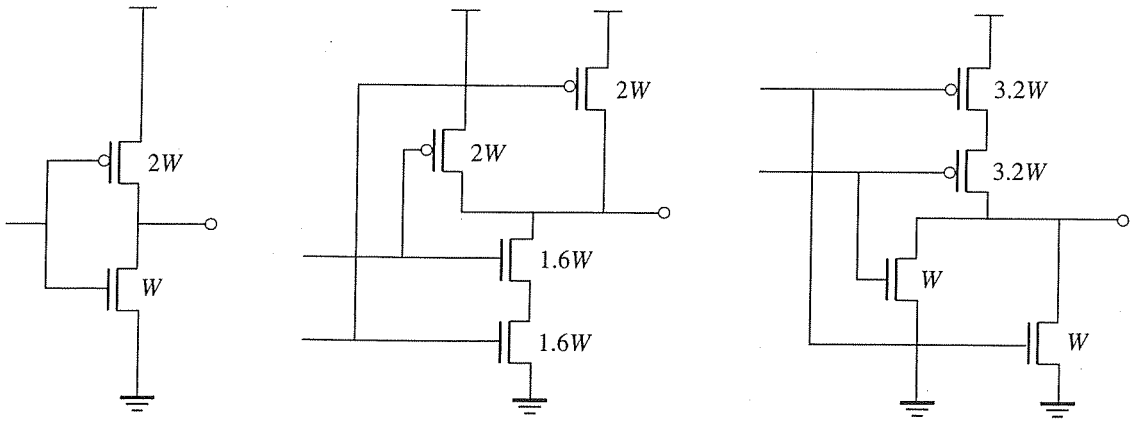


Figure 6.6
Transistor sizing for NAND and NOR including velocity saturation.

to a more linear dependence of current on $V_{GS} - V_T$. As a result, they are able to deliver a larger current.

The NAND2 and NOR2 gates can be resized including this effect, as shown in Figure 6.6. Here we have assumed that stacked devices must be scaled by 0.8 to produce the same current as a single device. The NAND gate pull-downs are resized to $1.6W$ to account for the velocity saturation effects. Similarly, the NOR gate pull-ups are resized to $3.2W$ to account for velocity saturation. These values will produce rise and fall delays that are equal to the reference inverter, assuming that the loading capacitance is a fixed value C_L. The correct values for a particular technology can be derived from SPICE simulation.

In general, there are many additional factors that control the final device sizes. For example, body-effect tends to reduce the current of the top device in the stack so a slightly larger device size is preferable as the number of transistors in the stack increases. There are also additional self-capacitances in NAND and NOR gates that must be charged and discharged. Furthermore, device sizes determine noise margins, and rise and fall times. When all factors are considered, the original sizes in Figure 6.4 provide satisfactory results. Therefore, we will continue to use the sizes shown in Figure 6.4 when sizing devices, although we now know how to compute more accurate values if needed.

6.3 Detailed Load Capacitance Calculation

So far, we have assumed a fixed external loading on the output of logic gates. The load capacitance is actually comprised of three components as shown in Figure 6.7: the self-loading capacitance, the interconnect or wire capacitance, and the fanout capacitance. For delay calculation, we sum these individual quantities to obtain a lumped capacitance:

$$C_{\text{load}} = C_{\text{self}} + C_{\text{wire}} + C_{\text{fanout}} \tag{6.7}$$

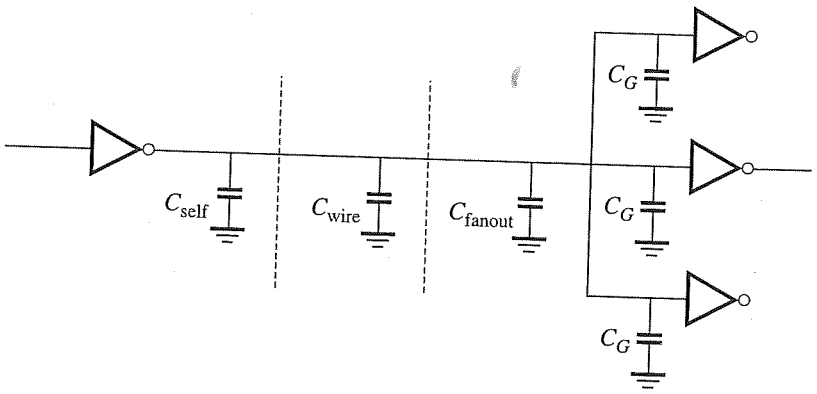


Figure 6.7
Components of the loading capacitance on a gate.

Although each of these components is somewhat complicated, our goal is to quickly compute the load capacitance using simplified equations.

6.3.1 Fanout Gate Capacitance

The first type of loading capacitance to consider is the fanout capacitance due to the inputs of subsequent gates, C_G. This capacitance can be large, depending on the number of fanouts being driven by the gate. The total fanout capacitance is the sum of each of the gate capacitances, as shown in Figure 6.7:

$$C_{\text{fanout}} = \Sigma C_G$$

The capacitances for each fanout, assuming that they are all inverters, are illustrated in Figure 6.8. We are interested in the specific terms associated with V_{in} since we are driving this input. Each transistor has a term due to the thin-oxide and two terms due to the overlap capacitance. The capacitances that must be taken into account are: C_{Gn}, C_{Gp}, and C_{OL}.

From Chapter 2, we know that the thin-oxide capacitance is voltage dependent, but since we are driving the transistor from the gate node, we can use $C_{ox}WL$ to compute its worst-case value. The total input capacitance for the inverter is the sum of all the components in Figure 6.8:

$$\begin{aligned} C_G &= C_{Gn} + 2C_{OL} + C_{Gp} + 2C_{OL} \\ &= C_{ox}LW_n + 2C_{ol}W_n + C_{ox}LW_p + 2C_{ol}W_p \\ &= (C_{ox}L + 2C_{ol})(W_n + W_p) \end{aligned} \tag{6.8}$$

The leading term in this expression is technology dependent and can be pre-computed. For 0.13 μm technology, the value of $C_{ox}L = 1.6 \times 10^{-6}$ F/cm^2 $\times$

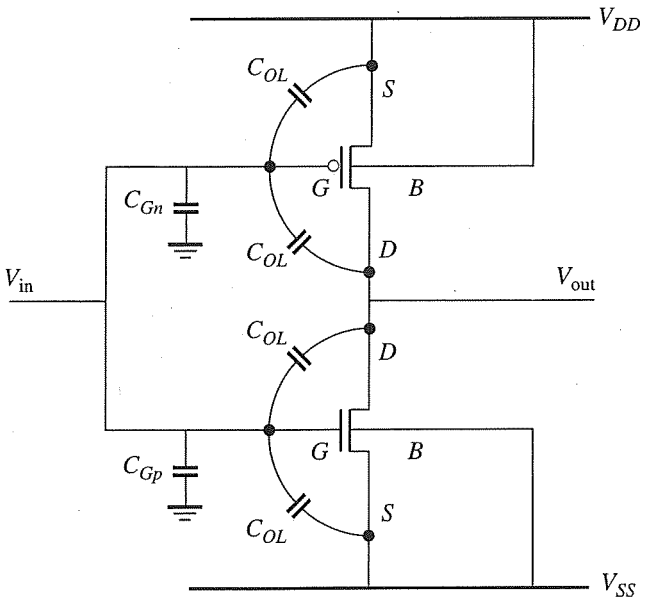


Figure 6.8
Input capacitance calculation.

0.1 μm = 1.6 fF/μm. In Chapter 2, we approximated the overlap capacitance as C_{ol} = 0.25 fF/μm. Therefore, we now redefine C_g as

$$C_g = C_{ox}L + 2C_{ol} = 1.6 \text{ fF}/\mu\text{m} + 2(0.25 \text{ fF}/\mu\text{m}) \cong 2 \text{ fF}/\mu\text{m} \quad (6.9)$$

The total capacitance due to the thin-oxide and the overlap capacitance is roughly 2 fF/μm. This value has been almost constant for over 20 years. We will use 2 fF/μm for the gate capacitance and multiply it by the width of the devices to obtain the total capacitance.

For an inverter

$$C_G = C_gW = 2 \text{ fF}/\mu\text{m} \times W_n + 2 \text{ fF}/\mu\text{m} \times W_p$$
$$= 2 \text{ fF}/\mu\text{m}(W_n + W_p)$$

If we are driving n identical gates with a capacitance of C_G each, then the total fanout capacitance is

$$C_{\text{fanout}} = \Sigma C_G = n \times C_G = n \times \Sigma C_gW$$

If we had n different inverters, then we could just add them up as follows:

$$\therefore \Sigma C_G = 2\frac{\text{fF}}{\mu\text{m}} \times (\underbrace{W_{p1} + W_{n1}}_{\text{first inverter}} + \underbrace{W_{p2} + W_{n2}}_{\text{second inverter}} + \cdots)$$

For NANDs, NORs, or other complex gates, we simply apply the above equation with the appropriate widths for the transistors being driven:

$$\therefore \Sigma C_G = 2\frac{\text{fF}}{\mu\text{m}} \times (\underbrace{W_{p1} + W_{n1}}_{\text{first gate}} + \underbrace{W_{p2} + W_{n2}}_{\text{second gate}} + \cdots)_{\text{etc.}}$$

6.3.2 Self-Capacitance Calculation

The self-capacitance is the sum of the capacitances connected to the output, V_{out}. We examine the inverter in Figure 6.9 to determine the important terms to include in the self-capacitance.

There are four main capacitances for each transistor: C_{GS}, C_{GD}, C_{DB}, and C_{SB}. We can immediately eliminate four capacitances from consideration: C_{GSn}, C_{GSp}, C_{SBn}, and C_{SBp}. These are not connected to the output. If we now consider a step input in either direction, one transistor is always off while the other one is in saturation. Since our calculation for t_{PHL} or t_{PLH} involves a transition to the 50% point, we can assume that the transistors are in saturation or cutoff. In either region, the gate-to-drain capacitance, C_{GD}, is negligible. This leaves only the overlap capacitances from gate-to-drain, and one junction capacitance per device, C_{DBn} and C_{DBp}.

The overlap capacitance must be handled in a special manner since it is connected from the input to output. When the input makes a transition from 0 to V_{DD}, the output makes a transition from V_{DD} to 0. As a result, the overlap capacitance experiences a voltage swing of $2V_{DD}$. We can model this by assuming that the swing is only V_{DD} and then doubling the size of the capacitance. This process of doubling

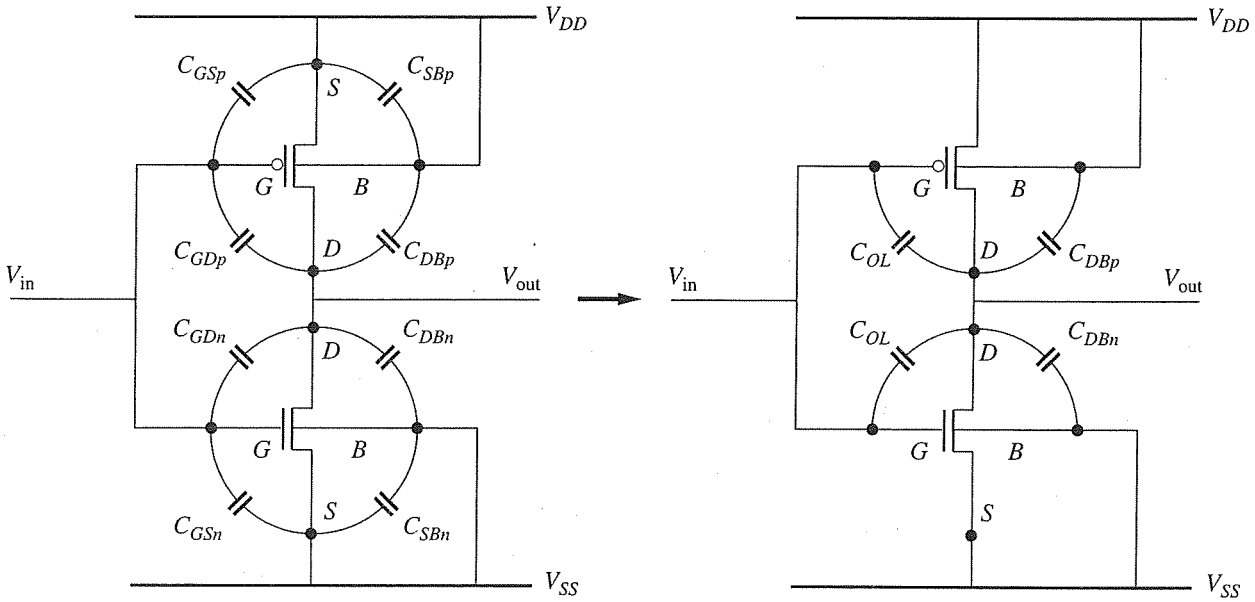


Figure 6.9
Output capacitance calculation.

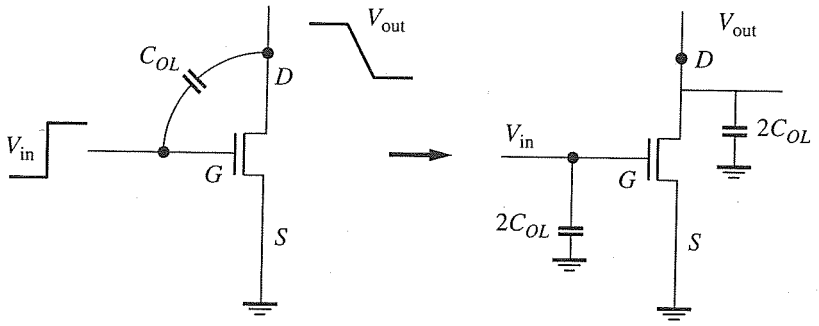


Figure 6.10
Handling the overlap capacitance using Miller Effect.

the capacitance value, commonly referred to as *Miller Effect* modeling, is shown in Figure 6.10. It implies that the output must deliver twice as much charge to the overlap capacitance to account for the fact that the input and output are switching by the same amount in opposite directions.

We can now compute the total self-capacitance of the inverter. It is simply the junction capacitances plus the overlap capacitances due to fringing and lateral diffusion (with the inclusion of Miller Effect):

$$\begin{aligned} C_{\text{self}} &= C_{DBn} + C_{DBp} + 2C_{OL} + 2C_{OL} \\ &= C_{jn}W_n + C_{jp}W_p + 2C_{ol}(W_n + W_p) \\ &= C_{eff}(W_n + W_p) \end{aligned} \tag{6.10}$$

Here, we relate the capacitance to an effective capacitance per width, C_{eff}. To simplify the calculations, we have combined C_{jn} and C_{jp} into one term. As we saw in Chapter 2, the junction capacitance depends on the doping levels, the type of junction, the voltage transition, and the areas of the source and drain regions. However, for hand calculation purposes, it is convenient to combine them. The average junction capacitance for the two junctions is approximately 0.5 fF/μm for 0.13 μm technology. The overlap capacitance is roughly 0.25 fF/μm. Therefore,

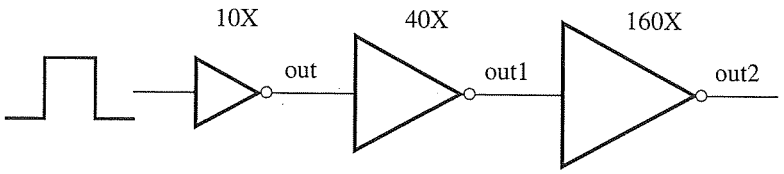
$$C_{eff} = C_j + 2C_{ol} \approx 0.5 \text{ fF}/\mu\text{m} + 2(0.25 \text{ fF}/\mu\text{m}) \approx 1 \text{ fF}/\mu\text{m}$$

The total self-capacitance is computed by multiplying C_{eff} by the device width, W.

Using SPICE to Determine C_g and C_{eff} in 0.18 μm Technology **Example 6.2**

Problem:

Find C_g and C_{eff} for an 0.18 μm technology from SPICE using the following circuit. The 10X inverter drives a 40X inverter which drives a 160X inverter. Apply positive and negative step inputs and measure the propagation delays at node *out*. Devise a way of extracting the two desired capacitance parameters.



