3

Logic Gates

Highlights:

Combinational logic.

Static logic gates.

Delay and power.

Alternate gate structures: switch, domino, etc.

Wire delay models.

3.1 Introduction

This chapter concentrates on the design of combinational logic functions. The knowledge gained in the last chapter on fabrication is important for combinational logic design—technology-dependent parameters for minimum size, spacing, and parasitic values largely determine how big a gate circuit must be and how fast it can run. We will start by reviewing some important facts about combinational logic functions. The first family of logic gate circuits we will consider are **static, fully complementary** gates, which are the mainstay of CMOS design. We will analyze the properties of these gates in detail: speed, power consumption, layout design, testability. We will also study some more advanced circuit families—pseudo-nMOS, DCVS, domino, and low-power gates—that are important in special design situations. We will also

study the delays through wires, which can be much longer than the delays through the gates.

3.2 Combinational Logic Functions

First, it is important to distinguish between *combinational logic expressions* and *logic gate networks*. A combinational logic expression is a mathematical formula which is to be interpreted using the laws of Boolean algebra: given the expression $a + b$, for example, we can compute its truth value for any given values of a and b; we can also evaluate relationships such as $a + b = c$. A logic gate computes a specific Boolean function, such as $(a + b)'$. The goal of logic design or optimization is to find a network of logic gates which together compute the combinational logic function we want. Logic optimization is interesting and difficult for two reasons:

- We may not have a logic gate for every possible function, or even for every function of n inputs. It therefore may be a challenge to rewrite our combinational logic expression so that each term represents a gate.
- Not all gate networks that compute a given function are alike—networks may differ greatly in their area and speed. We want to find a network that satisfies our area and speed requirements, which may require drastic restructuring of our original logic expression.

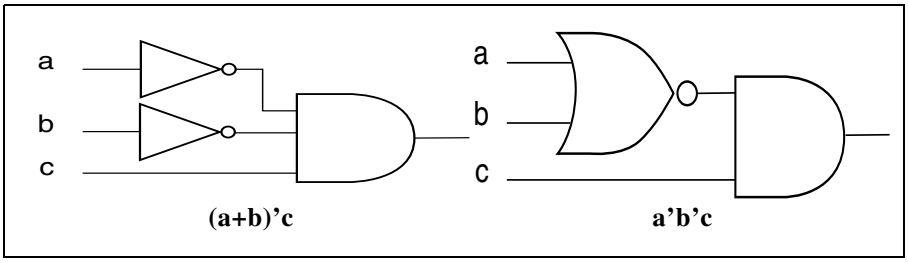


Figure 3-1: *Two logic gate implementations of a Boolean function.*

Figure 3-1 illustrates the relationship between logic expressions and gate networks. The two expressions are logically equivalent: $(a + b)'c = a'b'c$. We have shown a logic gate network for each expression which directly implements each function—each term in the expression becomes a gate in the network. The two logic networks have very different structures. Which is best depends on the requirements—the relative importance of area and delay—and the characteristics of the technology. But we

must work with both logic expressions and gate networks to find the best implementation of a function, keeping in mind the relationships:

- combinational logic expressions are the specification;
- logic gate networks are the implementation;
- area, delay, and power are the costs.

We will use fairly standard notation for logic expressions: if a and b are variables, then a' (or $\bar{a}$) is the complement of a, $a \cdot b$ (or ab) is the AND of the variables, and $a + b$ is the OR of the variables. In addition, for the NAND function (ab)' we will use the | symbol[1], for the NOR function ($a + b$)' we will use *a NOR b*, and for exclusive-or ($a \ XOR \ b = ab' + a'b$) we will use the $\oplus$ symbol. (Students of algebra know that XOR and AND form a ring.) We use the term **literal** for either the true form (a) or complemented form (a') of a variable. Understanding the relationship between logical expressions and gates lets us study problems in the model that is simplest for that problem, then transfer the results. Two problems that are of importance to logic design but easiest to understand in terms of logical expressions are **completeness** and **irredundancy**.

A set of logical functions is complete if we can generate every possible Boolean expression using that set of functions—that is, if for every possible function built from arbitrary combinations of +, ·, and ', an equivalent formula exists written in terms of the functions we are trying to test. We generally test whether a set of functions is complete by inductively testing whether those functions can be used to generate all logic formulas. It is easy to show that the NAND function is complete, starting with the most basic formulas:

- 1: $a|(a|a) = a|a' = 1$.
- 0: $\{a|(a|a)\}|\{a|(a|a)\} = 1|1 = 0$.
- a': $a|a = a'$.
- ab: $(a|b)|(a|b) = ab$.
- $a + b$:$(a|a)|(b|b) = a'|b' = a + b$.

1. The Scheffer stroke is a dot with a negation line through it. C programmers should note that this character is used as OR in the C language.

From these basic formulas we can generate all the formulas. So the set of functions {|} can be used to generate any logic function. Similarly, any formula can be written solely in terms of NORs.

The combination of AND and OR functions, however, is not complete. That is fairly easy to show: there is no way to generate either 1 or 0 directly from any combination of AND and OR. If NOT is added to the set, then we can once again generate all the formulas: $a + a' = 1$, etc. In fact, both {', ·} and {',+} are complete sets.

Any circuit technology we choose to implement our logic functions must be able to implement a complete set of functions. Static, complementary circuits naturally implement NAND or NOR functions, but some other circuit families do not implement a complete set of functions. Incomplete logic families place extra burdens on the logic designer to ensure that the logic function is specified in the correct form.

A logic expression is irredundant if no literal can be removed from the expression without changing its truth value. For example, $ab + ab'$ is redundant, because it can be reduced to a. An irredundant formula and its associated logic network have some important properties: the formula is smaller than a logically equivalent redundant formula; and the logic network is guaranteed to be testable for certain kinds of manufacturing defects. However, irredundancy is not a panacea. Irredundancy is not the same as **minimality**—there are many irredundant forms of an expression, some of which may be smaller than others, so finding one irredundant expression may not guarantee you will get the smallest design. Irredundancy often introduces added delay, which may be difficult to remove without making the logic network redundant. However, simplifying logic expressions before designing the gate network is important for both area and delay. Some obvious simplifications can be done by hand; CAD tools can perform more difficult simplifications on larger expressions.

3.3 Static Complementary Gates

This section concentrates on one family of logic gate circuits: the static complementary gate. These gates are static because they do not depend on stored charge for their operation. They are complementary because they are built from complementary (dual) networks of p-type and n-type transistors. The important characteristics of a logic gate circuit are its layout area, delay, and power consumption. We will concentrate our analysis on the inverter because it is the simplest gate to analyze and its analysis extends straightforwardly to more complex gates.

3.3.1 Gate Structures

A static complementary gate is divided into a **pullup network** made of p-type transistors and a **pulldown network** made of n-type transistors. The gate's output can be connected to V_{DD} by the pullup network or V_{SS} by the pulldown network. The two networks are complementary to ensure that the output is always connected to exactly one of the two power supply terminals at any time: connecting the output to neither would cause an indeterminate logic value at the output, while connecting it to both would cause not only an indeterminate output value, but also a low-resistance path from V_{DD} to V_{SS}. The structures of an inverter, a two-input NAND gate, and a two-input NOR gate are shown in Figure 3-2, Figure 3-3, and Figure 3-4, respectively; + stands for V_{DD} and the triangle stands for V_{SS}. Inspection shows that they satisfy the complementarity requirement: for any combination of input values, the output value is connected to exactly one of V_{DD} or V_{SS}.

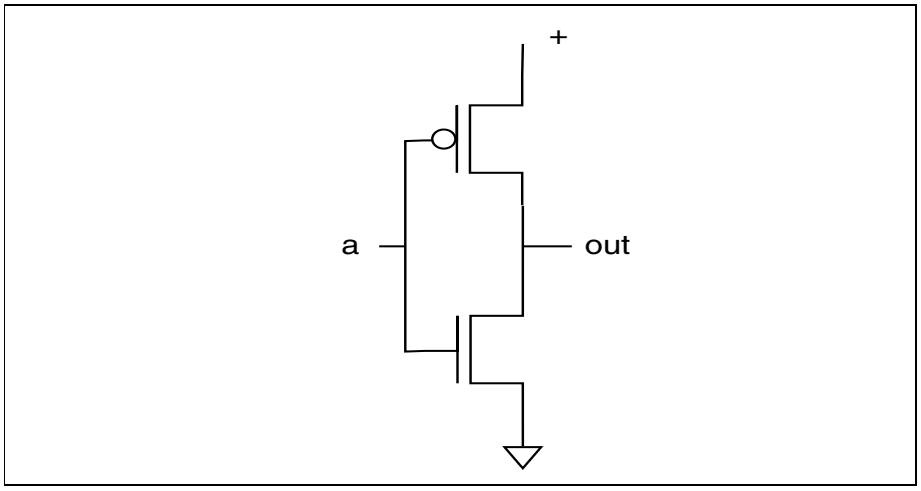


Figure 3-2: *Transistor schematic of a static complementary inverter.*

Gates can be designed for functions other than NAND and NOR by designing the proper pullup and pulldown networks. Networks that are series-parallel combinations of transistors can be designed directly from the logic expression the gate is to implement. In the pulldown network, series-connected transistors or subnetworks implement AND functions in the expression and parallel transistors or subnetworks implement OR functions. The converse is true in the pullup network because p-type transistors are off when their gates are high. Consider the design of a two-input NAND gate as an example. To design the pulldown network, write the gate's logic expression to have negation at the outermost level: (*ab*)' in the case of the NAND. This expression specifies a series-connected pair of n-type transistors. To design the pullup network, rewrite the expression to have the inversion pushed down to the

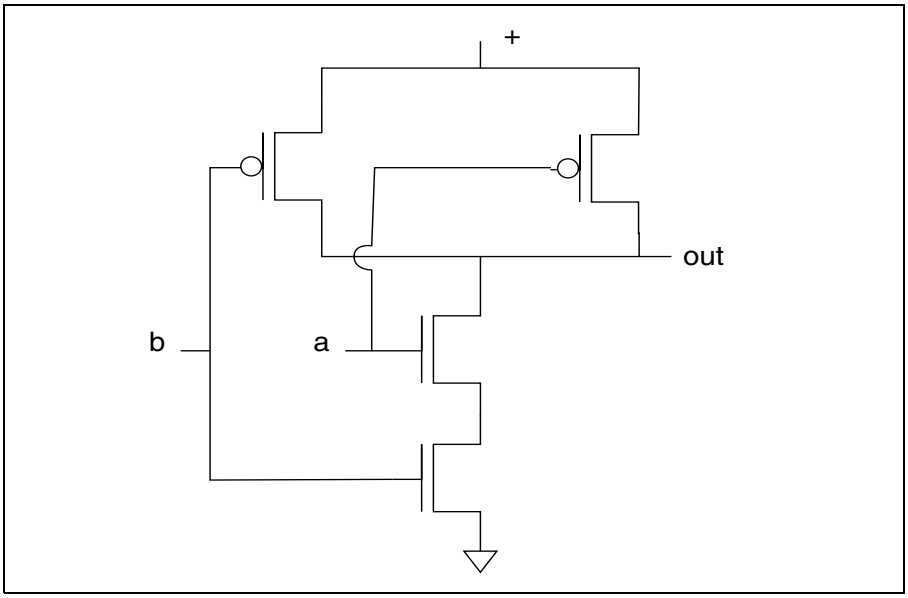


Figure 3-3: *A static complementary NAND gate.*

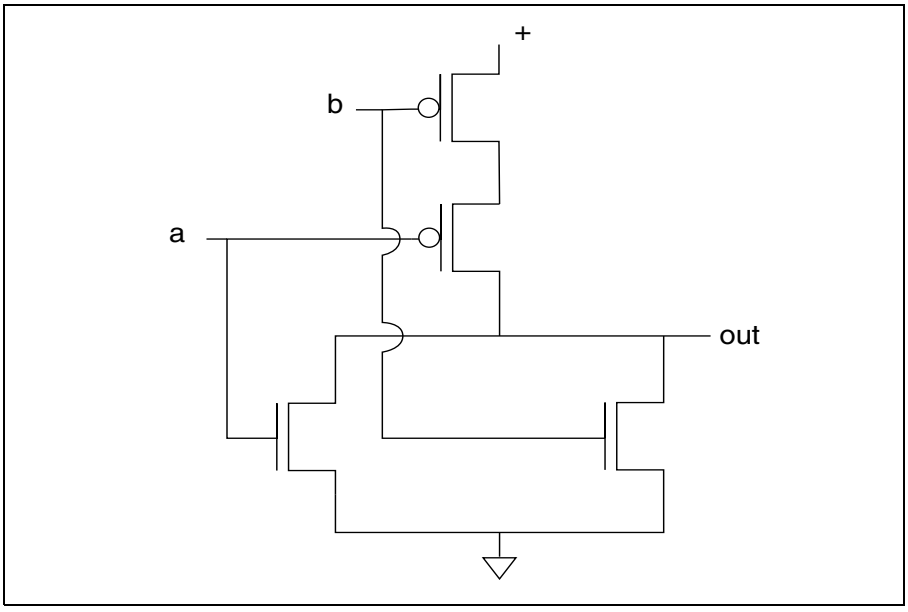


Figure 3-4: *A static complementary NOR gate.*

innermost literals: *a*' + *b*' for the NAND. This expression specifies a parallel pair of p-type transistors, completing the NAND gate design of Figure 3-3. Figure 3-5 shows the topology of a gate which computes [*a*(*b*+*c*)]': the pulldown network is given by

the expression, while the rewritten expression $a' + (b'c')$ determines the pullup network.

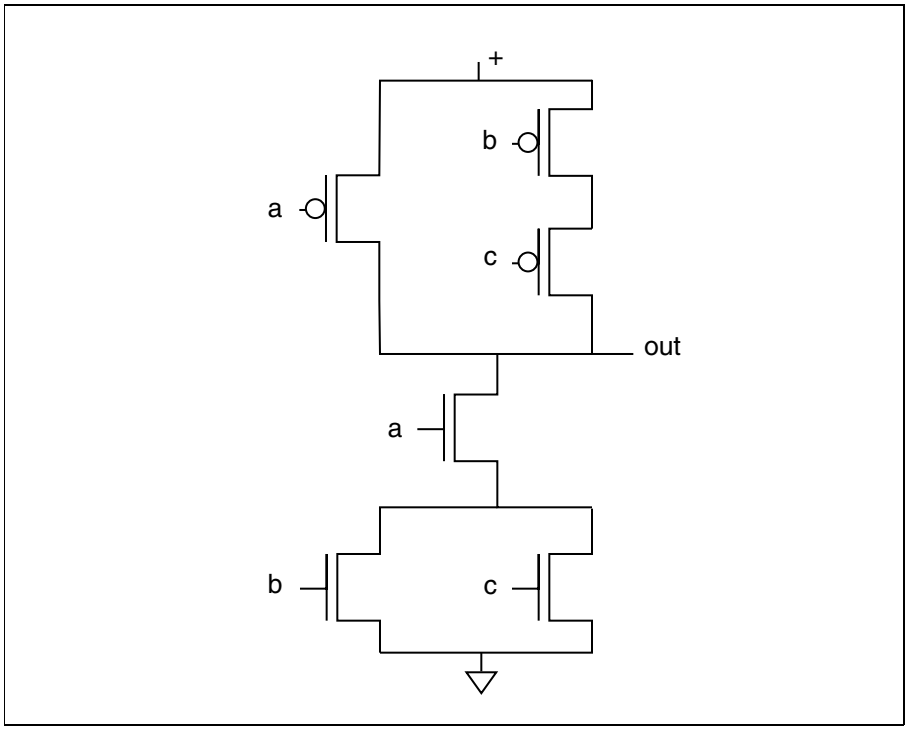


Figure 3-5: *A static complementary gate that computes [a(b+c)]'.*

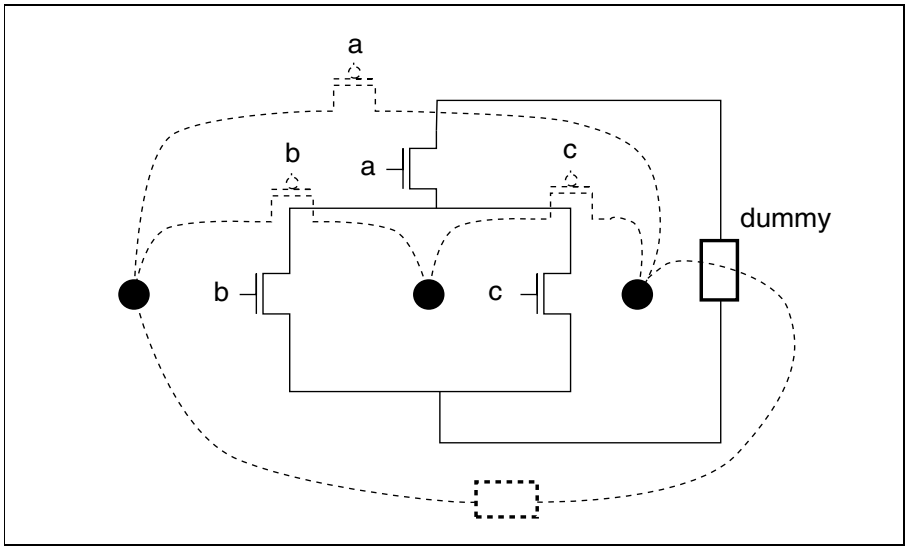


Figure 3-6: *Constructing the pullup network from the pulldown network.*

You can also construct the pullup network of an arbitrary logic gate from its pulldown network, or vice versa, because they are **duals**. Figure 3-6 illustrates the dual construction process using the pulldown network of Figure 3-5. First, add a dummy component between the output and the V_{SS} (or V_{DD}) terminals. Assign a node in the dual network for each region, including the area not enclosed by wires, in the non-dual graph. Finally, for each component in the non-dual network, draw a dual component which is connected to the nodes in the regions separated by the non-dual component. The dual component of an n-type transistor is a p-type, and the dual of the dummy is the dummy. You can check your work by noting that the dual of the dual of a network is the original network.

Common forms of complex logic gates are **and-or-invert** (AOI) and **or-and-invert** (OAI) gates, both of which implement sum-of-products/product-of-sums expressions. The function computed by an AOI gate is best illustrated by its logic symbol, shown in Figure 3-7: groups of inputs are ANDed together, then all products are ORed together and inverted for output. An AOI-21 gate, like that shown in the figure, has two inputs to its first product and one input (effectively eliminating the AND gate) to its second product; an AOI-121 gate would have two one-input products and one two-input product.

It is possible to construct large libraries of complex gates with different input combinations. An OAI gate computes an expression in product-of-sums form: it generates sums in the first stage which are then ANDed together and inverted. An AOI or OAI function can compute a sum-of-products or product-of-sums expression faster and using less area than an equivalent network of NAND and NOR gates. Human designers rarely make extensive use of AOI and OAI gates, however, because people have difficulty juggling a large number of gate types in their heads. Logic optimization programs, however, can make very efficient use of AOI, OAI, and other complex gates to produce very efficient layouts.

3.3.2 Basic Gate Layouts

Figure 3-8 shows a layout of an inverter, Figure 3-10 shows a layout of a static NAND gate, and Figure 3-11 shows a layout of a static NOR gate. Transistors in a gate can be densely packed—the NAND gate is not much larger than the inverter. Layouts can vary greatly, depending on the requirements of the cell: transistor sizes, positions of terminals, layers used to route signals. CMOS technology allows few major variations of the basic cell organization: V_{DD} and V_{SS} lines run in metal along the cell, with n-type transistors along the V_{SS} rail and p-types along the V_{DD} rail. The input and output signals of the NAND are presented at the cell's edge on different layers: the inputs are in poly while the output is in metal 1. If we want to cascade two cells, with the

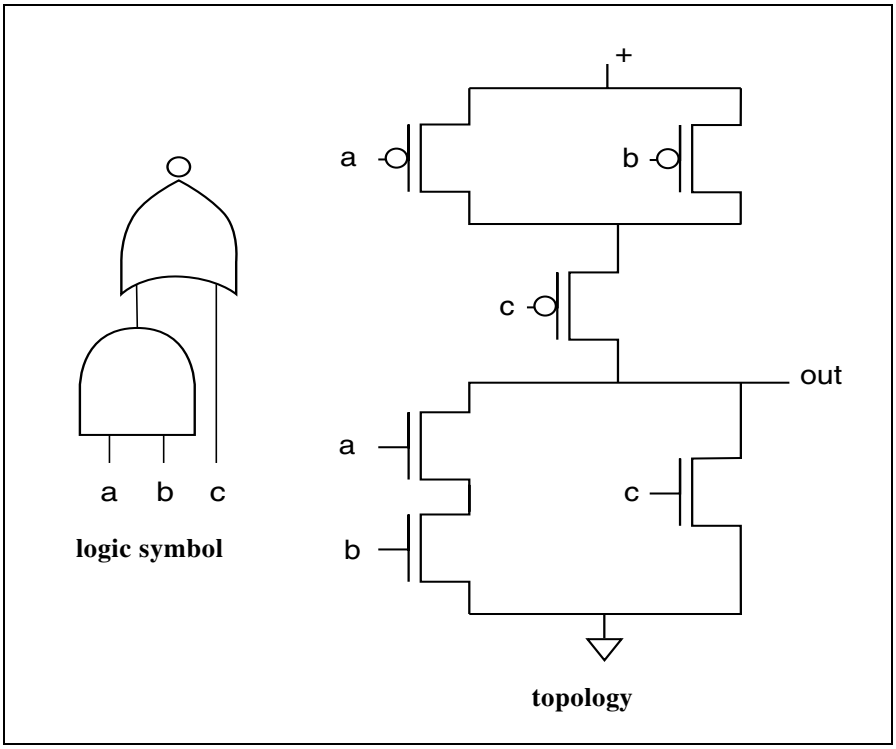


Figure 3-7: *An and-or-invert-21 (AOI-21) gate.*

output of one feeding an input of another, we will have to add a via to switch layers; we will also have to add the space between the cells required for the via and make sure that the gaps in the V_{DD} and V_{SS} caused by the gap are bridged. The p-type transistors in the NAND and NOR gate were made wide to compensate for their lower current capability; in practice, the inverter layout would probably have a wider pullup as well. We routed both input wires of the NAND to the transistor gates entirely in poly, while we used a metal 1 jumper in one of the NOR inputs.

If you are truly concerned with cell size, many variations are possible. Figure 3-9 shows a very wide transistor. A very wide transistor can create too much white space in the layout, especially if the nearby transistors are smaller. We have split this transistor into two pieces, each half as wide, and turned one piece 180 degrees, so that the outer two sections of diffusions are used as drains and the inner sections become sources.