Solution:

Run two simulations. In the first case, set all area and perimeter values to zero to eliminate the junction capacitances from consideration. Measure the propagation delay for both the rising and falling cases and extract C_g. Next, set all the area and perimeter values to their proper quantities and measure the propagation delays again. The increase in delay is due to junction capacitances so the value of C_{eff} can be extracted.

```
*Cg Calculation (no area or perimeter specified)
.param      Supply=1.8                *Set value of Vdd
.lib        'bsim3v3.cmosp18.lib'     *Set 0.18um library
.opt        scale=0.1u                *Set lambda
.global vdd    gnd
vdd     vdd       gnd 1.8
m1      out in1 vdd vdd          pch   l=2 w=40  ad=0   as=0 pd=0   ps=0
m2      out in1 0 0              nch   l=2 w=20  ad=0   as=0 pd=0   ps=0
m3      out1 out vdd vdd         pch   l=2 w=160 ad=0   as=0 pd=0   ps=0
m4      out1 out 0 0             nch   l=2 w=80  ad=0   as=0 pd=0   ps=0
m5      out2 out1 vdd vdd        pch   l=2 w=640 ad=0   as=0 pd=0   ps=0
m6      out2 out1 0 0            nch   l=2 w=320 ad=0   as=0 pd=0   ps=0
vin1     in1       gnd  pwl (0 0 9ps 0 10ps 1.8 199ps 1.8 200ps 0)
.tran 5ps 350ps
.print tran v(in1) v(out)
.end
```

The propagation delay for the falling case is 40ps and the rising case is 55ps. We can use the delay equation to solve for C_g for these two cases.

$$t_{PHL} = 0.7\, R_N C_{\text{fanout}}$$

$$\therefore 40\text{ ps} = 0.7\,\frac{(12.5\text{ k}\Omega)}{10}\,C_g(8\ \mu\text{m} + 16\ \mu\text{m})$$

$$\therefore C_g \cong 1.9\text{ fF}/\mu\text{m}$$

$$t_{PLH} = 0.7\, R_P C_{\text{fanout}}$$

$$\therefore 55\text{ ps} = 0.7\,\frac{(30\text{ k}\Omega)}{20}\,C_g(8\ \mu\text{m} + 16\ \mu\text{m})$$

$$\therefore C_g \cong 2.2\text{ fF}/\mu\text{m}$$

The results show that the gate capacitance is slightly higher for the low to high transition. An average value of $C_g = 2$ fF/μm is obtained, as expected. Next we run a simulation with the areas and perimeters specified and compute the value of C_{eff} based on the increase in the propagation delays relative to the previous SPICE runs.

```
*Ceff Calculation (areas and perimeters specified)
.param      Supply=1.8          *Set value of Vdd
.lib        'bsim3v3.cmosp18.lib'     *Set 0.18um library
.opt        scale=0.1u          *Set lambda
.global vdd gnd
vdd     vdd  gnd 1.8
m1      out in1 vdd vdd    pch  l=2 w=40 ad=200 as=200 pd=40 ps=40
m2      out in1 0 0        nch  l=2 w=20 ad=100 as=100 pd=20 ps=20
m3      out1 out vdd vdd   pch  l=2 w=160 ad=800 as=800 pd=160 ps=160
m4      out1 out 0 0       nch  l=2 w=80 ad=200 as=200 pd=80 ps=80
m5      out2 out1 vdd vdd  pch  l=2 w=640 ad=3200 as=3200 pd=640 ps=640
m6      out2 out1 0 0      nch  l=2 w=320 ad=1600 as=1600 pd=320 ps=320
vin1     in1       gnd  pwl (0 0 9ps 0 10ps 1.8 199ps 1.8 200ps 0)
.tran 5ps 350ps
.print tran v(in1) v(out)
.end
```

The propagation delay for the falling case is 44 ps and the rising case is 59 ps. We can use the same delay equations to solve for C_{eff} for these two cases.

$$t_{PHL} = 0.7\, R_N C_{\text{fanout}}$$

$$\therefore 44 \text{ ps} = 0.7 \frac{(12.5\ \text{k}\Omega)}{10} (C_g(24\ \mu\text{m}) + C_{eff}(6\ \mu\text{m}))$$

$$\therefore C_{eff} \cong 0.8\ \text{fF}/\mu\text{m}$$

$$t_{PLH} = 0.7\, R_P C_{\text{fanout}}$$

$$\therefore 59 \text{ ps} = 0.7 \frac{(30\ \text{k}\Omega)}{20} (C_g(24\ \mu\text{m}) + C_{eff}(6\ \mu\text{m}))$$

$$\therefore C_{eff} \cong 0.7\ \text{fF}/\mu\text{m}$$

The results are also close to the values computed by hand calculations. It is acceptable to use $C_{eff} = 1$ fF/μm for hand calculations since it slightly over-estimates the capacitance. In fact, when using ramp inputs, the output capacitance increases slightly due to a Miller Effect on C_{gd} of any device in the linear region of operation. We will find that 1 fF/μm is a more suitable value.

We now extend our analysis to CMOS NAND and NOR gates. Since the considerations for NANDs and NORs are similar in nature, we will illustrate the process with the NOR gate in Figure 6.11. A symbolic layout of the NOR gate is shown in

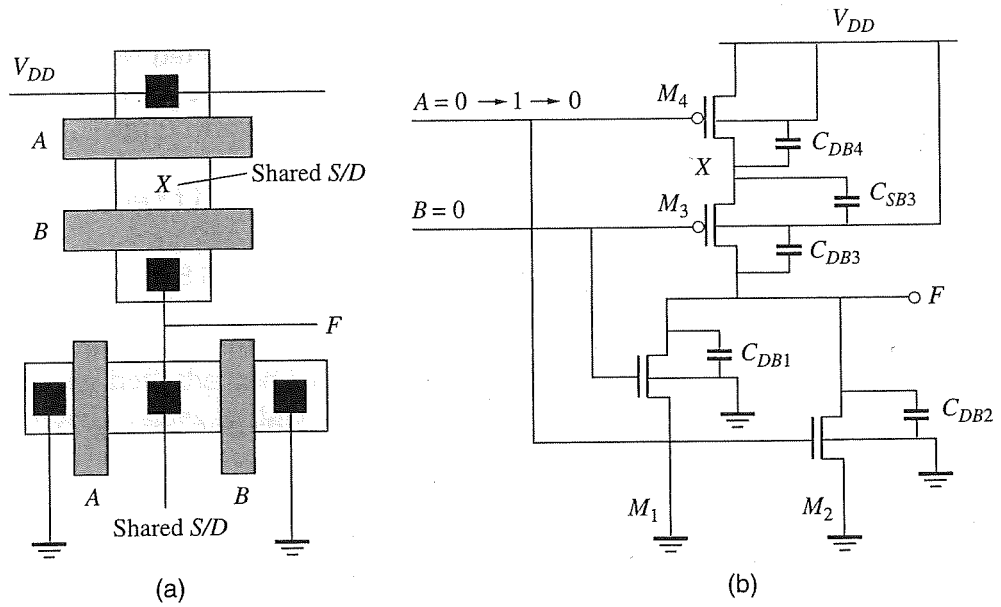


Figure 6.11
Self-capacitances for a NOR gate.

Figure 6.11a and the corresponding transistor schematic in Figure 6.11b. To compute the worst-case self-capacitance, consider the pull-down case first with a propagation delay of t_{PHL}. The output node must be discharged to Gnd by one of the pull-down devices. In the worst-case, input A switches from low to high while B remains low since the capacitances at both the output node and the internal node X must be discharged. Assuming a step change at input A, the capacitance at the output node is comprised of the indicated junction capacitances of the n-channel and p-channel devices. Therefore,

$$\begin{aligned} C_{\text{self}} &= \underbrace{C_{DB1} + C_{DB2}}_{n^+\text{shared } S/D} + C_{DB3} + \underbrace{C_{SB3} + C_{DB4}}_{p^+\text{shared } S/D} \\ &= C_{DB12} + C_{DB3} + C_{SDB34} \end{aligned} \tag{6.11}$$

Note that the two n-channel devices have a single shared source/drain region at node F, which we have called C_{DB12}, and the two p-channel devices have a shared drain region at node X, referred to as C_{SDB34}. We should not "double count" these junction capacitances. The layout of a gate will dictate whether source/drain sharing must be incorporated into the calculations. Therefore, a sketch of the layout is useful when computing self-capacitance.

Note also that if input A stays low but B goes high, the only output capacitances to be charged are $C_{DB12} + C_{DB3}$. This will lead to a faster switching time. However, we are not interested in this case since it is not the worst case.

We now consider the analysis for the charging from low to high at the output. If we assume that input A switches from high to low while input B remains low,

then the total capacitance to be charged is given by Equation (6.11). However, if B switches rather than A, only the capacitances connected to F are charged up. Therefore, this is the best-case situation and of less interest for delay calculation. A similar kind of analysis can be carried out for the NAND gate and is left as an exercise for the reader.

Exercise 6.1

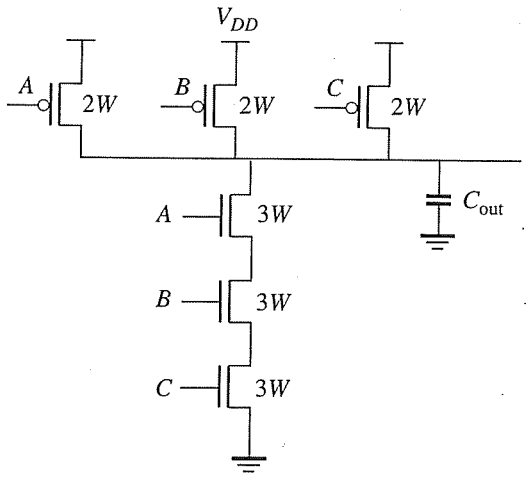
Draw a simplified layout of a 2-input NAND gate and determine the worst-case capacitance components at the output assuming a step input. Specify the input combinations that create the worst-case scenarios.

Example 6.3

Capacitance Calculation for 3-input NAND Gate

Problem:

For the NAND3 gate below, determine the worst-case capacitance components at the input and output assuming a step input. Express the result in terms of W, C_g, and C_{eff}. Consider the source/drain sharing carefully.



Solution:

The worst-case input capacitance is simply: $C_{in} = C_g(W_n + W_p) = C_g(3W + 2W) = C_g(5W)$.

The output capacitance is the self-capacitance. The n-channel transistors share source/drain regions at their intermediate nodes. Two of the three p-channel devices also share their source/drain region. Therefore,

$$\begin{aligned} C_{load} &= C_{eff}(W_n + W_n + W_n) + C_{eff}(W_p + W_p) \\ &= C_{eff}(3W + 3W + 3W) + C_{eff}(2W + 2W) \\ &= C_{eff}(13W) \end{aligned}$$

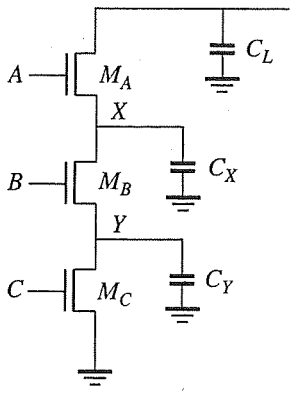


Figure 6.12
Propagation delay depends on arrival time of A, B, and C.

One aspect that is implicit in the descriptions above is that the actual delay depends on the order in which inputs switch. We illustrate the input-dependent delay concepts with the series stack of Figure 6.12. Assume that all inputs are low and the output node is high. If input A arrives first, transistor M_A will turn on and charge node X to a high value. Then, when input B arrives, transistor M_B turns on and charges up node Y to a high value. When C arrives, transistor M_C turns on and must discharge all the capacitances in Figure 6.12. If we reverse the order of arrival, then C arrives first and forces a discharge of node Y. Then B arrives and discharges node X. Finally A arrives and only the output node must be discharged.

We can make a general statement about arrival times of inputs and its effect on delay. In a series stack, the delay increases as the late arriving input is further from the output. For the three transistors shown in Figure 6.12, the C input closest to ground would cause the longest delay, assuming that it arrives last, while the A input closest to the output would result in the shortest delay. For example, if the late arriving signal is at input C, the circuit must discharge $C_L + C_X + C_Y$. If input B arrived late then it would have to discharge $C_L + C_X$ since C_Y would already be discharged by an earlier arriving signal at input C. If input A was the late arriving signal, then it would only need to discharge C_L, since C_Y and C_X would have already been discharged.

There are a number of ways to design around this delay difference. If we know which input is going to be delayed, we could make sure that it was closest to the output; that is, we would reorder the inputs such that the earliest signals arrive lower in the stack and the latest signals arrive near the top of the stack. In Figure 6.12, we hope that A arrives last and C arrives first.

Another approach is to resize the devices to accommodate the worst-case situation. To reduce delay, we should size $M_C > M_B > M_A$. Each device is progressively larger as we move from the output to ground since each one must discharge a progressively larger capacitance. The problem with this approach is that the device capacitances are increased as the device sizes are increased. The layout of each

transistor requires more diffusion area due to the spacing requirements of design rules. The correspondingly larger capacitances act to offset the advantages of progressive sizing.

6.3.3 Wire Capacitance

A third component of load capacitance is wire capacitance, or interconnect capacitance. In the past, the interconnect capacitance was not a significant part of the overall capacitance calculation due to the fact that the wires were relatively short and the devices were very large. Today, wires are much longer and devices are much smaller so we need to include this component in the load capacitance calculation. For very short wires, such as those less than a few microns, we can ignore the capacitance. For wires that are greater than a few microns, we should include the lumped wire capacitance. For very long wires, we will have to deal with distributed RC effects and capacitive coupling effects, as described in Chapter 10. In the meantime, for hand calculations, the lumped wire capacitance can be computed as

$$C_{\text{wire}} = C_{\text{int}} L_W = 0.2 \text{ fF}/\mu\text{m} \times (\text{wirelength}) \tag{6.12}$$

Capacitance Calculation for Inverter **Example 6.4**

Problem:

A CMOS inverter has a pull-up device that is 8λ:2λ and a pull-down device that is 4λ:2λ. It drives four identical inverters. Compute the load capacitance using 0.18 μm technology parameters. Assume that the wire capacitance is negligible.

Solution:

Fanout capacitance: inverter is driving four identical inverters:

$$C_{\text{fanout}} = 4 \times C_g(W_N + W_P) = 4(2 \text{ fF}/\mu\text{m})(0.4\ \mu\text{m} + 0.8\ \mu\text{m}) = 9.6 \text{ fF}$$

Self-capacitance:

$$C_{\text{self}} = C_{\text{eff}}(W_N + W_P) = (1 \text{ fF}/\mu\text{m})(0.4\ \mu\text{m} + 0.8\ \mu\text{m}) = 1.2 \text{ fF}$$

Total capacitance:

$$C_{\text{load}} = C_{\text{fanout}} + C_{\text{self}} + C_{\text{wire}} = 9.6 + 1.2 + 0 = 10.8 \text{ fF}$$

6.4 Improving Delay Calculation with Input Slope

So far we have assumed a step input at each gate when computing the delay. In reality, each input behaves more like a ramp with an exponential tail. This increases the delay relative to the step input case. In this section, we explore the effect of a finite input slope on the propagation delay.

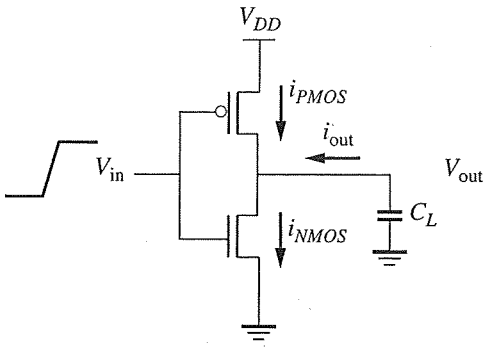


Figure 6.13
KCL at output of CMOS inverter.

Consider the inverter of Figure 6.13 where the input is switching as a ramp function with a given slope. We can apply KCL to the output node to obtain the relationship between the three currents:

$$i_{out} = C_L \frac{dV_{out}}{dt} = i_{NMOS} - i_{PMOS} \tag{6.13}$$

All three currents are a function of V_{in} and V_{out}. We are most interested in the charging and discharging current, i_{out}. It is possible to compute the output current using Equation (6.13) by selecting different values of V_{in} and V_{out}, calculating i_{NMOS} and i_{PMOS}, and then taking their difference. The results of such i_{out} computations are plotted in Figure 6.14 as a contour map on a two-dimensional plane of V_{out} versus V_{in}. Normally the voltage transfer characteristic is plotted in this plane. Here we plot the output current under all possible operating conditions. The regions with the same shading have i_{out} values that are in the same range.

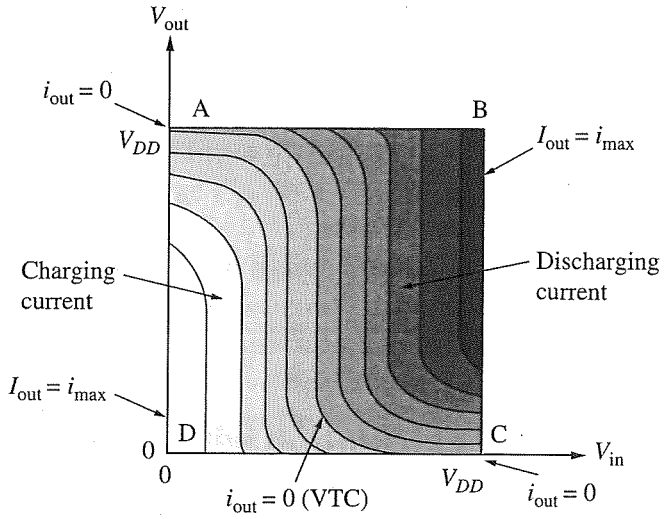


Figure 6.14
Inverter output current as a function of V_{out} and V_{in}.

Notice that the curve in the center of the plot (where $i_{out} = 0$) is exactly the voltage transfer characteristic (VTC). In fact, this line represents the dc operating point of the inverter as the input changes very slowly. However, when the input is varied rapidly, the operating points occur away from the VTC. If the input is increased rapidly, we move to the right of the VTC. There is more NMOS current than PMOS current, so the output capacitance is discharged. If the input is decreased rapidly, we move to the left of the VTC where the current through the PMOS device increases to charge the output capacitance. As we move to contours further away from the VTC in either direction, the current increases in magnitude (whether charging or discharging). Therefore, the VTC curve in the center represents a "valley" if the contour map is viewed as a three-dimensional diagram. As a result, the outermost edges are labeled as $i_{out} = i_{max}$ since these are the regions of highest current levels.

It is instructive to consider what happens on this contour map when a step input or a ramp input is applied. These two cases are shown in Figure 6.15. First consider the step input case of Figure 6.15a. If the input starts at 0 V and the output is at V_{DD}, we are at the top left corner of the plot labeled A. When a positive going step occurs, we move instantaneously to the opposite corner labeled B where $i_{out} = i_{max}$ (since the input is at V_{DD}). Then, the discharging current, due entirely to the NMOS device, follows along the right-most edge of the graph until point C is

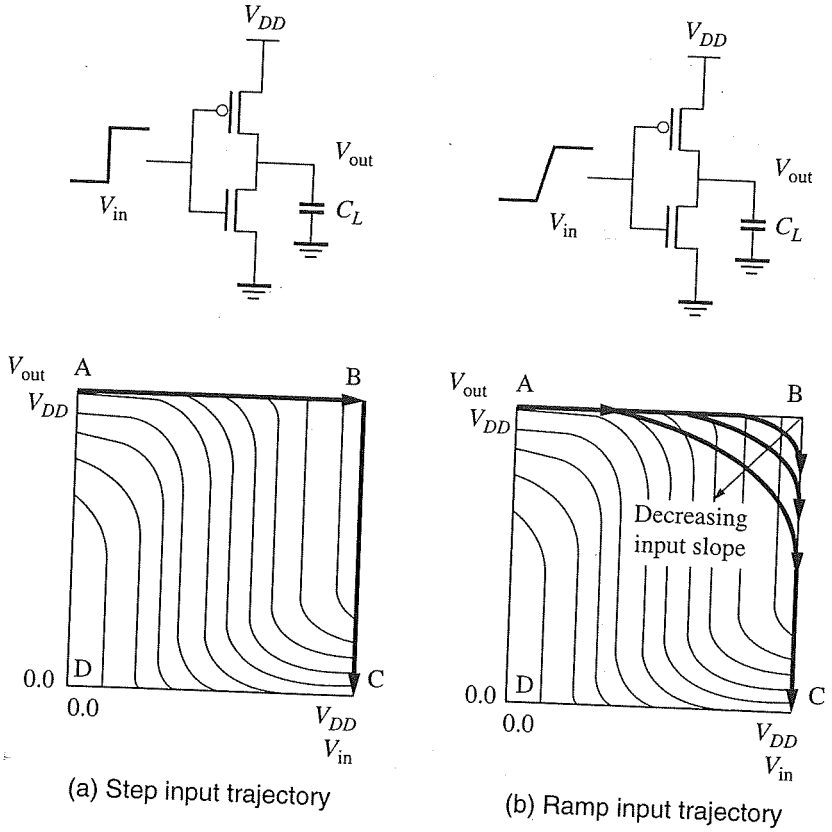


Figure 6.15
Simplified inverter output current as a function of V_{out} and V_{in}.

reached where $i_{out} = 0$. As the NMOS device moves from point B to point C, it switches from saturation to the linear region of operation. A negative going step would start at point C, switch immediately to point D and then eventually return to point A.

Next, examine the positive ramp input case in Figure 6.15b. In this case, the current trajectory depends on the slope of the input ramp. If the slope is high, it behaves similar to the step input case. As the slope is decreased, the trajectory "cuts the corner" as shown in the figure. This implies that the current gradually ramps up to i_{max} rather than instantaneously reaching this value. Therefore, the discharging process will take longer in the ramp case, since there is less current initially to discharge the capacitance.

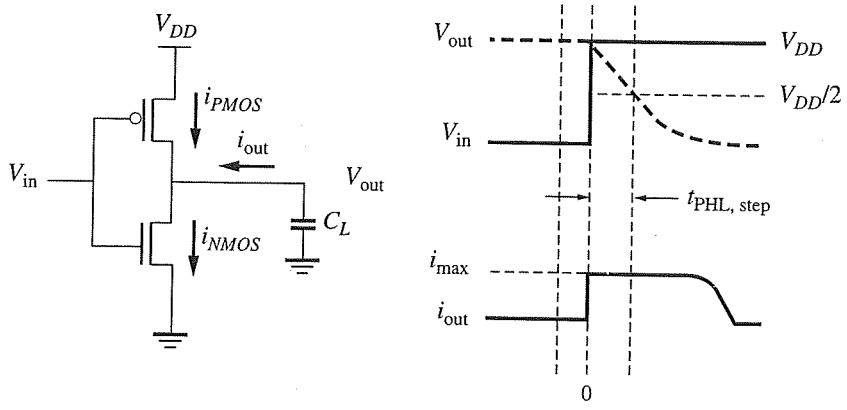
A similar kind of result is obtained for the negative ramp case except that the "corner-cutting" will be in the vicinity of D on the graph. Again, the effect of the actual input/output trajectory is to decrease the current supplied initially to charge the output. Therefore, the delay increases when a ramp input is applied.

The exact increase in the delay depends on a number of factors, but is ultimately controlled by the output current waveform as a function of time. The example below illustrates how analytical expressions can be obtained for the delay with step and ramp inputs.

Example 6.5 **Delay Calculation with Step and Ramp Inputs**

Problem:

(a) Compute the $t_{PHL,step}$ delay for the following waveforms due to a step input.



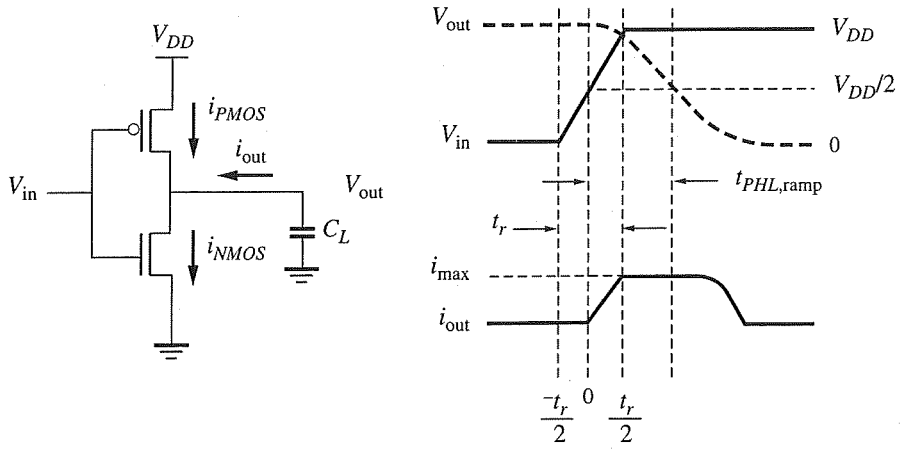
Solution:

The propagation delay calculation for the step case is very simple:

$$i_{out} = i_{max} = C_L \frac{dV_{out}}{dt} = C_L \frac{V_{DD}/2}{t_{PHL,step}}$$

$$\therefore t_{PHL,step} = C_L \frac{V_{DD}/2}{i_{max}}$$

(b) Compute the $t_{PHL,ramp}$ delay for the following waveforms for the input ramp using the corresponding output current. The rise time of the input voltage is t_r.



Solution:

The propagation delay for the ramp case can be derived as follows:

$$i_{out} = C_L \frac{dV_{out}}{dt}$$

$$\therefore \int i_{out}\, dt = \int C_L\, dV_{out}$$

$$\therefore \int_0^{t_r/2} i_{max} \frac{t}{t_r/2}\, dt + \int_{t_r/2}^{t_{PHL,ramp}} i_{max}\, dt = C_L \int_0^{V_{DD}/2} dV_{out}$$

$$\therefore \frac{i_{max}}{t_r/2} \frac{t^2}{2}\Big|_0^{t_r/2} + i_{max}\, t\Big|_{t_r/2}^{t_{PHL,ramp}} = C_L V_{DD}/2$$

$$\therefore i_{max} \frac{t_r}{4} + i_{max}(t_{PHL,ramp} - t_r/2) = C_L V_{DD}/2$$

$$\therefore t_{PHL,ramp} = \frac{t_r}{4} + \frac{C_L V_{DD}/2}{i_{max}}$$

(c) Write an expression for the ramp delay in terms of the step delay and the input rise time.

Solution:

By examining the results of the two solutions above, we find that:

$$t_{PHL,ramp} = \frac{t_r}{4} + t_{PHL,step}$$

If we use the approximation that $t_r \cong 2t_{PLH,step}$, then

$$t_{PHL,ramp} \approx \frac{2t_{PLH,step}}{4} + t_{PHL,step}$$

$$= \frac{t_{PHL,step}}{2} + t_{PHL,step}$$

From the above example, it is clear how the input ramp affects the propagation delay. It adds a delay term Δt_{ramp} that depends on the rise/fall delay of the input ramp. Therefore, the total delay can be written as:

$$t_{ramp} = \Delta t_{ramp} + t_{step}$$

If we were able to relate Δt_{ramp} to the propagation delay for a step input, it would greatly simplify the expression. Let us refer to the input rise/fall propagation delay as t_{in}. Then, according to the example above, we find that $\Delta t_{ramp} = t_{in}/2$ for the given output waveform. That is,

$$t_{ramp} \approx \frac{t_{in}}{2} + t_{step} \tag{6.14}$$

This is a very convenient result for the delay of an inverter chain with a ramp input as shown in Figure 6.16. The value of t_{step} for an inverter is given by 0.7RC and the value of Δt_{ramp} for the next stage can be approximated as 0.7RC/2, which we round down to 0.3RC for convenience. If we combine terms in the propagation delay as shown in Figure 6.16 and assume that the input propagation delay is roughly equal to the propagation delay of the last stage (a reasonable assumption if propagation delays are roughly equal in the circuit), then the total delay can be computed very simply as:

$$\text{Total\_delay} \approx \sum_i R_i C_i \tag{6.15}$$

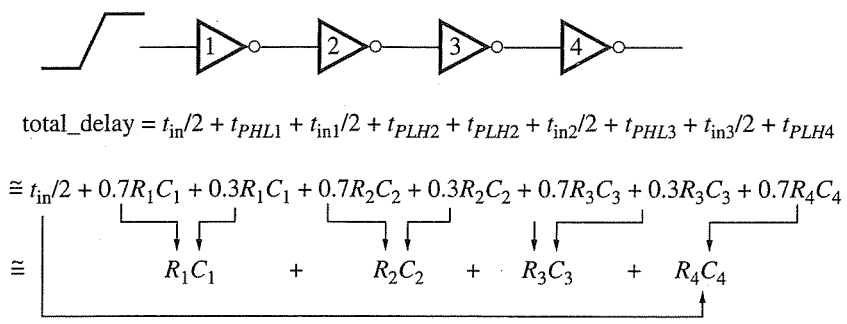


Figure 6.16
Inverter chain delay for a ramp input.

That is, we no longer have to carry around a factor of 0.7 in the delay calculation if we are dealing with ramp inputs. Since most circuits that we will encounter have ramp inputs, we can safely use Equation (6.15) for delay calculation.

In summary, accurate propagation delay estimates are obtained for inputs with finite slopes by simply adding up the product of the on-resistance and the load capacitance for each stage of logic. The resulting delays are consistent with SPICE results if the on-resistances and capacitances are computed accurately.

Delay Calculation for Inverter Driving Four Fanout Inverters **Example 6.6**

Problem:

(a) A CMOS inverter has a pull-up device that is 8λ:2λ and a pull-down device that is 4λ:2λ. It drives four identical inverters. Compute the inverter delay using 0.18 μm technology parameters. Assume a ramp input and negligible wire capacitance.

(b) Compute the delay for a chain of four inverters assuming a ramp input. Include the effect of differing rise and fall delays.

Solution:

(a) From a previous example, we obtained the values of the fanout and self capacitance. We will set the wire capacitance to zero for simplicity:

$$C_{load} = C_{fanout} + C_{self} + C_{wire} = 9.6 + 1.2 + 0 \approx 10.8 \text{ fF}$$

The t_{PHL} delay for an inverter driving four identical inverters is

$$t_{PHL} = R_{eff}C_{load} = (12.5\ \text{k}\Omega)\left(\frac{L}{W}\right)(10.8\ \text{fF})$$

$$= (12.5\ \text{k}\Omega)\left(\frac{1}{2}\right)(10.8\ \text{fF}) = (6.25\ \text{k}\Omega)(10.8\ \text{fF}) \approx 68\ \text{ps}$$

The t_{PLH} delay for an inverter driving four identical inverters is

$$t_{PHL} = R_{eff}C_{load} = (30\ \text{k}\Omega)\left(\frac{L}{W}\right)(10.8\ \text{fF})$$

$$= (30\ \text{k}\Omega)\left(\frac{1}{4}\right)(10.8\ \text{fF}) = 7.5\ \text{k}\Omega(10.8\ \text{fF}) \approx 81\ \text{ps}$$

$$\therefore t_p = \frac{t_{PHL} + t_{PLH}}{2} = \frac{68 + 81}{2} = 74.5\ \text{ps}$$

Therefore, the average fanout-of-four (FO4) delay of the inverter is roughly 75 ps.

(b) For a chain of four inverters, compute the FO1 rise and fall delays.

$$t_{PHL} = R_{eff}C_{load} = (6.35\ \text{k}\Omega)(2.4 + 1.2)\text{fF} = 22.5\ \text{ps}$$

$$t_{PLH} = R_{eff}C_{load} = (7.5\text{k}\Omega)(2.4 + 1.2)\text{fF} = 27\ \text{ps}$$

The total delay is twice the sum of each delay:

$$\tau_{total} = 2(t_{PHL}+t_{PLH}) = 2(22.5\ \text{ps}+27\ \text{ps}) = 99\ \text{ps}$$

Example 6.7 **Optimal PMOS Device Size For Inverter Chain**

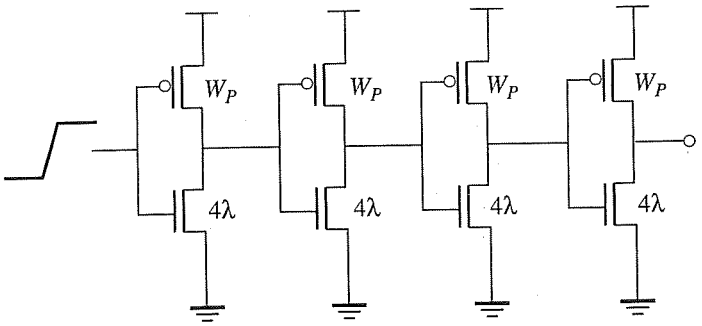
Problem:

Inverter sizing can be performed to equalize rise/fall delays or to minimize the propagation delay. Consider the four inverters in a chain shown below. Assuming that the NMOS devices are all 4λ, size the PMOS devices with the following objectives:

(a) equalize the rise/fall delays

(b) minimize the delay through the chain

What is the delay through four such inverters in the two cases?