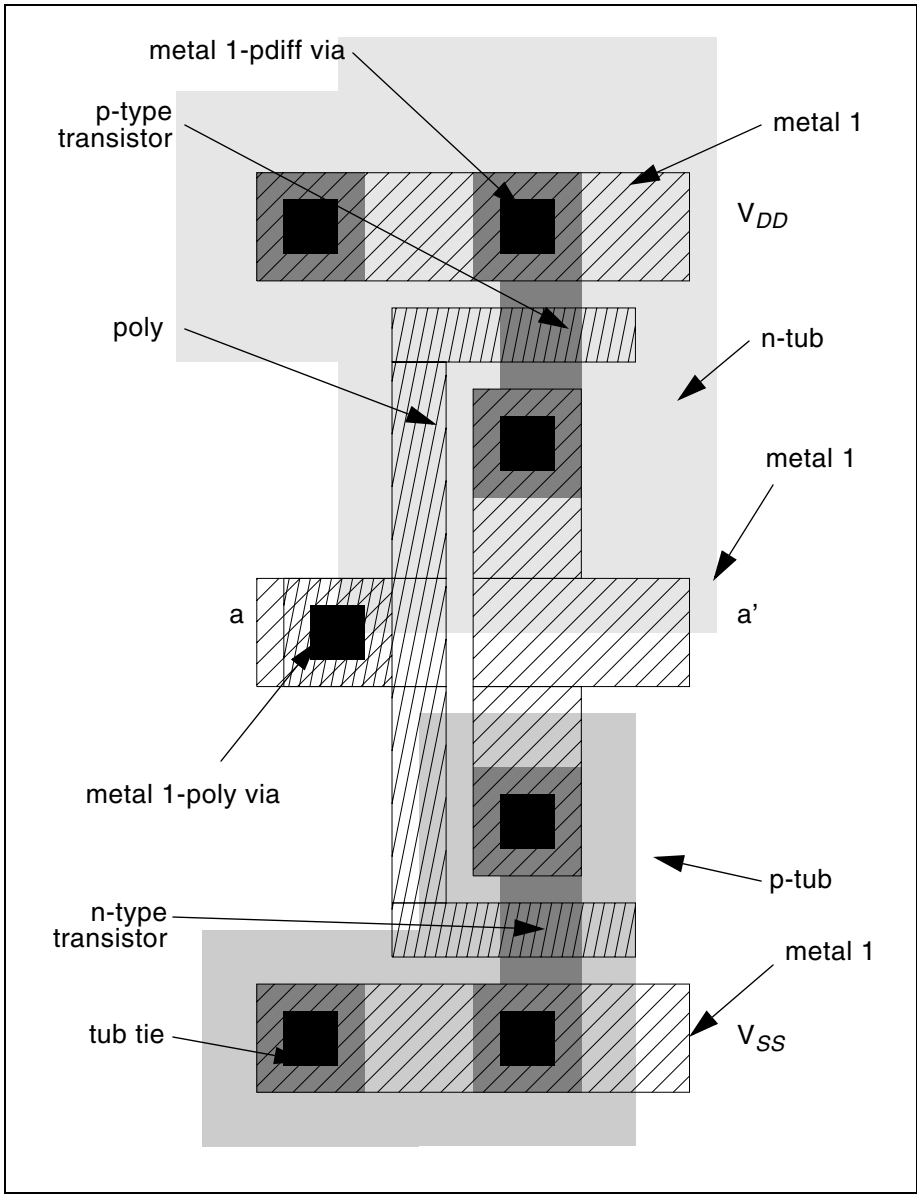


Figure 3-8: *A layout of an inverter.*

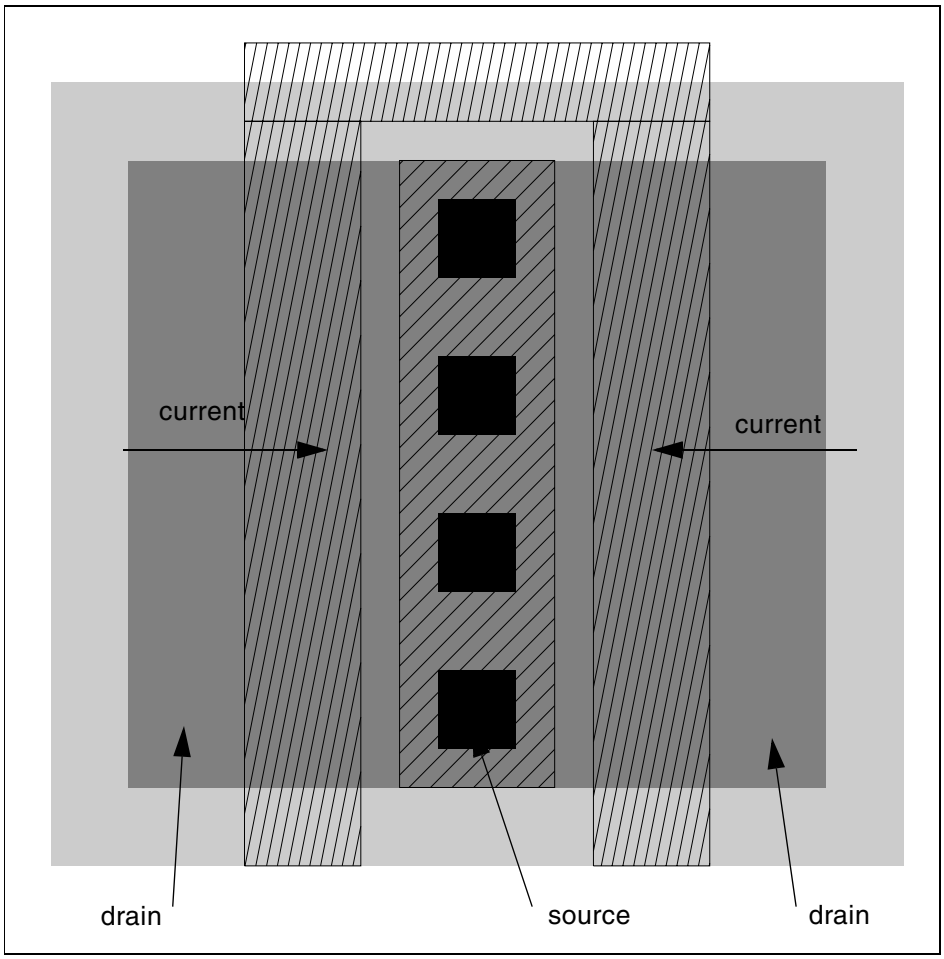


Figure 3-9: *A wide transistor split into two sections.*

3.3.3 Logic Levels

Since we must use voltages to represent logic values, we must define the relationship between the two. As Figure 3-12 shows, a range of voltages near V_{DD} corresponds to logic 1 and a band around V_{SS} corresponds to logic 0. The range in between is X, the unknown value. Although signals must swing through the X region while the chip is operating, no node should ever achieve X as its final value.

We want to calculate the upper boundary of the logic 0 region and the lower boundary of the logic 1 region. In fact, the situation is slightly more complex, as shown in

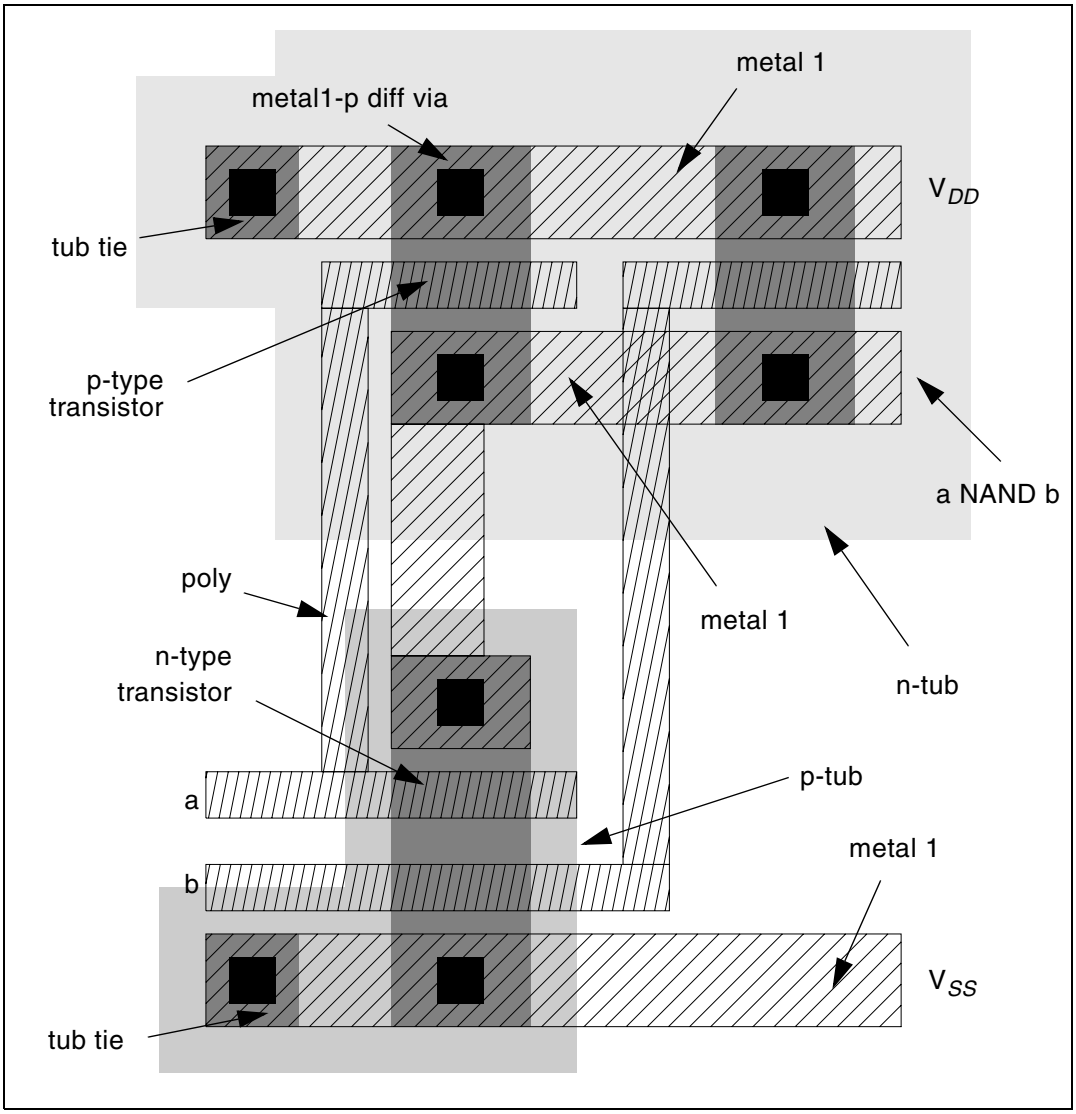


Figure 3-10: *A layout of a NAND gate.*

Figure 3-13, because we must consider the logic levels produced at outputs and required at inputs. Given our logic gate design and process parameters, we can guarantee that the maximum voltage produced for a logic 0 will be some value V_{OL} and that the minimum voltage produced for a logic 0 will be V_{OH}. These same constraints place limitations on the input voltages which will be interpreted as a logic 0 (V_{IL}) and

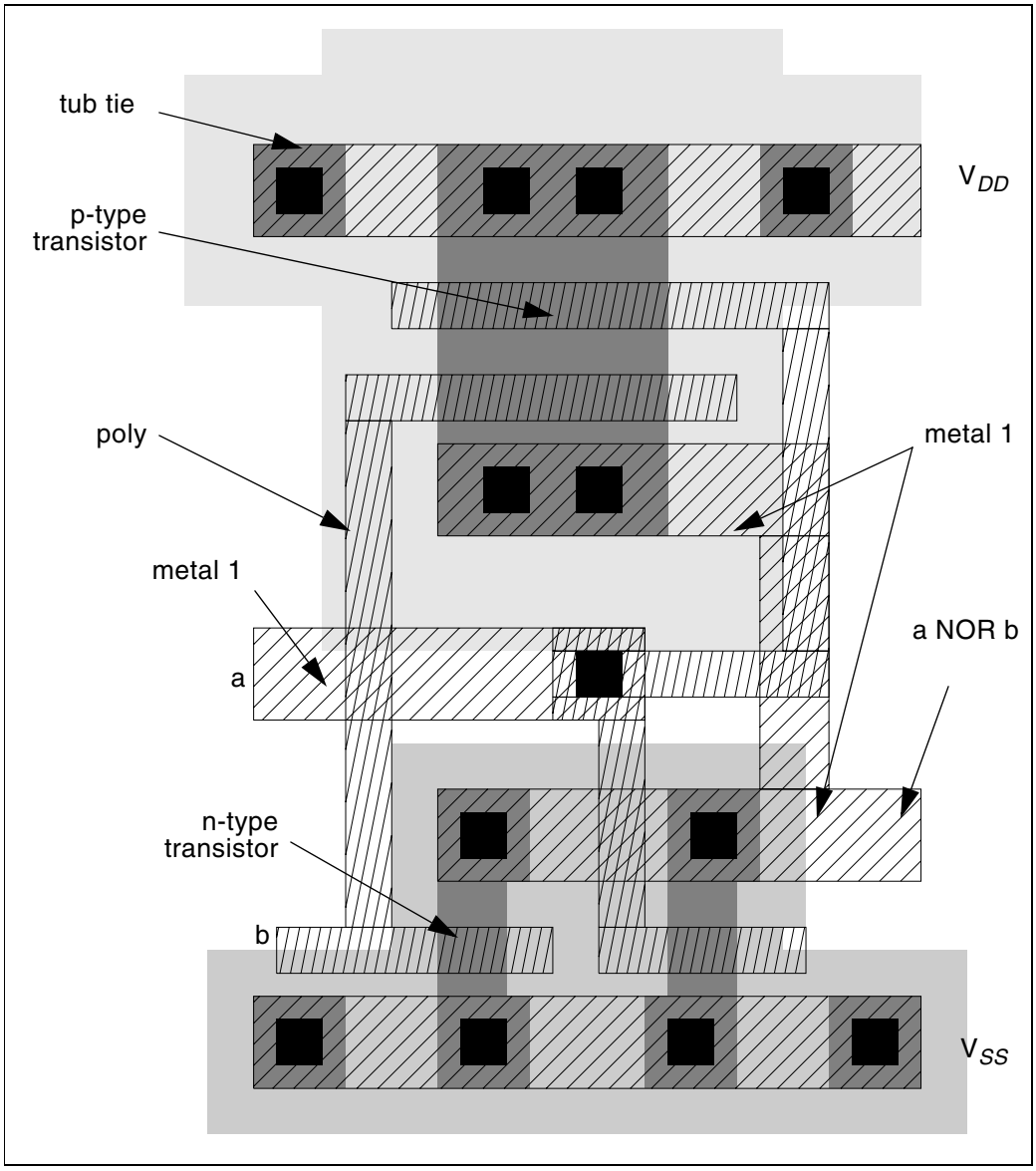


Figure 3-11: *A layout of a NOR gate.*

logic 1 (V_{IH}). If the gates are to work together, we must ensure that $V_{OL} < V_{IL}$ and $V_{OH} > V_{IH}$.