Solution:

(a) To equalize the delays, the PMOS devices would have to be approximately 2.4 times larger than the 4λ NMOS device. Therefore, the PMOS device size is approximately 10λ. The total capacitance at each output is 4.2 fF.

The delay of the chain is simply four times the delay of each stage:

$$\tau_{total} = 4t_{PHL} = 4R_{eff}C_{load} = 4(12.5\ \text{k}\Omega)\left(\frac{2}{4}\right)(4.2\ \text{fF}) \approx 105\ \text{ps}$$

This is a larger value than we obtained with a PMOS device of 8λ in the previous example.

(b) To minimize the delay, write the delay equation for the first two inverters:

$$D = \sum RC = (R_P)(C_g + C_{eff})(W_P + W_N) + (R_N)(C_g + C_{eff})(W_P + W_N)$$

$$= \left(\frac{30L_P}{W_P} + \frac{12.5L_N}{W_N}\right)(C_g + C_{eff})(W_P + W_N)$$

$$= \left(\frac{30(2\lambda)}{W_P} + \frac{12.5(2\lambda)}{4\lambda}\right)(2 + 1)(W_P + 4\lambda)$$

$$= \left(\frac{60\lambda}{W_P} + \frac{25\lambda}{4\lambda}\right)(2 + 1)(W_P + 4\lambda)$$

$$= \left(\frac{60\lambda(3)(W_P + 4\lambda)}{W_P} + \frac{25}{4}(3)(W_P + 4\lambda)\right) = \left(180\lambda + \frac{720\lambda^2}{W_P}\right) + \frac{75}{4}(W_P + 4\lambda)$$

To optimize the delay, take the derivative of this expression with respect to W_P.

$$\frac{\partial D}{\partial W_P} = -\frac{720\lambda^2}{W_P^2} + \frac{75}{4} = 0$$

$$\therefore W_P = \sqrt{\frac{720\lambda^2}{75/4}} = 2\lambda\sqrt{720/75} = 6.2\lambda$$

Based on the simple models for delay, the optimal ratio of the PMOS device to the NMOS device is approximately 6:4.

From the previous example, we can recalculate the delay for a chain of four inverters:

$$C_{load} = C_{fanout} + C_{self} + C_{wire} = 2 + 1 + 0 \approx 3 \text{ fF}$$

The rise delay for an inverter driving another identical inverter is:

$$t_{PLH} = R_{eff}C_{load} = (30\ \text{k}\Omega)\left(\frac{2}{6}\right)(3\ \text{fF}) \approx 30\ \text{ps}$$

The fall delay for an inverter driving another identical inverter is:

$$t_{PHL} = R_{eff}C_{load} = (12.5\ \text{k}\Omega)\left(\frac{2}{4}\right)(3\ \text{fF}) \approx 18.8\ \text{ps}$$

The total delay is

$$\tau_{total} = 2(t_{PHL} + t_{PLH}) = 2(18.8\ \text{ps} + 30\ \text{ps}) = 97.6\ \text{ps}$$

which is slightly faster than the inverters in the previous example.

Exercise 6.2 Use SPICE simulations to find the optimal sizing of the PMOS and NMOS devices that produce minimum delay through the inverter chain in the previous example. How do your results compare with the hand calculations?

Clearly the selection of the PMOS:NMOS ratio from a timing perspective is a function of many factors. We can equalize the rise and fall delays, or minimize the delay through a chain. A number of nonlinear effects, such as ramp inputs and oxide and junction capacitances will determine the actual delay. Designers typically use a 2:1 ratio since it is a reasonable compromise between minimum delay and equal rise/fall times.

6.5 Gate Sizing for Optimal Path Delay

6.5.1 Optimal Delay Problem

We now address the general design problem of *optimizing* gate sizes to minimize the delay of a logic path. In order to carry out circuit optimization, the problem must be specified carefully. Consider the circuit of Figure 6.17a where we are trying to drive a large load, C_{load}, with the objective of minimizing the delay. Without any other constraints, we would try to make the inverter as large as possible, as shown in Figure 6.17b, so that the $R_{eff}C_{load}$ is as small as possible. However, there is no gate driving this inverter and, in this unconstrained (and unrealistic) situation, it is difficult to produce a meaningful solution. Therefore, the optimization problem has not been specified properly.

Next, consider the inverter chain of Figure 6.18 driving the same capacitive load. If the problem is to minimize the delay through the chain, our first instinct would be to size the third inverter to be as large as possible. However, this time the third inverter places a load on the second inverter and its delay will increase as we increase the size of the third inverter. This is illustrated in Figure 6.19 along with a corresponding increase in its input capacitance. We have effectively shifted the delay problem to the previous gate.

If we make the second inverter large so that it can drive the third inverter, then the first inverter is heavily loaded. As a result, we are producing locally optimal

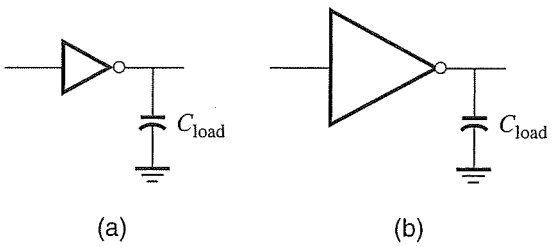


Figure 6.17
Sizing inverters to drive a large capacitive load.

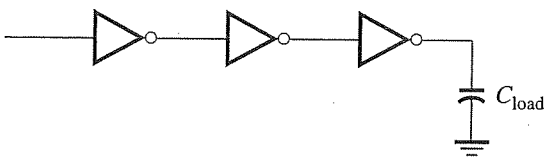


Figure 6.18
Optimal path delay problem.

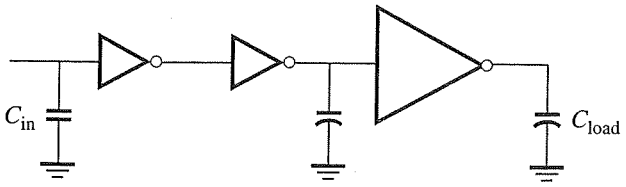


Figure 6.19
Input loading effects of large drivers.

solutions, but inadvertently passing the problem to the previous stage. Again, this problem is not properly constrained and does not have a meaningful solution. The proper specification of the optimal path delay problem involves the minimization of the path delay given both input *and* output loading constraints. In Figure 6.19, if we specify both C_{in} and C_{load} and ask for the optimal sizing to minimize the delay, the problem is properly specified, and we can focus on its solution.

Given the proper specification, what is the optimal sizing of the gates in the logic path? Actually, there are two unknowns in general: the number of logic gates needed in the path and their corresponding sizes. To carry out this optimization, we write the delay equation in the form that was derived earlier:

$$\text{path\_delay} = \sum R_i C_i$$

where R_i is the driving resistance of each gate and C_i is the output loading capacitance on the gate. Then, we select the transistor widths to minimize the delay.

6.5.2 Inverter Chain Delay Optimization—FO4 Delay

We now turn our attention to the problem of driving a large capacitive load starting with a small input capacitance. The assumption here is that we need several logic stages of increasing size to drive the load. How many stages do we need and how should they be scaled in size? In the previous section, we noticed that when we increased the inverter size, two things happened: We reduced the delay of the gate and increased the input capacitance seen by the previous gate. That is, changing the size of a gate affects both the input capacitance and the output resistance of a gate.

The input capacitance of a gate completely specifies the sizes of the devices and is given by

$$C_{\text{in}} = C_g(W_n + W_p) = C_g(W_n + 2W_n) = C_g(3W_n) \tag{6.16}$$

If we know the total input capacitance of an inverter, we can assign the specific PMOS and NMOS device widths using Equation (6.16).

The effective output resistance for the NMOS device is given by

$$R_{eff} = R_{eqn}\left(\frac{L_n}{W_n}\right) \tag{6.17}$$

A useful property of a gate can be obtained by taking the product of the input capacitance and the output resistance. The product generates a time constant which is an intrinsic property of the gate. In particular, if we multiply Equations (6.16) and (6.17) together, we obtain

$$\tau_{\text{inv}} = R_{eff}C_{\text{in}} = R_{eqn}\left(\frac{L_n}{W_n}\right) C_g\,(3W_n) = 3R_{eqn}C_gL_n \tag{6.18}$$

The reason why this value is interesting is that it captures the input and output effects of sizing in one term. We want to know how effective a gate is at driving a load capacitance while minimizing its input capacitance. This single quantity tells us about these two aspects of the gate. Note that the size of the inverter can be doubled, tripled, or made any size, and the time constant will remain the same. Of course, the self-capacitance term increases when the size increases, but it is not part of this intrinsic time constant. However, it is part of another term associated with the parasitic capacitance of the gate, as we will find shortly.

Next, consider the delay of a single-stage inverter shown in Figure 6.20. If the input has a finite rise time, then we can write the delay as

$$t_{\text{delay}} = R_{eff}[C_{\text{out}} + C_{\text{self}}]$$

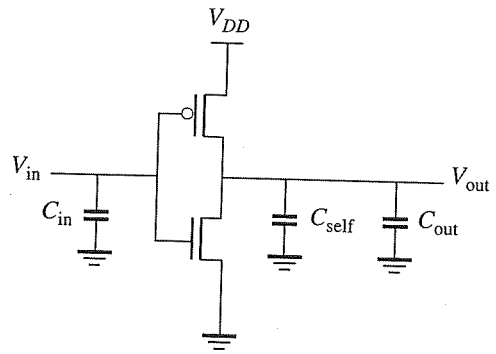


Figure 6.20
Delay for inverter driving a load.

In order to incorporate the intrinsic time constant term, τ_{inv}, into the formulation, we multiply and divide through by C_{in}:

$$t_{delay} = R_{eff}C_{in}\left[\frac{C_{out}}{C_{in}} + \frac{C_{self}}{C_{in}}\right] = \tau_{inv}\left[\frac{C_{out}}{C_{in}} + \gamma_{inv}\right] \tag{6.19}$$

where γ_{inv} is the ratio of self-capacitance to input capacitance for the inverter:

$$\gamma_{inv} = \frac{C_{self}}{C_{in}} \tag{6.20}$$

This value is highly dependent on the layout of the gate, since C_{self} depends on the layout. The ratio of C_{out}/C_{in} is called the fanout ratio, f, or the electrical effort.

Computation of τ_{inv} and γ_{inv} for 0.13 μm Technology **Example 6.8**

Problem:

Compute τ_{inv} and γ_{inv} for an inverter in a 0.13 μm technology.

Solution:

$$\tau_{inv} = 3R_{eqn}C_gL_n = 3(12.5\ \text{k}\Omega)(2\ \text{fF}/\mu\text{m})(0.1\ \mu\text{m}) = 7.5\ \text{ps}$$

$$\gamma_{inv} = \frac{C_{self}}{C_{in}} = \frac{C_{eff}(3W)}{C_g(3W)} = \frac{1\ \text{fF}/\mu\text{m}}{2\ \text{fF}/\mu\text{m}} = 0.5$$

Compute τ_{inv} and γ_{inv} for a 0.18 μm technology. **Exercise 6.3**

Returning to the inverter sizing problem, we want to optimize the delay of the chain in Figure 6.21.

The delay through the stages can be computed as:

$$\text{total\_delay} = \sum_{j=1}^{N} \tau_{inv}\left(\frac{C_{j+1}}{C_j} + \gamma_{inv}\right)$$

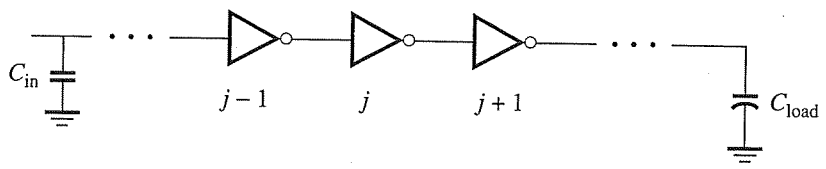


Figure 6.21
Optimal sizing of inverter chain.

By substituting in the widths to compute the capacitances, we obtain

$$\text{total\_delay} = \sum_j \tau_{\text{inv}}\left(\frac{C_g W_{j+1}}{C_g W_j} + \gamma_{\text{inv}}\right) = \sum_j \tau_{\text{inv}}\left(\frac{W_{j+1}}{W_j} + \gamma_{\text{inv}}\right)$$

We can gain some insight into what happens at the optimal solution by taking two consecutive delay terms that are dependent upon the size of inverter j:

$$D_j = \tau_{\text{inv}}\left(\frac{W_j}{W_{j-1}} + \gamma_{\text{inv}}\right) + \tau_{\text{inv}}\left(\frac{W_{j+1}}{W_j} + \gamma_{\text{inv}}\right)$$

To obtain the optimal delay for these two stages, we take the partial derivative of D_j with respect to W_j:

$$\frac{\partial D_j}{\partial W_j} = \tau_{\text{inv}} \frac{1}{W_{j-1}} - \tau_{\text{inv}} \frac{W_{j+1}}{W_j^2} = 0$$

$$\therefore \frac{W_j}{W_{j-1}} = \frac{W_{j+1}}{W_j}$$

$$\therefore W_j = \sqrt{W_{j+1} W_{j-1}}$$

An important result has been produced: to obtain the minimum delay, the size of the middle inverter should be the *geometric mean* of the size of the previous inverter and the next inverter. In this case, the geometric mean is simply the square root, since we are only considering two stages. In general, it is the Nth root if we were optimizing all N stages of inverters at once.

One way to achieve this geometric sizing is to increase each inverter by a fanout factor f which can be determined using the relationship between the input capacitance, the output capacitance, and the number of stages. The general case is shown in Figure 6.22. Each stage drives an inverter that is f times larger than itself. The size of each inverter is the geometric mean of the inverter before and after it, as required for optimality. It is interesting to note that the delay through each gate must be the

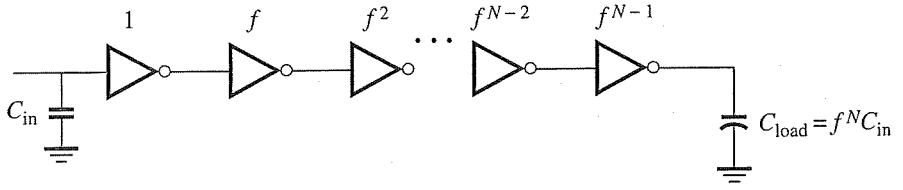


Figure 6.22
Series chain of inverters.

same since each one drives an inverter that is f times itself. The overall delay is simply N times the delay of one inverter.

Since both f and N are unknowns, we can use the relationship between them to determine their optimal values. Using Figure 6.22 we see that

$$f^N C_{\text{in}} = C_{\text{load}}$$

$$\therefore N = \frac{\ln(C_{\text{load}}/C_{\text{in}})}{\ln f} \tag{6.21}$$

From Equation (6.19), the gate delay and the total delay can be written as

$$\text{gate\_delay} = \tau_{\text{inv}}\left(\frac{C_j}{C_{j-1}} + \gamma_{\text{inv}}\right)$$

$$\text{total\_delay} = N \times \tau_{\text{inv}}\left(\frac{C_j}{C_{j-1}} + \gamma_{\text{inv}}\right) \tag{6.22}$$

Substituting in Equation (6.21), and setting $f = C_j/C_{j-1}$, we obtain:

$$\text{total\_delay} = \frac{\ln(C_{\text{load}}/C_{\text{in}})}{\ln f} \times \tau_{\text{inv}}(f + \gamma_{\text{inv}}) \tag{6.23}$$

The total delay is plotted as a function of f and γ in Figure 6.23. The curves indicate that the optimal value of f lies in the range of 2.5 to 4. The curves have a very shallow optimum so any value in this range may be appropriate, depending on γ. In prior textbooks, the optimal value of f was found to be e, the exponential value.

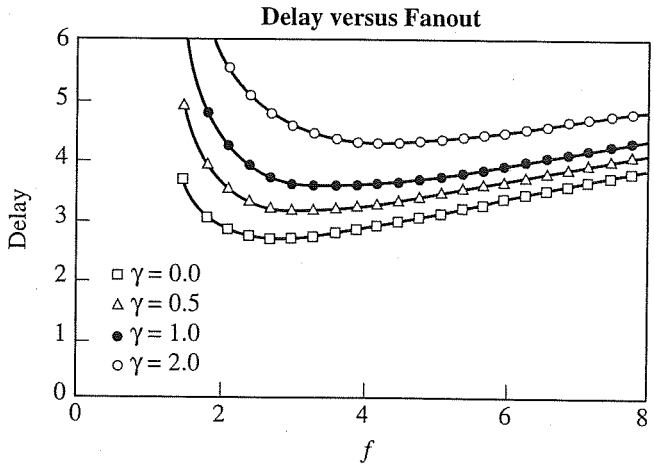


Figure 6.23
Delay versus fanout f for different γ values.

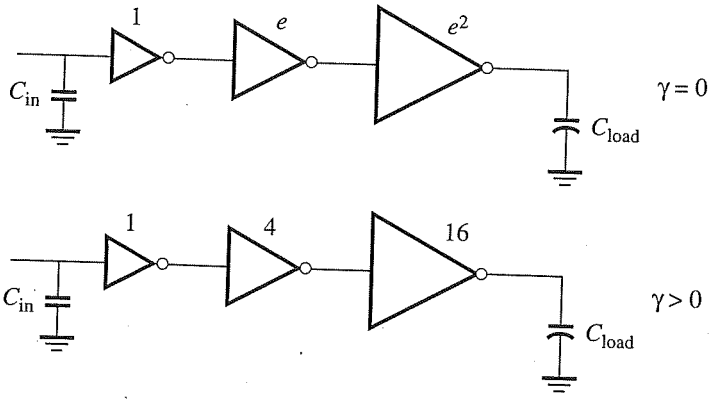


Figure 6.24
Inverter sizing for optimal delay.

From Equation (6.23), this value is obtained when $\gamma = 0$ which implies that the junction capacitance is ignored. From the curve for $\gamma = 0$ in Figure 6.23, the optimum value of e can also be extracted. This type of sizing is shown in Figure 6.24. Today the junction capacitance and interconnect capacitance can act to increase the needed drive of the inverter. In a 0.13 μm technology, $\gamma = 0.5$. Therefore, the use of $f = 3$ or $f = 4$ is more suitable. Typically, we use the value $f = 4$ and give the associated delay a special name: the fanout of 4 delay, or FO4 delay. The sizing for a three-stage inverter chain under FO4 rules is given in Figure 6.24 where each successive inverter is 4 times larger than the previous one.

Example 6.9 **Optimal Sizing for Inverter Chains**

Problem:

Compute the optimal inverter fanout ratio f for a three-stage inverter chain with $C_{load} = 200$ fF and $C_{in} = 1$ fF. Recompute the value of f if the optimal number of stages is used. Then, compute the delay through the chain in the two cases, assuming $\tau_{inv} = 7.5$ ps and $\gamma = 0.5$.

Solution:

Use the fact that

$$N \ln f = \ln\left(\frac{C_{load}}{C_{in}}\right)$$

Therefore,

$$\ln f = \frac{1}{3} \ln(200)$$

$$\therefore f = 5.84$$

This is the optimal fanout if we are constrained to use three stages. If we want to improve the delay, we first remove the restriction of the number of stages. Then, we can compute the number of stages using the equation:

$$N = \ln\left(\frac{C_{\text{load}}}{C_{\text{in}}}\right)/\ln f$$

To solve for N, we need to estimate f. From Figure 6.23 $f \approx 3.6$. Therefore,

$$N = \ln(200)/\ln(3.6) = 4.1$$

Now try using four stages (since an integer must be used). Then, f can be recomputed as

$$\ln f = \frac{1}{4}\ln(200)$$

$$\therefore f = 3.8$$

The delays of the two solutions can be computed using Equation (6.22):

$$\text{total\_delay} = N \times \tau_{\text{inv}}\left(\frac{C_j}{C_{j-1}} + \gamma_{\text{inv}}\right)$$

For the three-stage case,

$$\text{total\_delay} = 3 \times (7.5\text{ ps})(5.84 + 0.5) = 142.8\text{ ps}$$

For the four-stage case,

$$\text{total\_delay} = 4 \times (7.5\text{ ps})(3.8 + 0.5) = 128\text{ ps}$$

Therefore, the best solution is to have four inverter stages with each inverter sized to be 3.8X larger than the previous inverter.

6.5.3 Optimizing Paths with NANDs and NORs

Chains of NAND or NOR gates can be handled in the same way as inverters with appropriate modifications. For the NAND chain of Figure 6.25, the total delay can be derived as

$$\text{total\_delay} = \sum_j \tau_{\text{nand}}\left(\frac{C_{j+1}}{C_j} + \gamma_{\text{nand}}\right)$$

where τ_{nand} is the intrinsic time constant for the NAND gate, and γ_{nand} is the ratio of the self-capacitance to the input gate capacitance. If the same curves of delay versus fanout are drawn for the NAND gates as we constructed for the inverters (see Figure 6.23), we would also find that $f \approx 4$. The optimal value of f would equalize the delays of all stages.

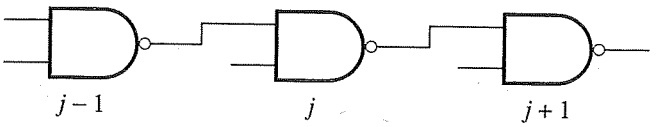


Figure 6.25
Series chain of NAND gates.

For a chain of consecutive NOR gates, we would produce a similar equation:

$$\text{total\_delay} = \sum_j \tau_{\text{nor}}\left(\frac{C_{j+1}}{C_j} + \gamma_{\text{nor}}\right)$$

where τ_{nor} is the intrinsic time constant for the NOR gate, and γ_{nor} is its ratio of the self-capacitance to the input gate capacitance. The fanout ratio $f \approx 4$ works well for this case. Again, the proper value of f would make the delays of all stages identical.

The intrinsic time constants for the NAND and NOR gates can be derived as follows. Considering the device sizes shown in Figure 6.4, we can compute the effective output resistance and the input capacitance, and then multiply the two together:

$$\tau_{\text{nand}} = R_{\text{eff}}C_{\text{in}} = R_{eqn}\left(\frac{L_n}{W_n}\right) 4W_nC_g = 4R_{eqn}C_gL_n$$

$$\tau_{\text{nor}} = R_{\text{eff}}C_{\text{in}} = R_{eqn}\left(\frac{L_n}{W_n}\right) 5W_nC_g = 5R_{eqn}C_gL_n \qquad (6.24)$$

Typical logic paths in a digital circuit may have a variety of gate types. Consider the optimal delay for the situation shown in Figure 6.26.

The total delay can be written as follows:

$$\text{total\_delay} = \tau_{\text{nand}}\left(\frac{C_{j+1}}{C_j} + \gamma_{\text{nand}}\right) + \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}} + \gamma_{\text{inv}}\right) + \tau_{\text{nor}}\left(\frac{C_{j+3}}{C_{j+2}} + \gamma_{\text{nor}}\right) \qquad (6.25)$$

The delay through stages j and j + 1 is given by

$$D_{j+1} = \tau_{\text{nand}}\left(\frac{C_{j+1}}{C_j} + \gamma_{\text{nand}}\right) + \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}} + \gamma_{\text{inv}}\right)$$

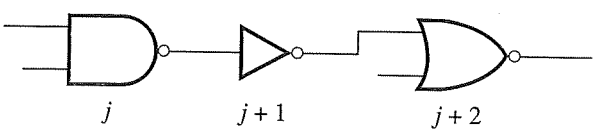


Figure 6.26
Series of mixed gates in a logic path.

For minimum delay, we take the derivative with respect to C_{j+1}:

$$\frac{\partial D_{j+1}}{\partial C_{j+1}} = \tau_{\text{nand}}\left(\frac{1}{C_j}\right) - \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}^2}\right) = 0$$

$$\therefore \tau_{\text{nand}}\left(\frac{C_{j+1}}{C_j}\right) = \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}}\right)$$

$$\therefore \tau_{\text{nand}} FO_j = \tau_{\text{inv}} FO_{j+1}$$

where $FO_j = C_{j+1}/C_j$ and $FO_{j+1} = C_{j+2}/C_{j+1}$. Another important result has been established. For this case, the delay is minimized when the $\tau \times FO$ of a given gate is equal to the $\tau \times FO$ of the next gate. That is, the gate delays do not have to be equal but rather the fanout portions of the delays must be equal to reach the optimal solution. To support this result, if we now apply the analysis to the next two stages we would obtain

$$D_{j+2} = \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}} + \gamma_{\text{inv}}\right) + \tau_{\text{nor}}\left(\frac{C_{j+3}}{C_{j+2}} + \gamma_{\text{nor}}\right)$$

$$\frac{\partial D_{j+2}}{\partial C_{j+2}} = \tau_{\text{inv}}\left(\frac{1}{C_{j+1}}\right) - \tau_{\text{nor}}\left(\frac{C_{j+3}}{C_{j+2}^2}\right) = 0$$

$$\therefore \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}}\right) = \tau_{\text{nor}}\left(\frac{C_{j+3}}{C_{j+2}}\right)$$

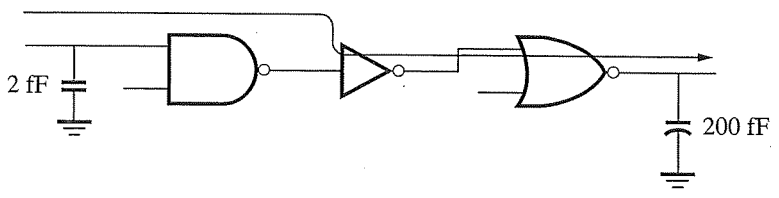
$$\therefore \tau_{\text{inv}} FO_{j+1} = \tau_{\text{nor}} FO_{j+2}$$

Again, the same result is obtained: to optimize the delay, we must set the fanout portion of the delay to be equal for all gates.

Computing Optimal Gate Sizes along a Critical Path **Example 6.10**

Problem:

Find the device sizes that optimize the delay through the indicated path for the circuit below.



Solution:

We must equalize the fanout portion of the delay. Therefore,

$$\therefore \tau_{\text{nand}}\left(\frac{C_{j+1}}{C_{\text{in}}}\right) = \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}}\right) = \tau_{\text{nor}}\left(\frac{C_{\text{load}}}{C_{j+2}}\right)$$

We take the product of these three components and then obtain the geometric mean:

$$\begin{aligned}\text{Fanout\_delay} &= \sqrt[3]{\tau_{\text{nand}}\left(\frac{C_{j+1}}{C_{\text{in}}}\right) \times \tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}}\right) \times \tau_{\text{nor}}\left(\frac{C_{\text{load}}}{C_{j+2}}\right)} \\ &= \sqrt[3]{\tau_{\text{nand}} \times \tau_{\text{inv}} \times \tau_{\text{nor}}\left(\frac{C_{\text{load}}}{C_{\text{in}}}\right)} = \sqrt[3]{4 \times 3 \times 5\left(\frac{200}{2}\right)} \times R_{eqn}C_gL_n \\ &= 18.2\, R_{eqn}C_gL_n\end{aligned}$$

Therefore, the input capacitance for each gate can be computed by setting the fanout delay to the above result:

$$\tau_{\text{nor}}\left(\frac{C_{\text{load}}}{C_{j+2}}\right) = 5R_{eqn}C_gL_n\left(\frac{200\text{ fF}}{C_{j+2}}\right) = 18.2\, R_{eqn}C_gL_n$$

$$\therefore C_{j+2} = 55\text{ fF}$$

$$\tau_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}}\right) = 3R_{eqn}C_gL_n\left(\frac{55\text{ fF}}{C_{j+1}}\right) = 18.2\, R_{eqn}C_gL_n$$

$$\therefore C_{j+1} = 9.1\text{ fF}$$

$$\tau_{\text{nand}}\left(\frac{C_{j+1}}{C_{\text{in}}}\right) = 4\, R_{eqn}C_gL_n\left(\frac{9.1\text{ fF}}{C_{\text{in}}}\right) = 18.2\, R_{eqn}C_gL_n$$

$$\therefore C_{\text{in}} = 2\text{ fF}$$

The final result is consistent with the input capacitance specified in the problem. The device sizes are determined by the relative sizes of the transistors in Figure 6.4.

For the NAND gate: $C_{\text{in}} = 2$ fF. Therefore, $W_p = W_n = 0.5\ \mu$m.

For the inverter: $C_{\text{in}} = 9.1$ fF. Therefore, $W_p = 3\ \mu$m and $W_n = 1.5\ \mu$m.

For the NOR gate: $C_{\text{in}} = 55$ fF. Therefore, $W_p = 22\ \mu$m and $W_n = 5.5\ \mu$m.

6.6 Optimizing Paths with Logical Effort

6.6.1 Derivation of Logical Effort

We need a more convenient method to optimize logic circuits based on the results of the previous section. Fortunately such an approach exists. Rather than carrying around the various τ values for each type of gate, one approach to simplifying the expressions is to normalize them relative to the inverter characteristic, τ_{inv}.

We can define a new term, called the *logical effort (LE)*, which is the ratio of the intrinsic time constant for a gate to the intrinsic time constant of an inverter. For example, the logical effort for an inverter would be $\tau_{inv}/\tau_{inv} = 1$; the logical effort of a NAND gate would be τ_{nand}/τ_{inv}; and, the logical effort of a NOR gate would be τ_{nor}/τ_{inv}. Using the *LE*, we can write normalized delay equations.

For example, Equation (6.25) for the total delay can be rewritten as

$$\frac{\text{total\_delay}}{\tau_{inv}} = \frac{\tau_{nand}}{\tau_{inv}}\left(\frac{C_{j+1}}{C_j} + \gamma_{nand}\right) + \frac{\tau_{inv}}{\tau_{inv}}\left(\frac{C_{j+2}}{C_{j+1}} + \gamma_{inv}\right) + \frac{\tau_{nor}}{\tau_{inv}}\left(\frac{C_{j+3}}{C_{j+2}} + \gamma_{nor}\right)$$

In normalized form, the delay equation is

$$D = (LE_{nand}FO_1 + P_{nand}) + (LE_{inv}FO_2 + P_{inv}) + (LE_{nor}FO_3 + P_{nor})$$

Each gate produces a term $LE \times FO + P$ in the delay expression. This may be written in a more compact form:

$$D = \sum (LE \times FO + P)$$

where LE = logic effort = τ_{gate}/τ_{inv}, FO_j = fanout = (C_{j+1}/C_j), and P = parasitic term = $LE \times \gamma_{gate}$.

Using these definitions, we can go back to the earlier equations for optimal delays and realize that, for the minimum delay, we need to equalize $LE \times FO$ for all gates. When we have only one type of gate, we try to make each gate delay the same as the others. This is possible since their P terms are all the same. When we have many different types of gates in the logic path, we require only that the fanout portion of the delay, $LE \times FO$, be the same for all gates, since their P terms may all be different.

The basic elements of logical effort have been described, but a number of questions remain unanswered at this stage:

1. How do we obtain the logical effort (*LE*) of any type of logic gate?
2. How do we compute the parasitic term (*P*) for any type of logic gate?
3. How do we use the *LE* and *P* values to find the optimal delay and device sizes along a critical path in a logic circuit?

To determine the *LE* of a gate, we revisit the definition of τ of an inverter:

$$\tau_{inv} = 3R_{eqn}C_gL_n$$

The corresponding values for the NAND and NOR gates are

$$\tau_{nand} = R_{eff}C_{in} = R_{eqn}\left(\frac{L_n}{W_n}\right)4W_nC_g = 4R_{eqn}C_gL_n$$

$$\tau_{nor} = R_{eff}C_{in} = R_{eqn}\left(\frac{L_n}{W_n}\right)5W_nC_g = 5R_{eqn}C_gL_n$$

Using these values, we find that

$$LE_{inv} = \frac{\tau_{inv}}{\tau_{inv}} = 1 \qquad LE_{nand} = \frac{\tau_{nand}}{\tau_{inv}} = \frac{4}{3} \qquad LE_{nor} = \frac{\tau_{nor}}{\tau_{inv}} = \frac{5}{3}$$

We can also use another approach to compute *LE*. If we take the definition of τ and multiply it by C_{load}, we have the following:

$$\tau \times C_{load} = R_{eff} C_{in} C_{load} = R_{eff} C_{load} C_{in}$$

$$\therefore LE = \frac{(R_{eff} C_{load} C_{in})_{gate}}{(R_{eff} C_{load} C_{in})_{inv}}$$

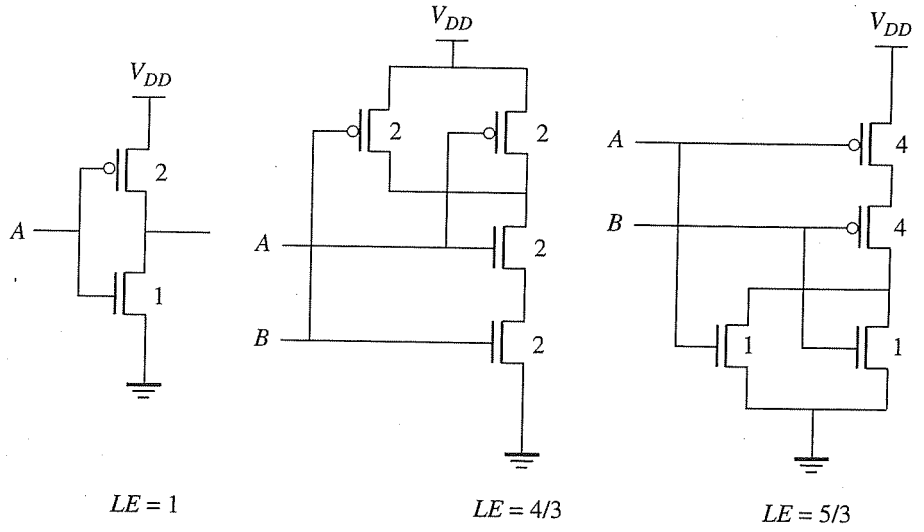
Using this form, there are two other ways to produce the logical effort:

1. Set the delays of the inverter and the gate to be the same; then, take the ratio of the input capacitances, or
2. Set the input capacitances to be same; then, take the delay ratio.