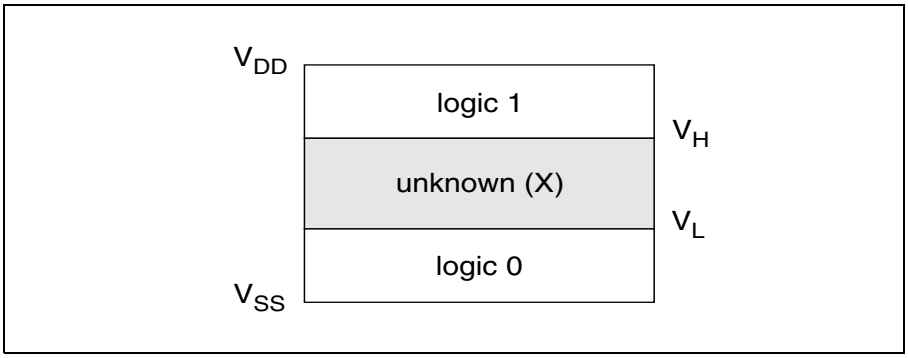


Figure 3-12: *How voltages correspond to logic levels.*

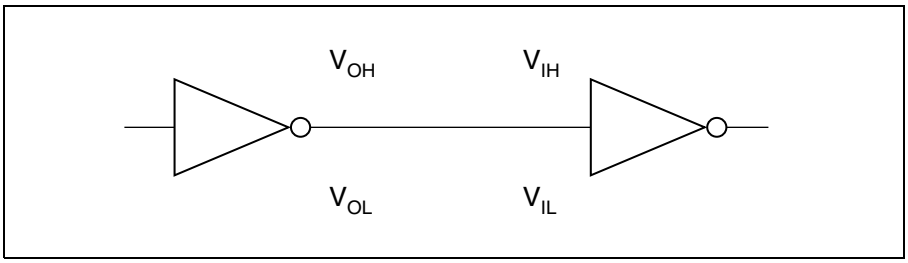


Figure 3-13: *Logic levels on cascaded gates.*

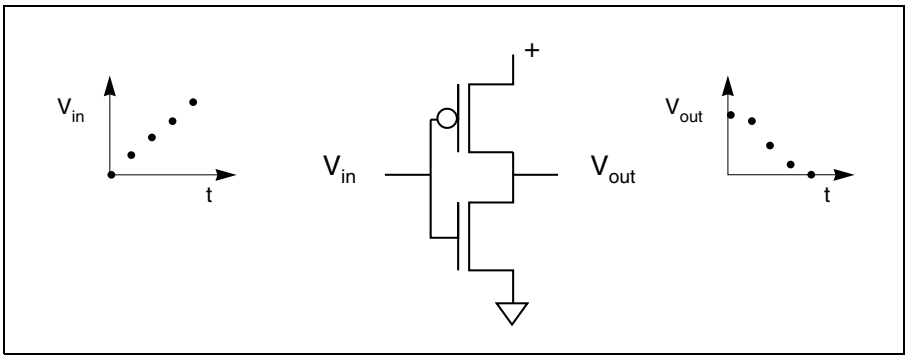


Figure 3-14: *The inverter circuit used to measure transfer characteristics.*

The output voltages produced by a static, complementary gate are V_{DD} and V_{SS}, so we know that the output voltages will be acceptable. (That isn't true of all gate circuits; the pseudo-nMOS circuit of Section 3.5.1 produces a logic 0 level well above V_{SS}.) We need to compute the values of V_{IL} and V_{IH} and to do the computation, we need to define those values. A standard definition is based on the transfer characteristic of the inverter—its output voltage as a function of its input voltage, assuming that the input voltage and all internal voltages and currents are at equilibrium. Figure 3-14 shows the circuit we will use to measure an inverter's transfer characteristic. We

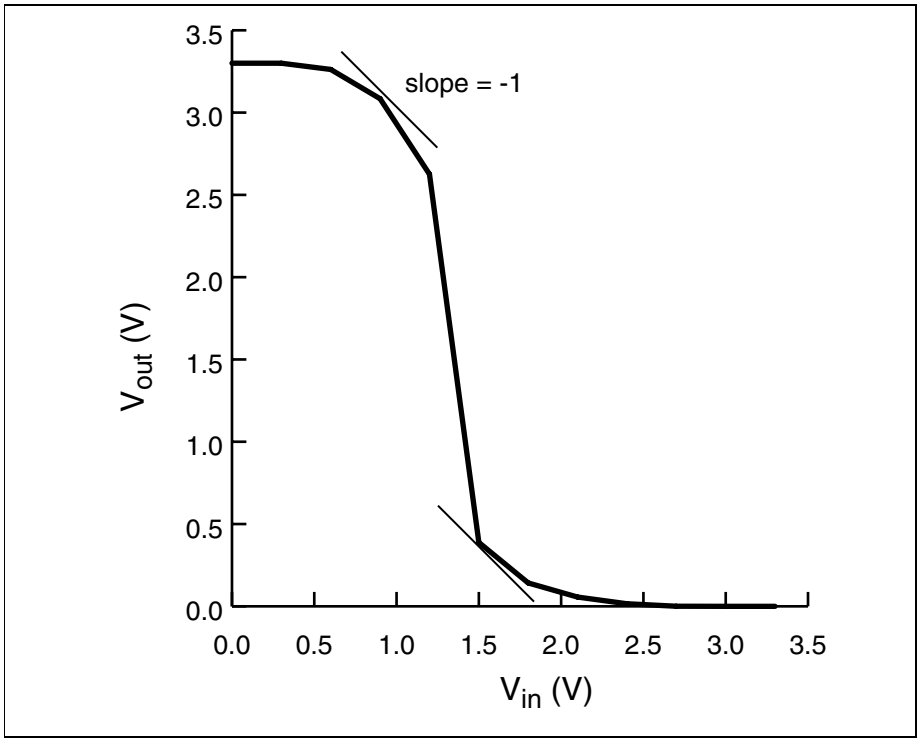


Figure 3-15: *Voltage transfer curve of an inverter with minimum-size transistors.*

apply a sequence of voltages to the input and measure the voltage at the output. (We can also sweep the input voltage if we do at a much slower rate than the circuit's transients.) Alternatively, we can solve the circuit's voltage and current equations to find V_{out} as a function of V_{in}: we equate the drain currents of the two transistors and set their gate voltages to be complements of each other (since the n-type's gate voltage is measured relative to V_{SS} and the p-type's to V_{DD}).

Figure 3-15 shows a **transfer characteristic** (simulated using Spice level 3 models) of an inverter with minimum-size transistors for both pullup and pulldown. We define V_{IL} and V_{IH} as the points at which the curve's tangent has a slope of -1. Between these two points, the inverter has high gain—a small change in the input voltage causes a large change in the output voltage. Outside that range, the inverter has a gain less than 1, so that even a large change at the input causes only a small change at the output, attenuating the noise at the gate's input. The curve is not symmetric because the pullup supplies less current than the pulldown when both are minimum size. In particular, the valid logic 1 range is smaller than the valid logic 0 range because the pullup's resistance is too high relative to the pulldown's.

The difference between V_{OL} and V_{IL} (or between V_{OH} and V_{IH}) is called the **noise margin**—the size of the safety zone that prevents production of an illegal X output value. Since real circuits function under less-than-ideal conditions, adequate noise margins are essential for ensuring that the chip operates reliably. Noise may be introduced by a number of factors: it may be introduced by off-chip connections; it may be generated by capacitive coupling to other electrical nodes; or it may come from variations in the power supply voltage.

3.3.4 Delay and Transition Time

Delay is one of the most important properties of a logic gate—the majority of chip designs are limited more by speed than by area. An analysis of logic gate delay not only tells us how to compute the speed of a gate, it also points to parasitics that must be controlled during layout design to minimize delay. Later, in Section 3.3.7, we will apply what we have learned from delay analysis to the design of logic gate layouts.

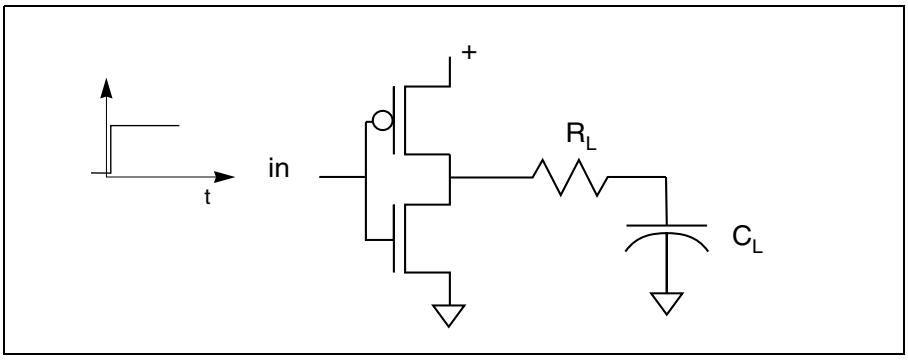


Figure 3-16: *The inverter circuit used for delay analysis.*

There are two interesting but different measures of combinational logic effort:

- **Delay** is generally used to mean the time it takes for a gate's output to arrive at 50% of its final value.
- **Transition time** is generally used to mean the time it takes for a gate to arrive at 10% (for a logic 0) or 90% (for a logic 1) of its final value; both **fall time** t_f and **rise time** t_r are transition times.

We will analyze delay and transition time on the simple inverter circuit shown in Figure 3-16; our analysis easily extends to more complex gates as well as more complex loads. We will assume that the inverter's input changes voltage instantaneously; since

the input signal to a logic gate is always supplied by another gate, that assumption is optimistic, but it simplifies analysis without completely misleading us.

It is important to recognize that we are analyzing not just the gate delay but delay of the combination of the gate and the load it drives. CMOS gates have low enough gain to be quite sensitive to their load, which makes it necessary to take the load into account in even the simplest delay analysis. The load on the inverter is a single resistor-capacitor (RC) circuit; the resistance and capacitance come from the logic gate connected to the inverter's output and the wire connecting the two. We will see in Section 4.5.1 that other models of the wire's load are possible. There are two cases to analyze: the output voltage V_{out} is pulled down (due to a logic 1 input to the inverter); and V_{out} is pulled up. Once we have analyzed the $1 \rightarrow 0$ output case, modifying the result for the $0 \rightarrow 1$ case is easy.

While the circuit of Figure 3-16 has only a few components, a detailed analysis of it is difficult due to the complexity of the transistor's behavior. We need to further simplify the circuit. A detailed circuit analysis would require us to consider the effects of both pullup and pulldown transistors. However, our assumption that the inverter's input changes instantaneously between the lowest and highest possible values lets us assume that one of the transistors turns off instantaneously. Thus, when V_{out} is pulled low, the p-type transistor is off and out of the circuit; when V_{out} is pulled high, the n-type transistor can be ignored.

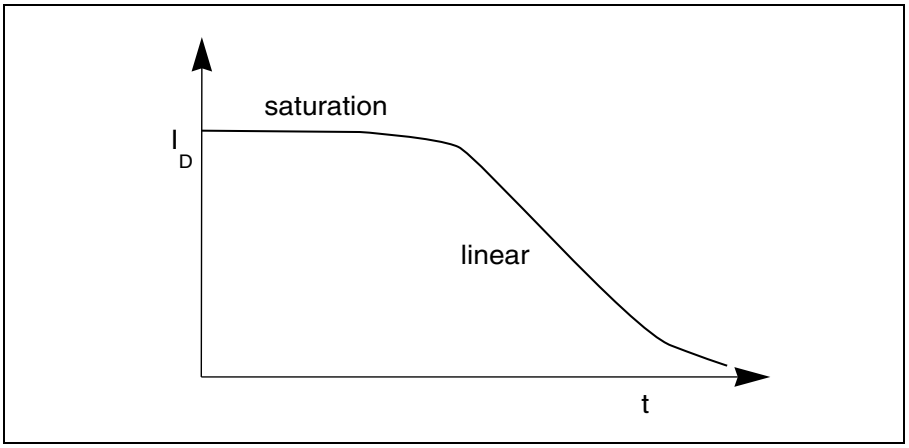


Figure 3-17: *Current through the pulldown during a $1 \rightarrow 0$ transition.*

There are several different models that people use to compute delay and transition time. The first is the τ **model**, which was introduced by Mead and Conway [Mea80] as a simple model for basic analysis of digital circuits. This model reduces the delay

of the gate to an RC time constant which is given the name τ. As the sizes of the transistors in the gate are increased, the delay scales as well.