The first approach is illustrated using Figure 6.27, where the widths have been omitted for convenience. The NAND and NOR gates have already been sized to have the same delay as the inverter. Therefore, we can simply use the input capacitance ratios. Consider input *A*:

$$\text{For the NAND gate:} \quad LE = \frac{(C_{in})_{nand}}{(C_{in})_{inv}} = \frac{2+2}{3} = \frac{4}{3}$$

$$\text{For the NOR gate:} \quad LE = \frac{(C_{in})_{nor}}{(C_{in})_{inv}} = \frac{4+1}{3} = \frac{5}{3}$$



LE = 1 *LE* = 4/3 *LE* = 5/3

Figure 6.27
Logical effort for gates with equal delays.

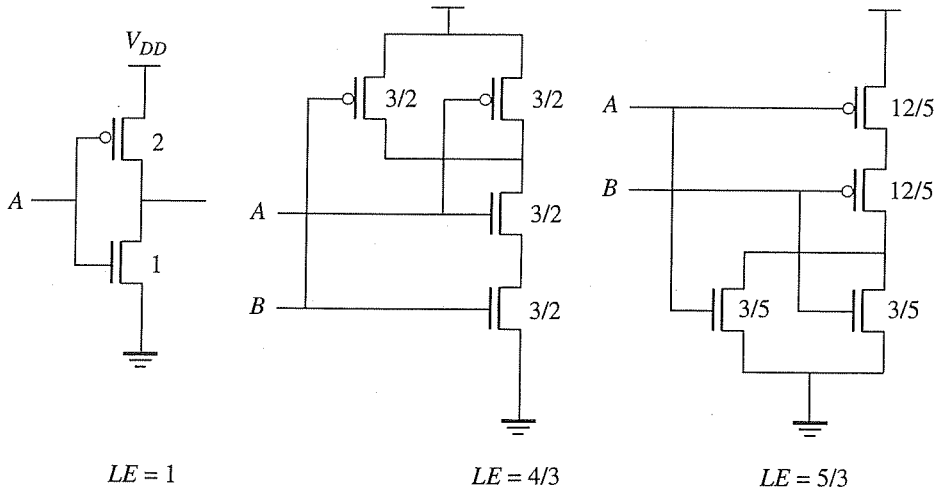


Figure 6.28
Logical effort using equal input capacitances.

The second approach is illustrated using Figure 6.28. Here, the device sizes of Figure 6.27 have been uniformly scaled such that the input capacitances of all three gates are equal. This is possible because the *LE* of a gate does not change if all devices are scaled uniformly. Since they all have the same input capacitance, we can take the ratio of the delays to obtain the *LE* values.

For the NAND gate, the two pull-down devices in series are 3/2 each. Their combination generates an equivalent device with size 3/4. The equivalent resistance is given by (4/3) R_{eff}. Therefore,

$$LE_{\text{nand}} = \frac{\left(\frac{4}{3} R_{eff}\right) C_{\text{out}}}{R_{eff} C_{\text{out}}} = \frac{4}{3}$$

For the NOR gate, the equivalent resistance of the pull-down device with size 3/5 is $(5/3)R_{eff}$. Therefore,

$$LE_{\text{nor}} = \frac{\left(\frac{5}{3} R_{eff}\right) C_{\text{out}}}{R_{eff} C_{\text{out}}} = \frac{5}{3}$$

In effect, we are taking the ratio of the driving resistance of gates that have equal input capacitance. This approach is somewhat more complicated than the first approach but may be used where the input capacitances of two gates are known to be equal.

The logical effort of the NOR gate is higher than the NAND gate. This implies that the NAND gate is better than a NOR gate in the context of logical effort; that

is, a lower *LE* is better than a higher *LE*. Recall that *LE* is evaluating a gate in terms of its output-drive and input-loading characteristics. The NAND gate is better in terms of its ability to drive an output while reducing the load on its input. In fact, we will be able to produce a shorter delay with the NAND compared to a NOR. Of the three gates, the inverter is actually the best gate to use, but it can only perform inversion. So when we have a choice, we should use NAND gates over NOR gates.

The *LE* for multi-input gates can also be calculated in the same manner. As an exercise, the reader should try to produce the results given in Table 6.1 for multi-input gates.

Table 6.1
Logical effort values of simple gates

Type of Gate	1 input	2 inputs	3 inputs	4 inputs
Inverter	1	—	—	—
NAND	—	4/3	5/3	6/3
NOR	—	5/3	7/3	9/3

The next step is to compute the parasitic term, *P*. Unfortunately, this term is technology- and gate-dependent. For a given technology, this term will have to be recomputed for all the different types of gates. Let's begin with a simple inverter and determine its *P* value. Recall that this term is given by

$$P = LE_{\text{inv}} \times \gamma_{\text{inv}} = LE \times \frac{C_{\text{self}}}{C_{\text{in}}} = LE \times \frac{C_{eff}3W}{C_g 3W} = LE \times \frac{C_{eff}}{C_g}$$

In effect, the value of *P* depends on the coefficient for junction capacitance and the coefficient for gate capacitance. Substituting in the *LE* and technology-dependent parameters, we obtain

$$P_{\text{inv}} = (1) \times \frac{1 \text{ fF}/\mu\text{m}}{2 \text{ fF}/\mu\text{m}} = \frac{1}{2}$$

For the NAND gate, we refer to Figure 6.27 to determine *P*. Accounting for shared source/drain regions in the layout:

$$P_{\text{nand}} = LE_{\text{nand}} \times \gamma_{\text{nand}} = LE \times \frac{C_{\text{self}}}{C_{\text{in}}}$$

$$= LE \times \frac{C_{eff}(2W + 2W + 2W)}{C_g(2W + 2W)} = LE \times \frac{C_{eff}}{C_g} \times \frac{3}{2} = \frac{4}{3} \times \frac{1}{2} \times \frac{3}{2} = 1$$

For the NOR gate, we again refer to Figure 6.27 to determine P. With proper accounting for shared source/drain regions in the layout:

$$P_{\text{nor}} = LE_{\text{nor}} \times \gamma_{\text{nor}} = LE \times \frac{C_{\text{self}}}{C_{\text{in}}}$$

$$= LE \times \frac{C_{eff}(W + 4W + 4W)}{C_g(W + 4W)} = LE \times \frac{C_{eff}}{C_g} \times \frac{9}{5} = \frac{5}{3} \times \frac{1}{2} \times \frac{9}{5} = \frac{3}{2}$$

The best way to accurately compute the value is through simulation. However, for the purposes of rapid hand calculations, we can use the following *approximate* table:

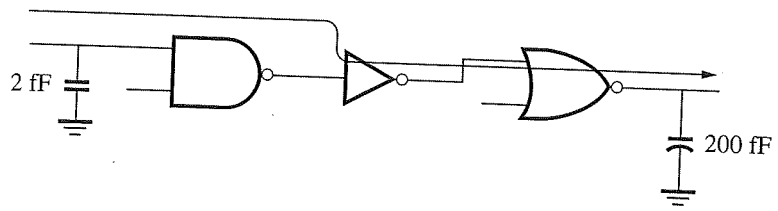
Table 6.2
Table of parasitic terms for simple gates

Type of Gate	1 input	2 inputs	3 inputs	4 inputs
Inverter	0.5	—	—	—
NAND	—	1	2	3
NOR	—	1.5	3	4.5

Path Optimization Using Logical Effort **Example 6.11**

Problem:

Repeat Example 6.10 using logical effort techniques. However, before specifying the sizes, compute the optimal delay.



Solution:

We need to equalize the $LE \times FO$ components of the delay for all gates. First compute the product of $LE \times FO$ for all the gates:

$$\text{total path effort} = LE_{\text{nand}} \times LE_{\text{inv}} \times LE_{\text{nor}} \times \left(\frac{C_{\text{load}}}{C_{\text{in}}}\right)$$

$$= (4/3)(1)(5/3)(200/2) = 222.2$$

Next, take the geometric mean of the result:

$$\text{Stage Effort} = \sqrt[3]{222.2} = 6$$

This is the fanout portion of the delay. The normalized path delay is three times this value (one for each gate) plus the parasitic term for each gate:

$$D = 3(6) + P_{\text{nand}} + P_{\text{inv}} + P_{\text{nor}}$$

$$= 18 + 1 + \left(\frac{1}{2}\right) + \left(\frac{3}{2}\right) = 21$$

The physical delay value is obtained by multiplying the normalized delay by $\tau_{\text{inv}} = 7.5$ ps:

$$\text{min\_path\_delay} = \tau_{\text{inv}} D = (7.5 \text{ ps})(21) = 157.5 \text{ ps}$$

Assuming that the delay is acceptable, we work backwards from the output to the input to compute the device sizes:

$$\therefore LE_{\text{nor}}\left(\frac{C_{\text{out}}}{C_{j+2}}\right) = 6 \Rightarrow LE_{\text{nor}}\left(\frac{C_{j+3}}{6}\right) = C_{j+2} = (5/3)(200 \text{ fF}/6) = 55 \text{ fF}$$

$$\therefore LE_{\text{inv}}\left(\frac{C_{j+2}}{C_{j+1}}\right) = 6 \Rightarrow LE_{\text{inv}}\left(\frac{C_{j+2}}{6}\right) = C_{j+1} = (1)(55/6) = 9.1 \text{ fF}$$

$$\therefore LE_{\text{nand}}\left(\frac{C_{j+1}}{C_{\text{in}}}\right) = 6 \Rightarrow LE_{\text{nand}}\left(\frac{C_{j+1}}{6}\right) = C_{\text{in}} = (4/3)(9.1/6) = 2 \text{ fF}$$

These are the same results as obtained in Example 6.10. The device sizing would proceed in the same manner as shown in that example.

It is interesting to note that we can determine the minimum delay without sizing the gates. This is one of the key advantages of the *LE* approach. If this minimum possible delay is not within specifications, the logic can be modified and the process repeated until a satisfactory solution is obtained. Once the target delay is achieved, gate sizing can be carried out.

6.6.2 Understanding Logical Effort

Further insight into logical effort can be obtained by examining the plot of normalized delay (D) versus electrical effort (FO) in Figure 6.29. The delay normalization is with respect to τ_{inv} while the electrical effort is the ratio of fanout capacitance to input capacitance. The slope of the inverter delay versus fanout is set to 1 by definition. The y-intercept is the parasitic term, which is $\frac{1}{2}$ for the inverter. The NAND gate has a y-intercept at 1 and a slope of 4/3, and the NOR gate has a y-intercept at 1.5 and a slope of 5/3. If we compare the delay of each gate with a fanout loading of 1, we see that the inverter is the fastest, the NAND is next and the NOR is the slowest.

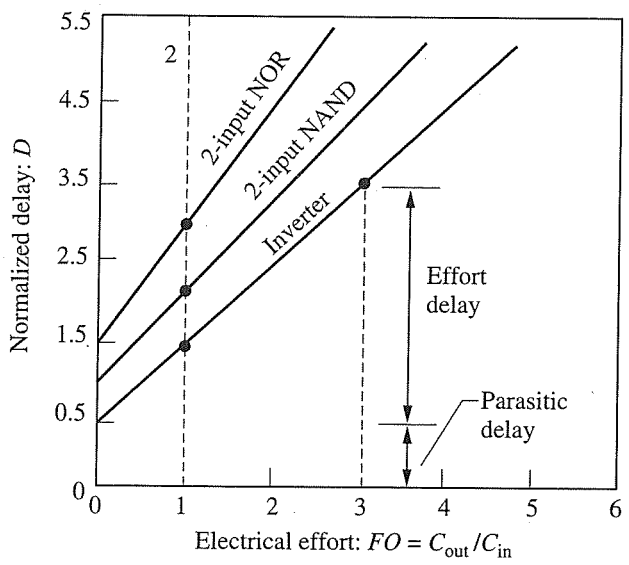


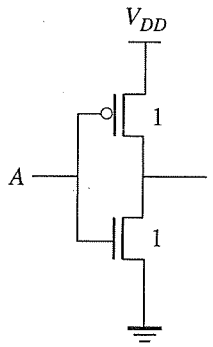
Figure 6.29
Practical interpretation of logical effort.

For improved accuracy of the *LE* method, these plots may be obtained directly from SPICE to determine more precise values of *LE* and *P* for a given technology.

***LE* for a Skewed Inverter** **Example 6.12**

Problem:

What is the *LE* of this inverter?



Solution:

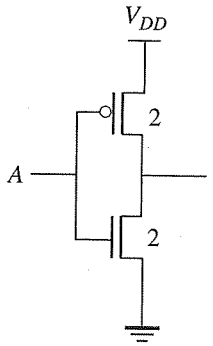
The delays are not the same and input capacitance is not the same as the reference inverter. One of the two must be adjusted to obtain the *LE*. We will show two methods of obtaining the *LE*. The rising and falling cases must be handled separately for this gate.

Method 1: Set the delays equal to that of the reference inverter and take the ratio of input capacitances.

Falling case—fall delay is already the same as the regular inverter:

$$LE_F = \frac{C_{in}|_{gate}}{C_{in}|_{inv}} = \frac{1+1}{1+2} = \frac{2}{3}$$

Rising case—scale up all devices by 2× to obtain the same rise delay:



$$LE_R = \frac{C_{in}|_{gate}}{C_{in}|_{inv}} = \frac{2+2}{1+2} = \frac{4}{3}$$

Method 2: Use the definition of logical effort:

$$LE = \frac{(R_{eff}C_{in})_{gate}}{(R_{eff}C_{in})_{inv}}$$

For the falling case:

$$LE_F = \frac{(R_{eqn})(2C_gW)}{(R_{eqn})(3C_gW)} = \frac{2}{3}$$

For the rising case:

$$LE_R = \frac{(2R_{eqn})(2C_gW)}{(R_{eqn})(3C_gW)} = \frac{4}{3}$$

Since the rising and falling *LE*s are different, we can optimize based on either one. But if we want to optimize both transitions, use the *average LE* to determine the transistor sizes.

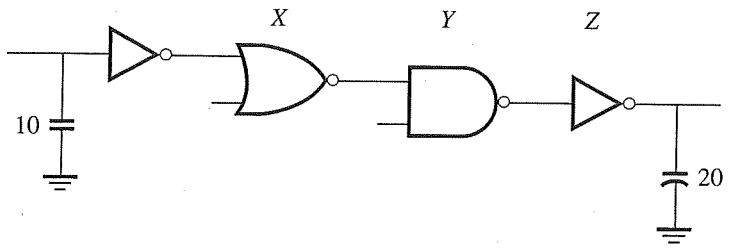
Therefore, the average logical effort for this gate is

$$LE = \frac{\frac{2}{3}+\frac{4}{3}}{2} = 1$$

Example 6.13 **Path Optimization Using Logical Effort**

Problem:

For the given logic circuit, determine the optimal stage effort, total path delay, and the sizes of the gates. Use normalized values of input and output capacitances, as given in the schematic, and compute the normalized delay and gate sizes.



Solution:

First compute the total path effort:

$$\text{Total path effort} = \prod(LE \times FO)$$

$$= 1\left(\frac{X}{10}\right) \times \frac{5}{3}\left(\frac{Y}{X}\right) \times \frac{4}{3}\left(\frac{Z}{Y}\right) \times 1\left(\frac{20}{Z}\right)$$

$$= \frac{20}{9} \times \frac{20}{10} = \frac{400}{90}$$

In this example, we sequenced through each stage and computed the $LE \times FO$. Then the total path effort was obtained by multiplying these factors together. Another way to obtain the same answer is to multiply the *LE*s together (to obtain 20/9), then multiply the *FO*s together (to obtain 20/10), and finally multiply these two terms together (to obtain 400/90). This approach works only for single paths with no branches.

Next compute the optimal stage effort, SE^*, by taking the geometric mean of the above result.

$$SE^* = \text{Optimal stage effort} = \left(\frac{400}{90}\right)^{1/4} = 1.45$$

We take the fourth root of the path effort since there are four stages in this example. This is the optimal stage effort that we assign to all gates in the path. The total path delay can be computed as

$$D = 4(SE^*) + 2P_{\text{inv}} + P_{\text{nor}} + P_{\text{nand}}$$

$$= 4(1.45) + 1.0 + 1.5 + 1.0 = 9.3$$

Note that the optimal stage effort, hence the optimal solution, is completely determined by the input and output capacitances, and the number and type of logic gates in the path of interest. The optimal value of the total path delay is known before the gate sizes have even been determined.

The actual sizes are obtained by working backwards from the output to the input. We use the fact that for each gate,

$$SE^* = LE \times FO = LE \times \frac{C_{\text{out}}}{C_{\text{in}}}$$

$$\therefore C_{\text{in}} = LE \times \frac{C_{\text{out}}}{SE^*}$$

Therefore, working backwards from output to input:

$$Z = (1) \times (20)/(1.45) = 13.8$$

$$Y = (4/3) \times Z/(1.45) = 12.7$$

$$X = (5/3) \times Y/(1.45) = 14.5$$

$$C_{in} = (1)(14.5)/1.45 = 10$$

The first inverter size can be verified by the last equation, which is consistent with the specified input capacitance value. This is a good way to confirm that your sizing is correct.

Example 6.14 **Designing an 8-Input AND Gate**

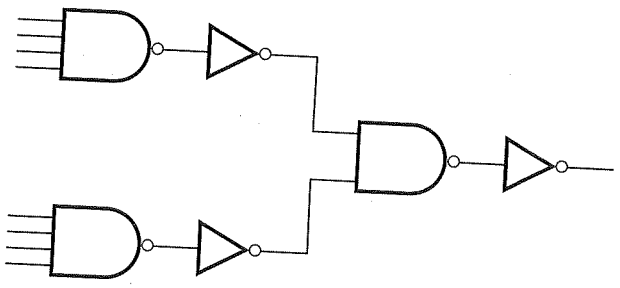
Problem:

An 8-input AND gate is to be designed to drive a load of 200 fF but is limited to an input capacitance of 20 fF. Since a single 8-input CMOS NAND gate is out of the question, choose two configurations that are more suitable and identify the solution with the fastest speed.

Solution:

1. The first option is the NAND4-INV-NAND2-INV as shown below. Here the *LE* and parasitic terms are all known for these gates since they are standard gates. So the following computation can be used to determine the delay:

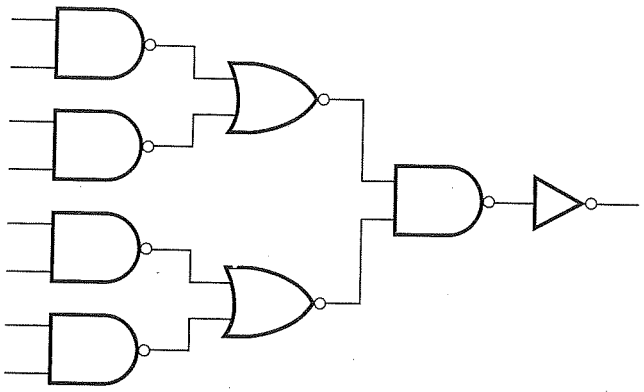
$$D = 4(\text{Path Effort})^{1/4} + \sum P = 4((2)(4/3)(10))^{1/4} + 3 + 1/2 + 1 + 1/2 = 14$$



2. Next, a NAND2-NOR2-NAND2-INV cascade is examined. Again, the *LE* and γ values are known. Therefore,

$$D = 4(\text{Path Effort})^{1/4} + \sum P$$

$$= 4((4/3)^2(5/3)(10))^{1/4} + 1 + 3/2 + 1 + 1/2 = 13.3$$



Option 2 is better than option 1.

We can make a few conclusions based on these results. The *LE* of the gates in a logic path, by themselves, do not tell us which configuration is faster. A delay calculation must be performed to make the determination. This is because the delay depends on the fanout capacitance. If the fanout capacitance is small, the parasitic terms will influence the answer. If the fanout is very large, then the delay is a function of the types of gates and the number of stages used. In practice, only single stage gates driving large capacitances can be compared purely on *LE*. Therefore, we should always use delay calculation to determine if one configuration is superior to another.

6.6.3 Branching Effort and Sideloads

So far we have considered a single path through the logic circuit. What if there are one or more branches from a given node? This would affect the sizing of gates and therefore must be taken into account in logical effort. For this purpose, we introduce a *branching factor* into the analysis. Alternatively, if a node has an additional load of fixed size, we treat it as a *sideload*. These methods are described below.

The first situation is one where the branch is identical to the path of interest. In Figure 6.30, we have an inverter driving two other inverters. This doubles the loading on the first inverter since both are sized in the same way. We are interested in

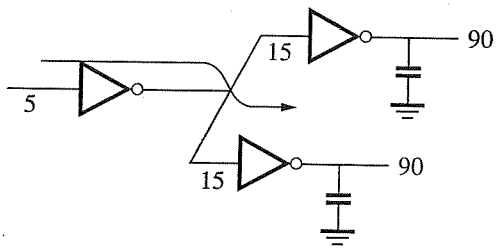


Figure 6.30
Branching factor.

optimizing the highlighted path, but we would like to optimize both paths at the same time. In fact, we need to introduce a new term called the branching effort which accounts for the branches emanating from the path of interest. In this circuit, there is a branching effort of 2 in the critical path since we drive two identical inverters.

The equation for the total path effort, including branches, is as follows:

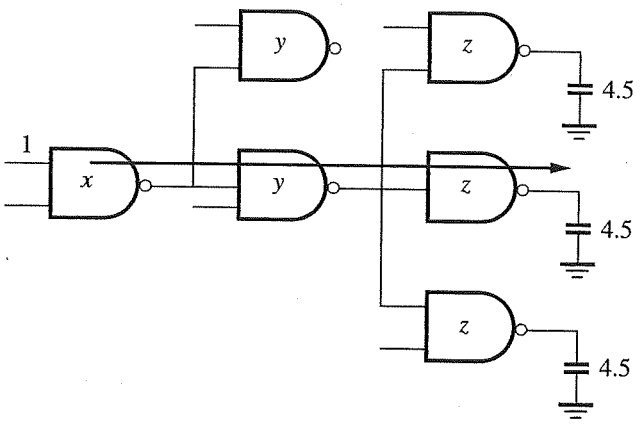
$$\text{total\_path\_effort} = \prod(LE \times BE \times FO) = \prod(LE \times BE) \times \frac{C_{\text{load}}}{C_{\text{in}}}$$

That is, the total path effort is the product of the logical efforts (LE) times the product of the branching efforts (BE) times the product of the fanouts (FO). The branching effort is often known in advance or can be estimated based on the circuit topology. The use of BE is best illustrated in an example.

Example 6.15 **Branching Effort Example**

Problem:

Select gate sizes y and z to minimize delay in the highlighted path:



Solution:

Logical Effort: $LE_p = (4/3)^3$

Electrical Effort: $FO_p = C_{\text{out}}/C_{\text{in}} = 4.5$

Branching Effort: $BE_p = (2)(3) = 6$

Path Effort: $PE = (LE_p)(BE_p)(FO_p) = 64$

Optimal Stage Effort: $SE^* = (PE)^{1/3} = 4$

Delay: $D = (N)(SE^*) + \text{Parasitics}$

Delay: $D = (3)(4) + (3)(1) = 15$

Work backwards for gate sizes:

$z = (1)(4.5)(4/3)/4 = 1.5$

$y = (3)(1.5)(4/3)/4 = 1.5$

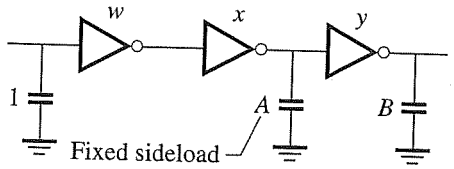
$x = (2)(1.5)(4/3)/4 = 1$

Another situation arises where we have a known fixed load in a circuit along the path of interest. The notion of branching effort cannot be used here. The reason is that branching effort should only be applied where the other fanouts uniformly scale in size with the path of interest. This does not handle the case where a node has a fixed capacitance value. We refer to these capacitances as *sideloads*. The approach is best illustrated using an example.

Computing Delay with Sideloads **Example 6.16**

Problem:

Compute the gate sizes w, x, and y for the following logic circuit to produce the minimum delay. Assume that $A = 8$ and $B = 64$. Here, A should be considered as a sideload.



Solution:

If we remove the sideload and use FO4 sizing rules, we would size the inverters as $w = 1, x = 4$, and $y = 16$. This satisfies the geometric relationship between consecutive stages and would produce the minimum delay.

If we now insert the sideload, we would derive the following delay equations

$$\text{First two stages: } D_1 = LE_{\text{inv}}\left(\frac{x}{w} + \gamma_{\text{inv}}\right) + LE_{\text{inv}}\left(\frac{A + y}{x} + \gamma_{\text{inv}}\right)$$

$$\frac{\partial D_1}{\partial x} = \frac{1}{w} - \frac{A + y}{x^2} \Rightarrow \frac{x}{w} = \frac{A + y}{x}$$

$$\text{Next two stages: } D_2 = LE_{\text{inv}}\left(\frac{A + y}{x} + \gamma_{\text{inv}}\right) + LE_{\text{inv}}\left(\frac{B}{y} + \gamma_{\text{inv}}\right)$$

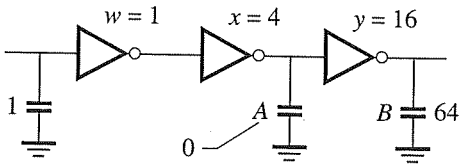
$$\frac{\partial D_2}{\partial y} = \frac{1}{x} - \frac{B}{y^2} \Rightarrow \frac{y}{x} = \frac{B}{y}$$

We can use the two equations to iterate the solution:

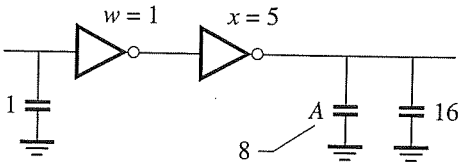
$$y^2 = Bx \rightarrow y = \sqrt{Bx} = \sqrt{64(4)} = 16$$
$$x^2 = (A + y)w \rightarrow x = \sqrt{(A + y)w} = 4.9$$
$$y = 17.7 \qquad x = 5.1$$
$$y \approx 18 \qquad x \approx 5$$

Setting up the iterative equations is complicated and prone to error. An alternative approach is recommended that is much simpler and produces a reasonable solution. First solve the sizing problem using logical effort without the sideload. Next, add in the sideload and the loading of the next gate. Then, remove the downstream gates and solve the remaining circuit again with logical effort to obtain their sizes.

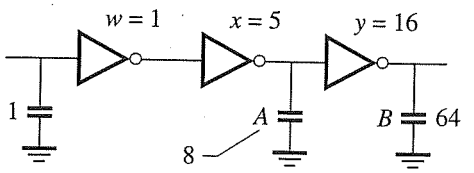
Solve without sideload



Add sideload. Remove gates beyond sideload by including their loading effects. Solve subproblem



Combine two solutions



The results obtained are slightly different from the exact solution but produce a similar delay. More importantly, it is much easier to carry out hand calculation using this method compared to the iterative approach.

It should be noted that logical effort is a useful tool for back of the envelope timing optimization. It allows the designer to quickly determine the optimal delay for a given path, and the corresponding device sizes to achieve this delay. It provides insight into the key factors in the delay of logic paths. There are many other aspects of logical effort that were not covered in this chapter. The reader is encouraged to read further in the references. The concepts outlined here will be extended in the chapters to follow.

6.7 Summary

Propagation delay:

$$t_P = \frac{t_{PLH} + t_{PHL}}{2}$$

Driving resistance:

$$R_{eqn} \approx 12.5\ \text{k}\Omega/\square$$

$$R_{eqp} \approx 30\ \text{k}\Omega/\square$$

$$R_N = R_{eqn}\left(\frac{L}{W}\right)_n$$

$$R_P = R_{eqp}\left(\frac{L}{W}\right)_p$$

Load capacitance components:

$$C_{\text{load}} = C_{\text{self}} + C_{\text{wire}} + C_{\text{fanout}}$$

Input capacitance:

$$C_g = C_{ox}L + 2C_{ol} \cong 2\ \text{fF}/\mu\text{m}$$

Self-capacitance:

$$C_{eff} = C_j + 2C_{ol} \cong 1\ \text{fF}/\mu\text{m}$$

Wire capacitance:

$$C_{\text{int}} = 0.2\ \text{fF}/\mu\text{m}$$

Total delay:

$$\text{Total\_delay} = \sum_i \text{R}_i\text{C}_i$$

Inverter delay equation:

$$t_{\text{delay}} = \tau_{\text{inv}}\left[\frac{C_{\text{out}}}{C_{\text{in}}} + \gamma_{\text{inv}}\right]$$

where

$$\tau_{\text{inv}} = R_{eff}C_{\text{in}} = R_{eqn}\left(\frac{L_n}{W_n}\right)C_g(3W_n) = 3R_{eqn}C_gL_n$$

$$\gamma_{\text{inv}} = \frac{C_{\text{self}}}{C_{\text{in}}}$$

NAND and NOR intrinsic time constants:

$$\tau_{\text{nand}} = R_{eff}C_{\text{in}} = R_{eqn}\left(\frac{L_n}{W_n}\right)4W_nC_g = 4R_{eqn}C_gL_n$$

$$\tau_{\text{nor}} = R_{eff}C_{\text{in}} = R_{eqn}\left(\frac{L_n}{W_n}\right)5W_nC_g = 5R_{eqn}C_gL_n$$

Normalized delay equation:

$$D = \sum (LE \times FO + P)$$

where

LE = logic effort = $\tau_{\text{gate}} / \tau_{\text{inv}}$ (normalized to inverter)

FO_j = fanout = $\dfrac{C_{j+1}}{C_j}$

P = parasitic term = LE $\times \gamma_{\text{gate}}$

Logical effort for inverter, NAND2, and NOR2:

$$LE_{\text{inv}} = \frac{\tau_{\text{inv}}}{\tau_{\text{inv}}} = 1$$

$$LE_{\text{nand2}} = \frac{\tau_{\text{nand2}}}{\tau_{\text{inv}}} = \frac{4}{3}$$

$$LE_{\text{nor2}} = \frac{\tau_{\text{nor2}}}{\tau_{\text{inv}}} = \frac{5}{3}$$

Path effort:

$$\text{total\_path\_effort} = \prod(LE \times BE \times FO) = \prod(LE \times BE) \times \frac{C_{\text{load}}}{C_{\text{in}}}$$

Optimal stage effort:

$$SE^* = (\text{Path Effort})^{1/N}$$

Gate sizing based on optimal stage effort:

$$C_{\text{in}} = LE \times \frac{C_{\text{out}}}{SE^*}$$

Normalized delay:

$$D = N\,(\text{Path Effort})^{1/N} + \sum P$$

REFERENCES

1. I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kaufman Publishers, San Francisco, CA, 1999.
2. D. Harris, *Skew-Tolerant Circuit Design*, Morgan Kaufman Publishers, San Francisco, CA, 2001.
3. J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Designer Perspective*, Second Edition, Prentice-Hall, Upper Saddle River, NJ, 2003.
4. S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits, Analysis and Design*, Third Edition, McGraw-Hill, New York, NY, 2003.
5. H. Veendrick, *Deep-Submicron CMOS ICs*, Second Edition, Kluwer Academic Publishers, Boston, MA, 2000.
6. J. P. Uyemura, *CMOS Logic Circuit Design*, Kluwer Academic Publishers, Boston, MA, 1999.