At the heart of the τ model is the assumption that the pullup or pulldown transistor can be modeled as a resistor. The transistor does not obey Ohm's law as it drives the gate's output, of course. As Figure 3-17 shows, the pulldown spends the first part of the 1→ 0 transition in the saturation region, then moves into the linear region. But the resistive model will give sufficiently accurate results to both estimate gate delay and to understand the sources of delay in a logic circuit.

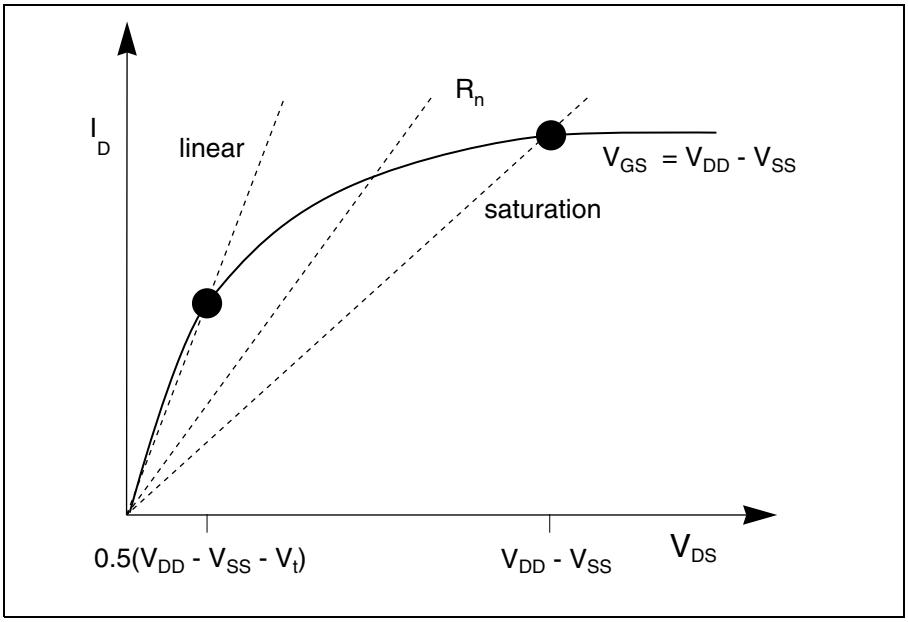


Figure 3-18: *How to approximate a transistor with a resistor.*

How do we choose a resistor value to represent the transistor over its entire operating range? A standard resistive approximation for a transistor is to measure the transistor's resistance at two points in its operation and take the average of the two values [Hod83]. We find the resistance by choosing a point along the transistor's I_d *vs.* V_{ds} curve and computing the ratio *V/I*, which is equivalent to measuring the slope of a line between that point and the origin. Figure 3-18 shows the approximation points for an n-type transistor: the inverter's maximum output voltage, $V_{DS} = V_{DD} - V_{SS}$, where the transistor is in the saturation region; and the middle of the linear region, $V_{DS} = (V_{DD}-V_{SS}-V_t)/2$. We will call the first value $R_{sat} = V_{sat}/I_{sat}$ and the second value $R_{lin} = V_{lin}/I_{lin}$. This gives the basic formula

$$R_n = \left(\frac{V_{sat}}{I_{sat}} + \frac{V_{lin}}{I_{lin}}\right) / 2 \qquad \textbf{(EQ 2-1)}$$

for which we must find the *V*s and *I*s.

The current through the transistor at the saturation-region measurement point is

$$I_{sat} = \frac{1}{2} k' \frac{W}{L} (V_{DD} - V_{SS} - V_t)^2 . \qquad \textbf{(EQ 3-2)}$$

The voltage across the transistor at that point is

$$V_{sat} = V_{DD} - V_{SS}. \qquad \textbf{(EQ 3-3)}$$

At the linear region point,

$$V_{lin} = (V_{DD} - V_{SS} - V_t)/2, \qquad \textbf{(EQ 3-4)}$$

so the drain current is

$$\begin{aligned} I_{lin} &= k' \frac{W}{L} \left[\frac{1}{2} (V_{DD} - V_{SS} - V_t)^2 - \frac{1}{2} \left(\frac{V_{DD} - V_{SS} - V_t}{2} \right)^2 \right] \\ &= \frac{3}{8} k' \frac{W}{L} (V_{DD} - V_{SS} - V_t)^2 \end{aligned} \qquad \textbf{(EQ 3-5)}$$

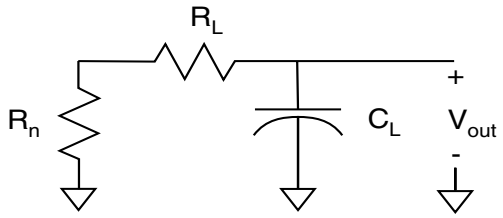
We can compute the effective resistances of transistors in the 0.5 µ*m* process by plugging in the technology values of Table 2-4. The resistance values for minimum-size n-type and p-type transistors are shown in Table 3-1 for two power supply voltages: 5V and 3.3V. The effective resistance of a transistor is scaled by *L/W*. The p-type transistor has about three-and-a-half times the effective resistance of an n-type transistor for this set of process parameters, which is what we expect from the ratio k'_n / k'_p . and the variation in threshold voltages between the two types of transistors. Note that the effective resistance of the transistors increases as the power supply voltage goes down.

Table 3-1 *Effective resistance values for minimum-size transistors in our 0.5 µm process.*

type	V_{DD}-V_{SS} = 5V	V_{DD}-V_{SS} = 3.3V
R_n	**3.9 kΩ**	**6.8 kΩ**
R_p	**14 kΩ**	**25 kΩ**

Given these resistance values, we can then analyze the delay and transition time of the gate.

Figure 3-19: *The circuit model for τ model delay.*



We can now develop the τ model that helps us compute delay and transition time. Figure 3-19 shows the circuit model we use: R_n is the transistor's effective resistance while R_L and C_L are the load. The capacitor has an initial voltage of V_{DD}. The transistor discharges the load capacitor from V_{DD} to V_{SS}; the output voltage as a function of time is

$$V_{\text{out}}(t) = V_{\text{DD}} e^{-t/[(R_n + R_L)C_L]}. \qquad \textbf{(EQ 3-6)}$$

We typically use R_L to represent the resistance of the wire which connects the inverter to the next gate; in this case, we'll assume that $R_L = 0$, simplifying the total resistance to $R = R_n$.

To measure delay, we must calculate the time required to reach the 50% point. Then

$$0.5 = e^{-t_d/[(R_n + R_L)C_L]}, \quad \textbf{(EQ 3-7)}$$

$$t_d = -(R_n + R_L)C_L \ln 0.5 = 0.69(R_n + R_L)C_L. \quad \textbf{(EQ 3-8)}$$

We generally measure transition time as the interval between the time at which $V_{out} = 0.9V_{DD}$ and $V_{out} = 0.1V_{DD}$; let's call these times t_1 and t_2. Then

$$t_f = t_2 - t_1 = -(R_n + R_L)C_L ln\frac{0.1}{0.9} = 2.2(R_n + R_L)C_L. \quad \textbf{(EQ 3-9)}$$

The next example illustrates how to compute delay and transition time using the τ model.

Example 3-1: *Inverter delay and transition time using the τ model*

Once the effective resistance of a transistor is known, delay calculation is easy. What is a minimum inverter delay and fall time with our 0.5 μ*m* process parameters? Assume a minimum-size pulldown, no wire resistance, and a capacitive load equal to two minimum-size transistors' gate capacitance. First, the τ model parameters:

$$R_n = 3.9k\Omega$$

$$C_L = 0.9\frac{fF}{\mu m^2} \times \left(3\lambda \times 2\lambda \times \frac{0.0625\mu m^2}{\lambda^2}\right) \times 2$$

$$= 0.68fF$$

Then delay is

$$t_d = 0.69 \cdot 3.9k\Omega \cdot 0.68 \times 10^{-15} = 1.8ps$$

and fall time is

$$t_f = 2.2 \cdot 3.9k\Omega \cdot 0.68\times10^{-15} = 5.8ps$$

If the transistors are not minimum size, their effective resistance is scaled by *L*/*W*. To compute the delay through a more complex gate, such as a NAND or an AOI, compute the effective resistance of the pullup/pulldown network using the standard Ohm's law simplifications, then plug the effective *R* into the delay formula.

If we decrease the supply voltage to 3.3 V, the load capacitance does not change but the effective resistance of the transistor does:

$$R_n = 6.8k\Omega,$$

$$t_d = 0.69 \cdot 6.8k\Omega \cdot 0.68\times10^{-15} = 3.1ps,$$

$$t_f = 2.2 \cdot 6.8k\Omega \cdot 0.68\times10^{-15} = 10ps$$

This simple RC analysis tells us two important facts about gate delay. First, if the pullup and pulldown transistor sizes are equal, the 0→ 1 transition will be about one-half to one-third the speed of the 1→ 0 transition. That observation follows directly from the ratio of the n-type and p-type effective resistances. Put another way, to make the high-going and low-going transition times equal, the pullup transistor must be twice to three times as wide as the pulldown. Second, complex gates like NANDs and NORs require wider transistors where those transistors are connected in series. A NAND's pulldowns are in series, giving an effective pulldown resistance of $2R_n$. To give the same delay as an inverter, the NAND's pulldowns must be twice as wide as the inverter's pulldown. The NOR gate has two p-type transistors in series for the pullup network. Since a p-type transistor must be two to three times wider than an n-type transistor to provide equivalent resistance, the pullup network of a NOR can take up quite a bit of area.

A second model is the **current source model**, which is sometimes used in power/delay studies because of its tractability. If we assume that the transistor acts as a current source whose V_{gs} is always at the maximum value, then the delay can be approximated as

$$t_f = \frac{C_L(V_{DD}\text{-}V_{SS})}{I_d} = \frac{C_L(V_{DD}\text{-}V_{SS})}{0.5k'(W/L)(V_{DD}\text{-}V_{SS}\text{-}V_t)^2}. \qquad \textbf{(EQ 3-10)}$$

A third type of model is the **fitted model**. This approach measures circuit characteristics and fits the observed characteristics to the parameters in a delay formula. This technique is not well-suited to hand analysis but it is easily used by programs that analyze large numbers of gates.

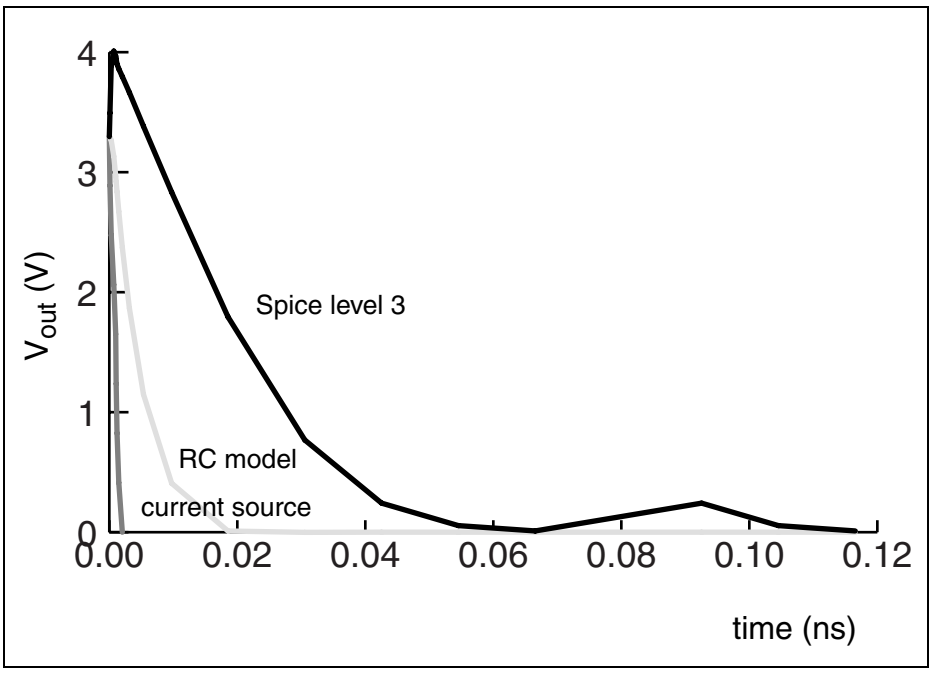


Figure 3-20: *Comparison of inverter delay to the RC and current source approximations.*

How accurate are the RC and current source approximations? Figure 3-20 shows the results of Spice simulation of three circuits: a resistance discharging a capacitance equal to the gate capacitance of an inverter with minimum-size transistors, a full inverter discharging the same capacitance, and the current source approximation. The

results show that, for these process parameters, the resistance value calculated by the two-step method is somewhat optimistic and the current source approximation is even more so. We can use Spice simulation to generate a more accurate model of an inverter for a particular process, but you should always remember that the RC delay model is meant as only a rough approximation. RC and current source delay are best

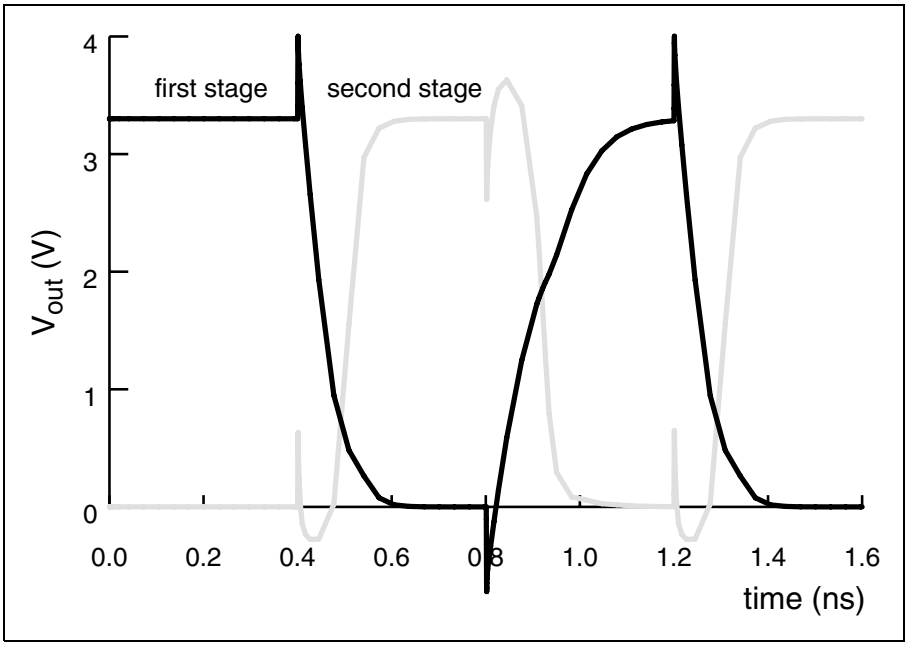


Figure 3-21: *Circuit simulation of a pair of inverters.*

used as relative measures of delay, not absolute measures.

The RC model assumes that the gate's input is a step, but the input in fact comes from another gate which may generate a relatively slow signal. Figure 3-21 shows the results of Spice simulation of one inverter driving another; the first is driven by a square wave, but the second is driven by the output of the first inverter. The second inverter's output response is somewhat slower than the first's.

The fundamental reason for developing an RC model of delay is that we often can't afford to use anything more complex. Full circuit simulation of even a modest-size chip is infeasible: we can't afford to simulate even one waveform, and even if we could, we would have to simulate all possible inputs to be sure we found the worst-case delay. The RC model lets us identify sections of the circuit which probably limit circuit performance; we can then, if necessary, use more accurate tools to more closely analyze the delay problems of that section.

Body effect, as we saw in Section 2.3.5, is the modulation of threshold voltage by a difference between the voltage of the transistor's source and the substrate—as the source's voltage rises, the threshold voltage also rises. This effect can be modeled by a capacitor from the source to the substrate's ground as shown in Figure 3-22. To eliminate body effect, we want to drive that capacitor to 0 voltage as soon as possible. If there is one transistor between the gate's output and the power supply, body effect is not a problem, but series transistors in a gate pose a challenge. Not all of the gate's input signals may reach their values at the same time—some signals may arrive earlier than others. If we connect early-arriving signals to the transistors nearest the power supply and late-arriving signals to transistors nearest the gate output, the early-arriving signals will discharge the body effect capacitance of the signals closer to the output. This simple optimization can have a significant effect on gate delay [Hil89].

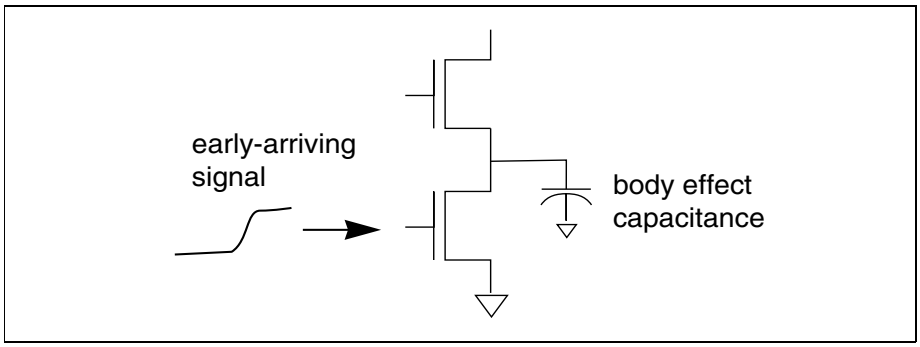


Figure 3-22: *Body effect and signal ordering.*

3.3.5 Power Consumption

Analyzing the power consumption of an inverter provides an alternate window into the cost and performance of a logic gate. Circuits can be made to go faster—up to a point—by causing them to burn more power. Power consumption always comes at the cost of heat which must be dissipated out of the chip. Static, complementary CMOS gates are remarkably efficient in their use of power to perform computation.

Once again we will analyze an inverter with a capacitor connected to its output. However, to analyze power consumption we must consider both the pullup and pulldown phases of operation. The model circuit is shown in Figure 3-23. The first thing to note about the circuit is that it has almost no steady-state power consumption. After the output capacitance has been fully charged or discharged, only one of the pullup and pulldown transistors is on. The following analysis ignores the leakage current; we will look at techniques to combat leakage current in Section 3.6.

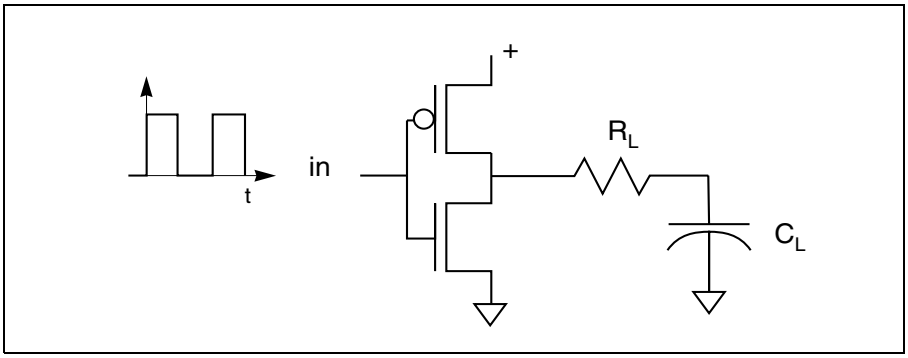


Figure 3-23: *Circuit used for power consumption analysis.*

Power is consumed when gates drive their outputs to new values. Surprisingly, the power consumed by the inverter is independent of the sizes/resistances of its pullup and pulldown transistors—power consumption depends only on the size of the capacitive load at the output and the rate at which the inverter's output switches. To understand why, consider the energy required to drive the inverter's output high calculated two ways: by the current through the load capacitor C_L and by the current through the pullup transistor, represented by its effective resistance R_p.

The current through the capacitor and the voltage across it are:

$$i_{CL}(t) = \frac{V_{DD}\text{-}V_{SS}}{R_p} e^{-(t/R_pC_L)} \quad , \qquad \textbf{(EQ 3-11)}$$

$$v_{CL}(t) = (V_{DD}\text{-}V_{SS})[1\text{-}e^{-(t/R_pC_L)}] \quad . \qquad \textbf{(EQ 3-12)}$$

So, the energy required to charge the capacitor is:

$$E_C = \int_0^\infty i_{C_L}(t)v_{C_L}(t))dt \qquad \textbf{(EQ 3-13)}$$

$$= C_L(V_{DD}\text{-}V_{SS})^2\left(e^{-t/R_pC_L}\text{-}\frac{1}{2}e^{-2t/R_PC_L}\right)\Bigg|_0^\infty$$

$$= \frac{1}{2}C_L(V_{DD}\text{-}V_{SS})^2$$

This formula depends on the size of the load capacitance but not the resistance of the pullup transistor. The current through and voltage across the pullup are:

$$i_p(t) = i_{CL}(t), \quad \textbf{(EQ 3-14)}$$

$$v_p(t) = Ve^{-(t/R_pC_L)}. \quad \textbf{(EQ 3-15)}$$

The energy required to charge the capacitor, as computed from the resistor's point of view, is

$$\begin{aligned} E_R &= \int_0^\infty i_p(t)v_p(t)dt \\ &= C_L(V_{DD}\text{-}V_{SS})^2(e^{-2t/R_pC_L})\Big|_0^\infty \\ &= \frac{1}{2}C_L(V_{DD}\text{-}V_{SS})^2 \end{aligned} \quad \textbf{(EQ 3-16)}$$

Once again, even though the circuit's energy consumption is computed through the pullup, the value of the pullup resistance drops from the energy formula. (That holds true even if the pullup is a nonlinear resistor.) The two energies have the same value because the currents through the resistor and capacitor are equal.

The energy consumed in discharging the capacitor can be calculated the same way. The discharging energy consumption is equal to the charging power consumption: $1/2\,C_L(V_{DD}\text{-}V_{SS})^2$. A single cycle requires the capacitor to both charge and discharge, so the total energy consumption is $C_L(V_{DD}\text{-}V_{SS})^2$.

Power is energy per unit time, so the power consumed by the circuit depends on how frequently the inverter's output changes. The worst case is that the inverter alternately charges and discharges its output capacitance. This sequence takes two clock cycles. The clock frequency is $f = 1/t$. The total power consumption is

$$fC_L(V_{DD}\text{-}V_{SS})^2. \quad \textbf{(EQ 3-17)}$$

Power consumption in CMOS circuits depends on the frequency at which they operate, which is very different from nMOS or bipolar logic circuits. Power consumption depends on clock frequency because most power is consumed while the outputs are

changing; most other circuit technologies burn most of their power while the circuit is idle. Power consumption depends on the sizes of the transistors in the circuit only in that the transistors largely determine C_L. The current through the transistors, which is determined by the transistor *W/Ls*, doesn't determine power consumption, though the available transistor current does determine the maximum speed at which the circuit can run, which indirectly determines power consumption.

Does it make sense that CMOS power consumption should be independent of the effective resistances of the transistors? It does, when you remember that CMOS circuits consume only dynamic power. Most power calculations are made on static circuits—the capacitors in the circuit have been fully charged or discharged, and power consumption is determined by the current flowing through resistive paths between V_{DD} and V_{SS} in steady state. Dynamic power calculations, like those for our CMOS circuit, depend on the current flowing through capacitors; the resistors determine only maximum operating speed, not power consumption.

Static complementary gates can operate over a wide range of voltages, allowing us to trade delay for power consumption. To see how performance and power consumption are related, let's consider changing the power supply voltage from its original value V to a new V'. It follows directly from Equation 3-17 that the ratio of power consumptions P'/P is proportional to V'^2/V^2. When we compute the ratio of rise times t'_r/t_r the only factor to change with voltage is the transistor's equivalent resistance R, so the change in delay depends only on R'/R. If we use the technique of Section 3.3.4 to compute the new effective resistance, we find that $t'_r/t_r \propto V/V'$. So as we reduce power supply voltage, power consumption goes down faster than does delay.

3.3.6 The Speed-Power Product

The **speed-power product**, also known as the **power-delay product**, is an important measure of the quality of a logic circuit family. Since delay can in general be reduced by increasing power consumption, looking at either power or delay in isolation gives an incomplete picture.