PROBLEMS

P6.1. Plot the on-resistance of a unit-sized NMOS device as a function of V_{DS}. Use 0.13 μm technology parameters and assume that $V_{DD} = 1.2$ V. How does this plot compare to the design value of 12.5 kΩ? Derive an expression for the average resistance value between V_{DD} to $V_{DD}/2$ using the velocity saturated model.

P6.2. Compute the oscillation frequency of a seven-stage ring oscillator using 0.13 μm technology parameters. Does the size of the inverters make any difference in the result? Compare your result with SPICE.

P6.3. Extract values of R_{eqn}, R_{eqp}, C_g, and C_{eff} using SPICE for a 0.13 μm technology. How do they compare against the values used for hand calculation?

P6.4. In each of the circuits of Figure P6.4, determine the self-capacitance at the output assuming step changes at the inputs shown.

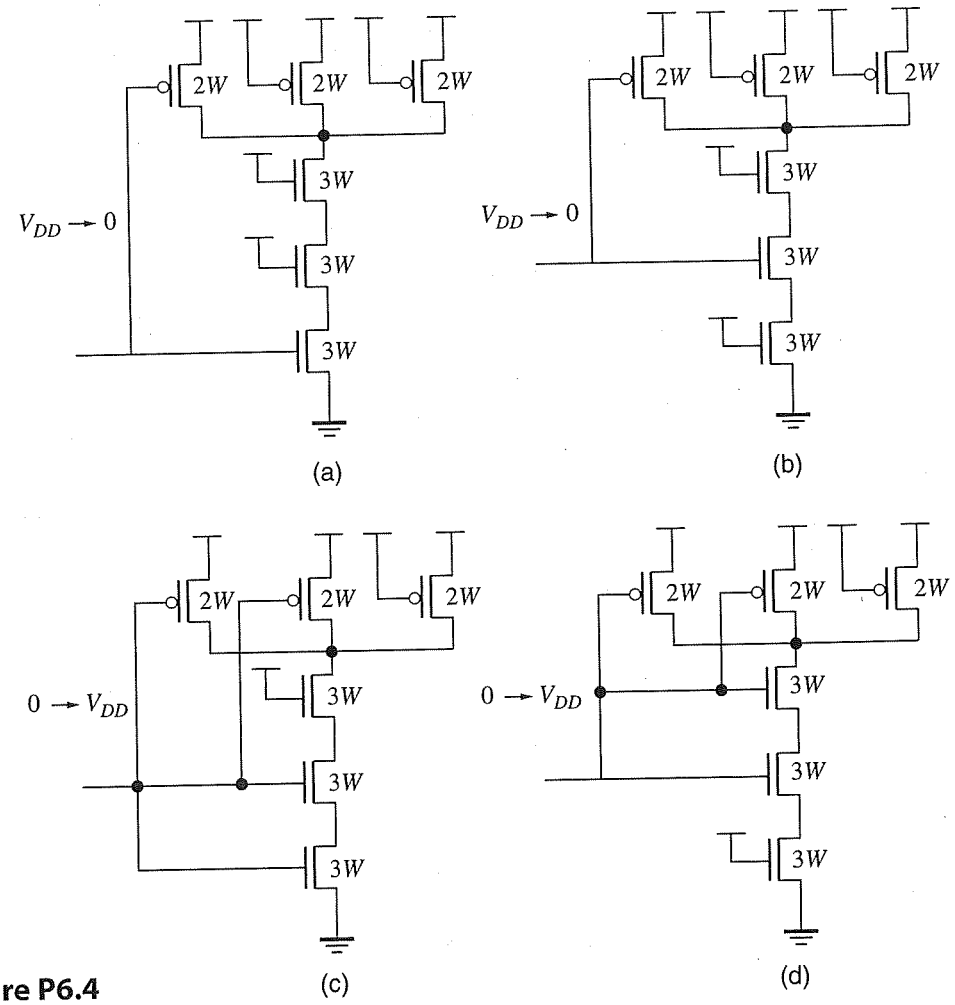


Figure P6.4

P6.5. In this problem, we examine the effect of the input selected and the velocity saturation model on the delay for the two circuits of Figure P6.5.

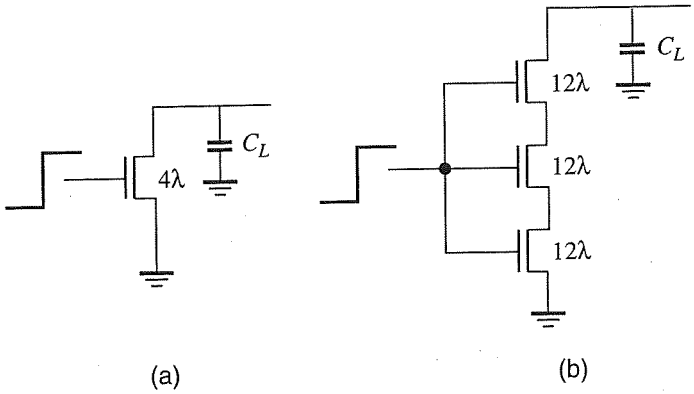


Figure P6.5

(a) Using HSPICE and a 0.13 μm technology file, compare the t_{PHL} delay for the two cases. The output capacitance is 50 fF. The input should be a step going from 0 V to 1.2 V. You can ignore parasitics in your simulation by setting *AS*, *AD*, *PS*, and *PD* to zero in the netlist. Which case has a shorter delay and why?

(b) Now, adjust the sizes of all three devices in circuit (b) until the delay for circuit (a) and (b) are the same. By what factor is the device size reduced in case (b)?

P6.6. Without adding additional inverters, size the inverters in Figure P6.6 to minimize the propagation delay, and then calculate the path delay. Next, calculate the number of devices in the path for optimum delay and recalculate the size of devices. Finally calculate the new path delay and compare it with the first case. C_{inv} is the input capacitance of a minimum size inverter.

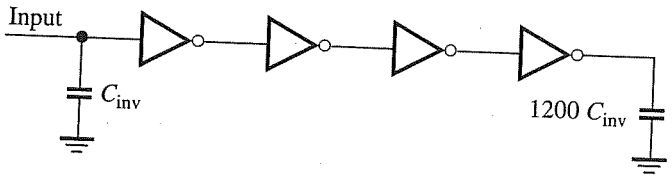


Figure P6.6

P6.7. Show that the logical effort of a gate does not change as you uniformly increase all the device sizes. Demonstrate this with a NAND3 and a NOR4 circuit.

P6.8. One way to improve the power dissipation of pseudo-NMOS gates is to connect one of the inputs to the pull-up device as shown in Figure P6.8. If we connect input *A* to the pull-up, we can reduce the size of the NMOS device to *W*/2. This configuration behaves as a static CMOS gate if the input comes through node *A*, but like the pseudo-NMOS gate if

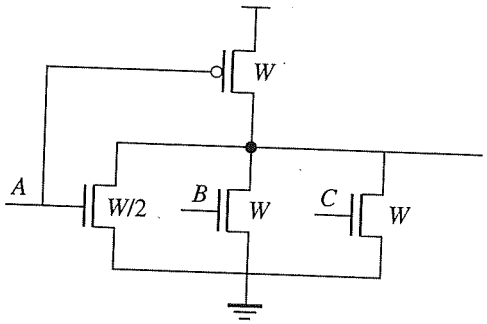


Figure P6.8

the input comes through B or C. Therefore, the power dissipation is smaller if node A goes high but will be the same as a pseudo-NMOS gate if B or C go high. Clearly, this type of configuration is suitable if we knew that A would arrive first.

(a) What is the LE of input A?

(b) What is the LE of inputs B and C?

P6.9. Calculate the logical effort of the gates in Figure P6.9. (All transistors have minimum length, $L = 2\lambda$). Use input A for the calculation.

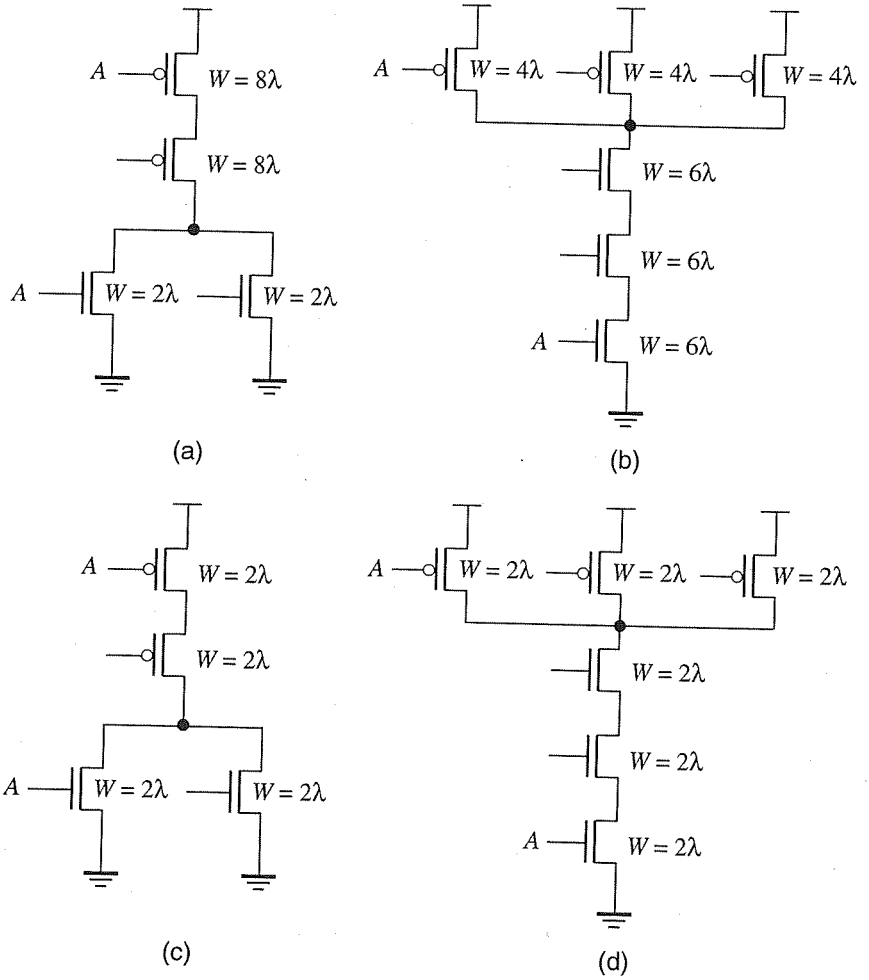


Figure P6.9

P6.10. What is the rising LE_R at input B of the first gate, and falling LE_F of the subsequent gate in Figure P6.10?

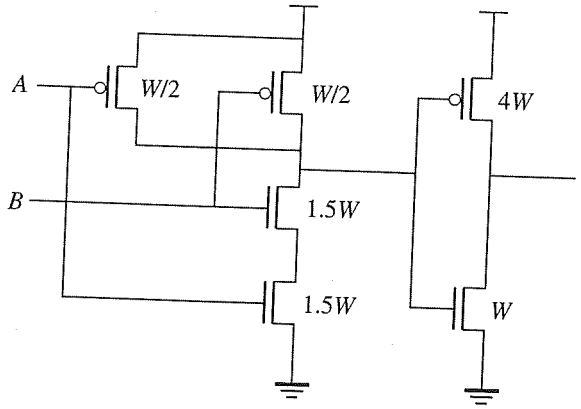


Figure P6.10

P6.11. In Figure P6.11, calculate the optimum delay and size of the transistors. (All devices are standard CMOS and all transistors have minimum length, $L = 0.1\ \mu m$). C_{inv} is the input capacitance of a minimum size inverter.

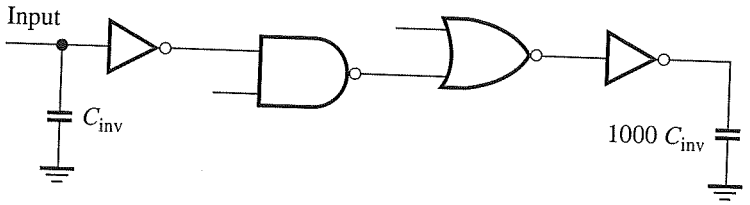


Figure P6.11

P.6.12. In Figure P6.12, calculate the optimum path delay and the transistor sizes. (All devices are standard CMOS and all transistors have minimum length, $L = 0.1\ \mu m$). Use 0.13 μm technology parameters. C_{inv} is the input capacitance of a minimum size inverter.

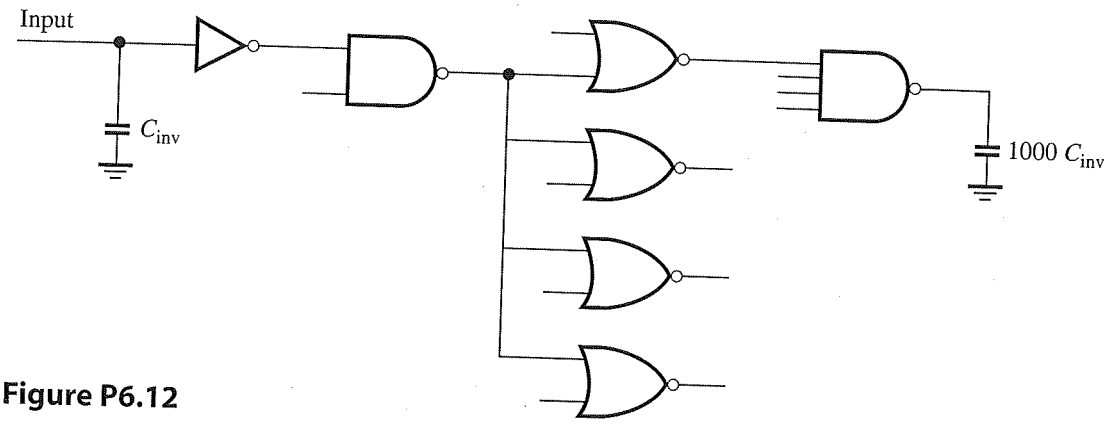


Figure P6.12

P6.13. Without adding additional inverters, size the gates in Figure P6.13 to minimize the propagation delay, and then calculate the path delay. How many inverters should be added (and where) to produce the optimal delay?

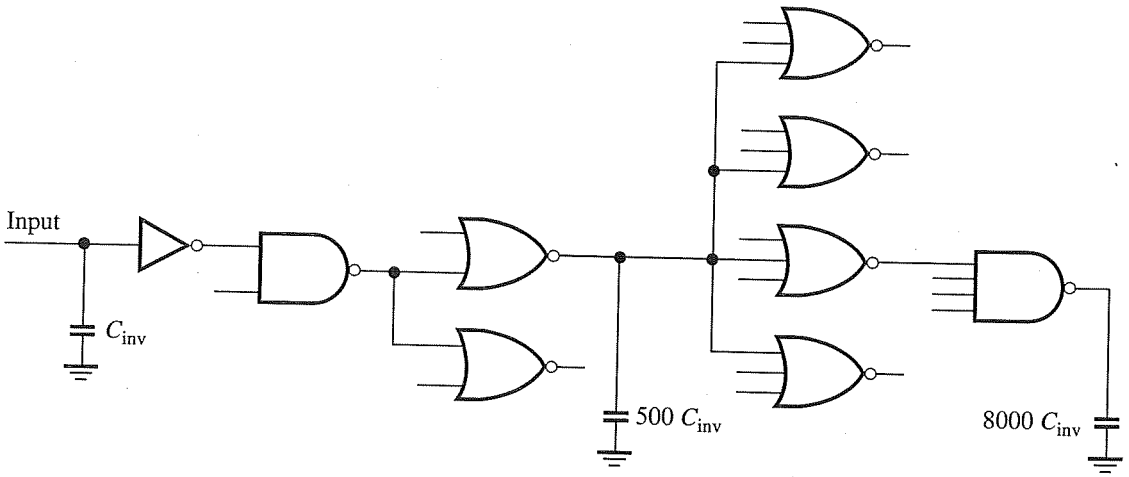


Figure P6.13

P6.14. In Example 6.2, it was found that the average C_g is larger for a low to high transition compared to the value for a high to low transition. Explain why this is always true.

P6.15. When ramp inputs are applied to an inverter, the average value of C_g is larger than when a step input is applied. Explain why this is always true.

P6.16. Derive a simple formula for the FO4 delay that can be used for back of the envelope calculations. It should have the form:

$$\text{FO4 delay} = \text{constant} \times L$$

where the constant is in units of picoseconds/μm and L is the minimum channel length. Compute the constant based on 0.18 μm and 0.13 μm technologies. Use the formula to estimate the FO4 delay for L = 90 nm, 45 nm, and 22 nm.