The speed-power product for static CMOS is easy to calculate. If we ignore leakage current and consider the speed and power for a single inverter transition, then we find that the speed-power product SP is

$$SP = \frac{1}{f}P = CV^2 . \qquad \textbf{(EQ 3-18)}$$

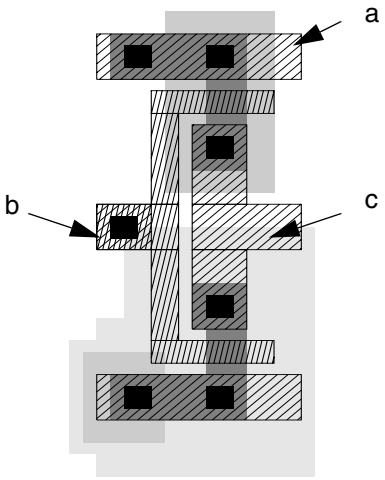
The speed-power product for static CMOS is independent of the operating frequency of the circuit. It is, however, a quadratic function of the power supply voltage. This result suggests an important method for power consumption reduction known as **voltage scaling**: we can often reduce power consumption by reducing the power supply voltage and adding parallel logic gates to make up for the lower performance. Since the power consumption shrinks more quickly than the circuit delay when the voltage is scaled, voltage scaling is a powerful technique. We will study techniques for low-power gate design in Section 3.6.

3.3.7 Layout and Parasitics

How do parasitics affect the performance of a single gate? Answering this question tells us how to design the layout of a gate to maximize performance and minimize area.

Example 3-2: *Parasitics and performance*

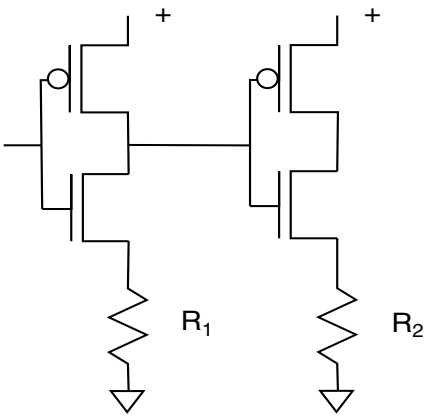
To answer the question, we will consider the effects of adding resistance and capacitance to each of the labeled points of this layout:



- *a*

Adding capacitance to point a (or its conjugate point on the V_{SS} wire) adds capacitance to the power supply wiring. Capacitance on this node doesn't slow down the gate's output.

Resistance at *a* can cause problems. Resistance in the V_{SS} line can be modeled by this equivalent circuit:



The power supply resistance is in series with the pulldown. That differential isn't a serious problem in static, complementary gates. The resistance slows down the gate, but since both the transistor gates of the pullup and pulldown are connected to the same electrical node, we can be sure that only one of them will be on in steady state. However, the dynamic logic circuits we will discuss in Section 3.5 may not work if the series power supply resistance is too high, because the voltages supplied by the gate with resistance may not properly turn on succeeding transistor gates.

The layout around point a should be designed to minimize resistance. A small length of diffusion is required to connect the transistors to the power lines, but power lines should be kept in metal as long as possible. If the diffusion wire is wider than a via (to connect to a wide transistor), several parallel vias should be used to connect the metal and diffusion lines.

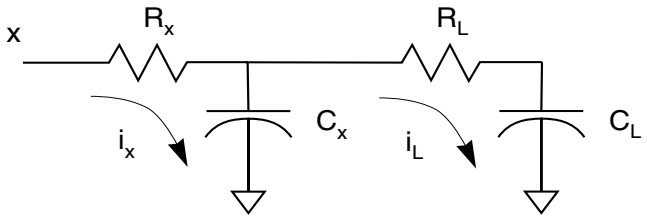
- b

  Capacitance at b adds to the load of the gate driving this node. However, the transistor gate capacitances are much larger than the capacitance added by the short wire feeding the transistor gates. Resistance at b actually helps isolate the previous gate from the load capacitance, as we will see when we discuss the π model in Section 4.5.1 Gate layouts should avoid making big mistakes by using large sections of diffusion wire or a single via to connect high-current wires.

- c

  Capacitance and resistance at c are companions to parasitics at b—they form part of the load that this gate must drive, along with the parasitics of the b zone of the next gate. But if we consider a more accurate model of the parasitics, we will see that not all positions for parasitic *R* and *C* are equally bad.

Up to now we have modeled the resistance and capacitance of a wire as single components. But now consider the inverter's load as two RC sections:



One *RC* section is contributed by the wires at point c, near the output; the *RC* section comes from the long wire connecting this gate to the next one. How does the voltage at point x—the input to the next gate—depend on the relative values of the *R*'s? The simplified circuit shows how a large value for R_x, which is supplied by the parasitics at point c, steals current from $R_L C_L$. As R_x grows relative to R_L, the voltage drop across $R_x$increases, increasing the current through R_x while decreasing the current through R_L. As a result, more of the current supplied by the gate will go through C_x; only after it is fully charged will C_L get the full current supplied by the gate. Since C_L is almost certainly significantly larger than C_x, since it includes both the transistor gate capacitances and the long-wire capacitance, it is more important to charge C_L to switch the next gate as quickly as possible. But charging/discharging of C_L has been delayed while R_x diverts current into C_x.

The moral is that resistance close to the gate output is worse than resistance farther away—close-in resistance must charge more capacitors, slowing down the signal swing at the far end of the wire. Therefore, the layout around c should be designed to minimize resistance. That requires:

- using as little diffusion as possible—diffusion should be connected to metal (or perhaps poly) as close to the channel as possible;

- using parallel vias at the diffusion/metal interface to minimize resistance.

3.3.8 Driving Large Loads

Logic delay increases as the capacitance attached to the logic's output becomes larger. In many cases, one small logic gate is driving an equally small logic gate, roughly matching drive capability to load. However, there are several situations in which the capacitive load can be much larger than that presented by a typical gate:

- driving a signal connected off-chip;
- driving a long signal wire;
- driving a clock wire which goes to many points on the chip.

The obvious answer to driving large capacitive loads is to increase current by making wider transistors. However, this solution begs the question—those large transistors simply present a large capacitive load to the gate which drives them, pushing the problem back one level of logic. It is inevitable that we must eventually use large transistors to drive the load, but we can minimize delay along the path by using a sequence of successively larger drivers.

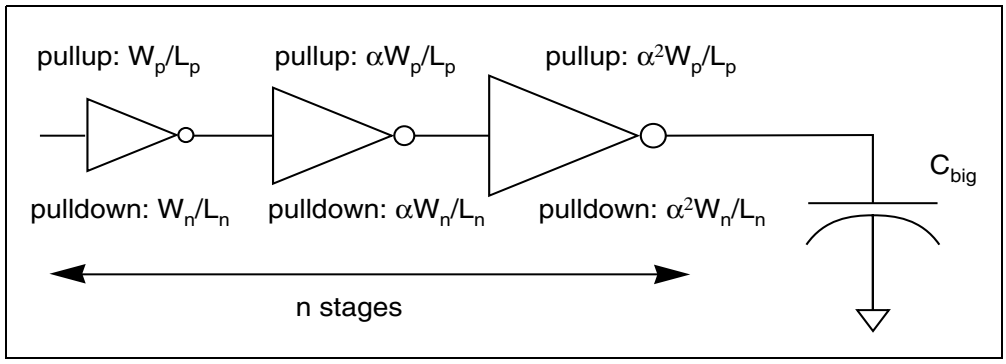


Figure 3-24: *Cascaded inverters driving a large capacitive load.*

The driver chain with the smallest delay to drive a given load is exponentially tapered—each stage supplies e times more current than the last [Jae75]. In the chain of inverters of Figure 3-24, each inverter can produce α times more current than the previous stage (implying that its pullup and pulldown are each α times larger). If C_g is the minimum-size load capacitance, the number of stages n is related to α by the formula $\alpha = (C_{big}/C_g)^{1/n}$. The time to drive a minimum-size load is t_{min}. We want to minimize the total delay through the driver chain:

$$t_{tot} = n\left(\frac{C_{big}}{C_g}\right)^{1/n} t_{min} \ . \qquad \textbf{(EQ 3-19)}$$

To find the minimum, we set $\frac{dt_{tot}}{dn} = 0$, which gives

$$n_{opt} = ln\left(\frac{C_{big}}{C_g}\right) . \qquad \textbf{(EQ 3-20)}$$

When we substitute the optimal number of stages back into the definition of α, we find that the optimum value is at $\alpha = e$. Of course, n must be an integer, so we will not in practice be able to implement the exact optimal circuit. However, delay changes slowly with n near the optimal value, so rounding n to the floor of n_{opt} gives reasonable results.

3.4 Switch Logic

How do we build switches from MOS transistors? One way is the **transmission gate** shown in Figure 3-25, built from parallel n-type and p-type transistors. This switch is built from both types of transistors so that it transmits logic 0 and 1 from drain to source equally well: when you put a V_{DD} or V_{SS} at the drain, you get V_{DD} or V_{SS} at the source. But it requires two transistors and their associated tubs; equally damning, it requires both true and complement forms of the gate signal.

An alternative is the **n-type switch**—a solitary n-type transistor. It requires only one transistor and one gate signal, but it is not as forgiving electrically: it transmits a logic 0 well, but when V_{DD} is applied to the drain, the voltage at the source is V_{DD} - V_{tn}. When switch logic drives gate logic, n-type switches can cause electrical problems. An n-type switch driving a complementary gate causes the complementary gate to run

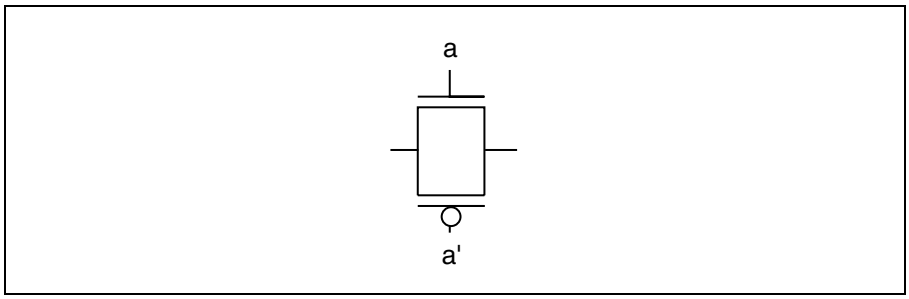


Figure 3-25: *A complementary transmission gate.*

slower when the switch input is 1: since the n-type pulldown current is weaker when a lower gate voltage is applied, the complementary gate's pulldown will not suck current off the output capacitance as fast. When the n-type switch drives a pseudo-nMOS gate, disaster may occur. A pseudo-nMOS gate's ratioed transistors depend on logic 0 and 1 inputs to occur within a prescribed voltage range. If the n-type switch doesn't turn on the pseudo-nMOS pulldown strongly enough, the pulldown may not divert enough current from the pullup to force the output to a logic 0, even if we wait forever. Ratioed logic driven by n-type switches must be designed to produce valid outputs for both polarities of input.

Both types of switch logic are sensitive to noise—pulling the source beyond the power supply (above V_{DD} or below V_{SS}) causes the transistor to start conducting. We will see in Section 4.7 that logic networks made of switch logic are prone to errors introduced by parasitic capacitance.

3.5 Alternative Gate Circuits

The static complementary gate has several advantages: it is reliable, easy to use in large combinational logic networks, and does not require any separate precharging steps. It is not, however, the only way to design a logic gate with p-type and n-type transistors. Other circuit topologies have been created that are smaller or faster (or both) than static complementary gates. Still others use less power.

In this section we will review the design of several important alternative CMOS gate topologies. Each has important uses in chip design. But it is important to remember that they all have their limitations and caveats. Specialized logic gate designs often require more attention to the details of circuit design—while the details of circuit and layout design affect only the speed at which a static CMOS gate runs, circuit and layout problems can cause a fancier gate design to fail to function correctly. Particular

care must be taken when mixing logic gates designed with different circuit topologies to ensure that one's output meets the requirements of the next's inputs. A good, conservative chip design strategy is to start out using only static complementary gates, then to use specialized gate designs in critical sections of the chip to meet the project's speed or area requirements.

3.5.1 Pseudo-nMOS Logic

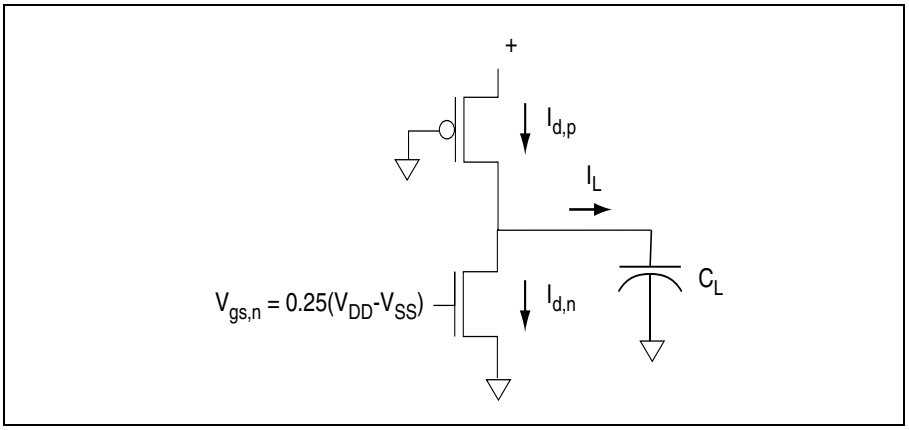


Figure 3-26: *A pseudo-nMOS NOR gate.*

The simplest non-standard gate topology is **pseudo-nMOS**, so called because it mimics the design of an nMOS logic gate. Figure 3-26 shows a pseudo-nMOS NOR gate. The pulldown network of the gate is the same as for a fully complementary gate. The pullup network is replaced by a single p-type transistor whose gate is connected to V_{SS}, leaving the transistor permanently on. The p-type transistor is used as a resistor: when the gate's inputs are ab = 00, both n-type transistors are off and the p-type transistor pulls the gate's output up to V_{DD}. When either a or b is 1, both the p-type and n-type transistor are on and both are fighting to determine the gate's output voltage.

We need to determine the relationship between the *W/L* ratios of the pullup and the pulldowns which provide reasonable output voltages for the gate. For simplicity, assume that only one of the pulldown transistors is on; then the gate circuit's output voltage depends on the ratio of the effective resistances of the pullup and the operating pulldown. The high output voltage of the gate is V_{DD}, but the output low voltage V_{OL} will be some voltage above V_{SS}. The chosen V_{OL} must be low enough to activate the next logic gate in the chain. For pseudo-nMOS gates which feed static or pseudo-nMOS gates, a value of $V_{OL} = 0.15(V_{DD}\text{-}V_{SS})$ is a reasonable value, though others could be chosen. To find the transistor sizes which give reasonable output voltages, we must consider the simultaneous operation of the pullup and pull-

down. When the gate's output has just switched to a logic 0, the n-type pulldown is in saturation with $V_{gs,n} = V_{in}$. The p-type pullup is in its linear region: its $V_{gs,p} = V_{DD} - V_{SS}$ and its $V_{ds,p} = V_{out} - (V_{DD} - V_{SS})$. We need to find V_{out} in terms of the *W/L*s of the pullup and pulldown. To solve this problem, we set the currents through the saturated pulldown and the linear pullup to be equal:

$$\frac{1}{2}k'_n\frac{W_n}{L_n}(V_{gs,n}-V_{tn})^2 = \frac{1}{2}k'_p[2(V_{gs,p}-V_{tp})V_{ds,p}-V_{ds,p}^2] \quad . \quad \textbf{(EQ 3-21)}$$

The simplest way to solve this equation is to substitute the technology and circuit values. Using the 0.5 µm values and assuming a 3.3V power supply and a full-swing input ($V_{gs,n} = V_{DD}-V_{SS}$), we find that

$$\frac{W_p/L_p}{W_n/L_n} \approx 3.9 \; . \quad \textbf{(EQ 3-22)}$$

The pulldown network must exhibit this effective resistance in the worst case combination of inputs. Therefore, if the network contains series pulldowns, they must be made larger to provide the required effective resistance.

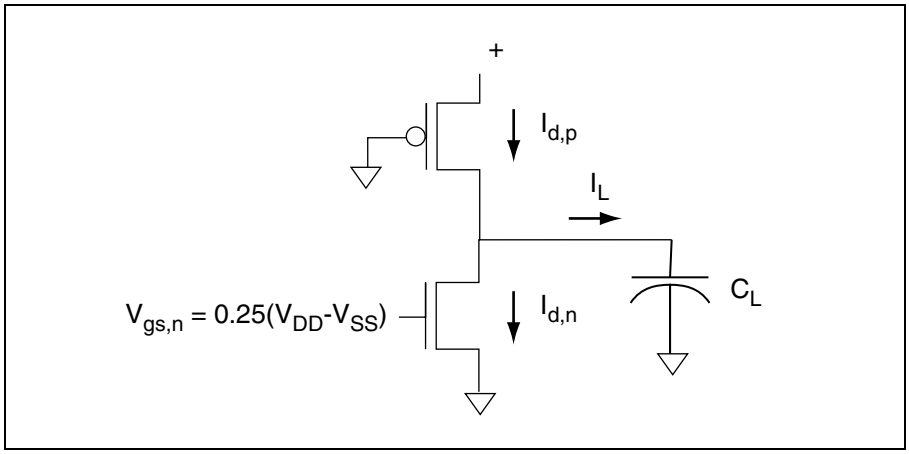


Figure 3-27: *Currents in a pseudo-nMOS gate during low-to-high transition.*

The pseudo-nMOS gate consumes static power, unlike the fully complementary gate. When both the pullup and pulldown are on, the gate forms a conducting path from V_{DD} to V_{SS}, which must be kept on to maintain the gate's logic output value. The choice of V_{OL} determines whether the gate consumes may consume static power

when its output is logic 1. If pseudo-nMOS feeds pseudo-nMOS and V_{OL} is chosen to be greater than $V_{t,n}$, then the pulldown will remain on. Whether the pulldown is in the linear or saturation region depends on the exact transistor characteristics, but in either case, its drain current will be low since $V_{gs,n}$ is low. As shown in Figure 3-27, so long as the pulldown drain current is significantly less than the pullup drain current, there will be enough current to charge the output capacitance and bring the gate output to the desired level.

The ratio of the pullup and pulldown sizes also ensures that the times for $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions are asymmetric. Since the pullup transistor has about three times the effective resistance of the pulldown, the $0 \rightarrow 1$ transition occurs much more slowly than the $1 \rightarrow 0$ transition and dominates the gate's delay. The long pullup time makes the pseudo-nMOS gate slower than the static complementary gate.

Why use a pseudo-nMOS gate? The main advantage of the pseudo-nMOS gate is the small size of the pullup network, both in terms of number of devices and wiring complexity. The pullup network of a static complementary gate can be large for a complex function. Furthermore, the input signals do not have to be routed to the pullup, as in a static complementary gate. The pseudo-nMOS gate is used for circuits where the size and wiring complexity of the pullup network are major concerns but speed and power are less important. We will see two examples of uses of pseudo-nMOS circuits in Chapter 6: busses and PLAs. In both cases, we are building **distributed NOR gates**—we use pulldowns spread over a large physical area to compute the output, and we do not want to have to run the signals which control the pulldowns around this large area. Pseudo-nMOS circuits allow us to concentrate the logic gate's functionality in the pulldown network.

3.5.2 DCVS Logic

Differential cascode voltage switch logic (DCVSL) is a static logic family that, like pseudo-nMOS logic, does not have a complementary pullup network, but it has a very different structure. It uses a latch structure for the pullup which both eliminates static power consumption and provides true and complement outputs.

The structure of a generic DCVSL gate is shown in Figure 3-28. There are two pulldown networks which are the duals of each other, one for each true/complement output. Each pulldown network has a single p-type pullup, but the pullups are cross-coupled. Exactly one of the pulldown networks will create a path to ground when the gate's inputs change, causing the output nodes to switch to the required values. The cross-coupling of the pullups helps speed up the transition—if, for example, the com-

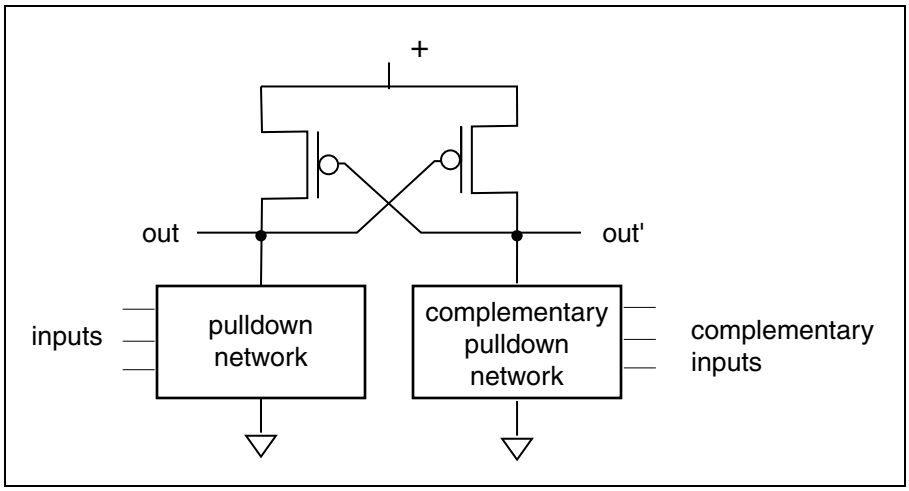


Figure 3-28: *Structure of a DCVSL gate.*

plementary network forms a path to ground, the complementary output goes toward V_{SS}, which turns on the true output's pullup, raising the true output, which in turn lowers the gate voltage on the complementary output's pullup. This gate consumes no DC power (except due to leakage current), since neither side of the gate will ever have both its pullup and pulldown network on at once.

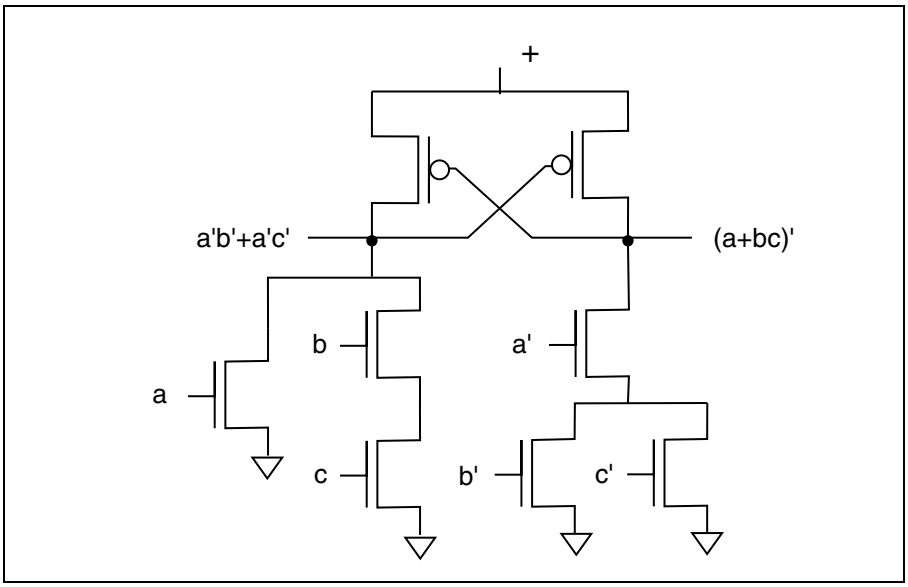


Figure 3-29: *An example DCVSL gate circuit.*

Figure 3-29 shows the circuit for a particular DCVSL gate. This gate computes $a+bc$ on one output and $(a+bc)' = a'b'+a'c'$ on its other output.

3.5.3 Domino Logic

Precharged circuits offer both low area and higher speed than static complementary gates. Precharged gates introduce functional complexity because they must be operated in two distinct phases, requiring introduction of a clock signal. They are also more sensitive to noise; their clocking signals also consume power and are difficult to turn off to save power.

The canonical precharged logic gate circuit is the **domino** circuit [Kra82]. A domino gate is shown in Figure 3-30, along with a sketch of its operation over one cycle. The gate works in two phases, first to precharge the storage node, then to selectively discharge it. The phases are controlled by the clock signal ϕ:

- **Precharge**. When ϕ goes low, the p-type transistor starts charging the precharge capacitance. The pulldown transistors controlled by the clock keep that precharge node from being drained. The length of the $\phi = 0$ phase is adjusted to ensure that the storage node is charged to a solid logic 1.
- **Evaluate**. When ϕ goes high, precharging stops (the p-type pullup turns off) and the evaluation phase begins (the n-type pulldowns at the bottom of the circuit turn on). The logic inputs a and b can now assume their desired value of 0 or 1. The input signals must monotonically rise—if an input goes from 0 to 1 and back to 0, it will inadvertently discharge the precharge capacitance. If the inputs create a conducting path through the pulldown network, the precharge capacitance is discharged, forcing its value to 0 and the gate's output (through the inverter) to 1. If neither a nor b is 1, then the storage node would be left charged at logic 1 and the gate's output would be 0.

The gate's logic value is valid at the end of the evaluation phase, after enough time has been allowed for the pulldown transistors to fully discharge the storage node. If the gate is to be used to compute another value, it must go through the precharge-evaluate cycle again.

Figure 3-31 illustrates the phenomenon which gave the domino gate its name. Since each gate is precharged to a low output level before evaluation, the changes at the primary inputs ripple through the domino network from one end to another. Signals at the far end of the network change last, with each change to a gate output causing a

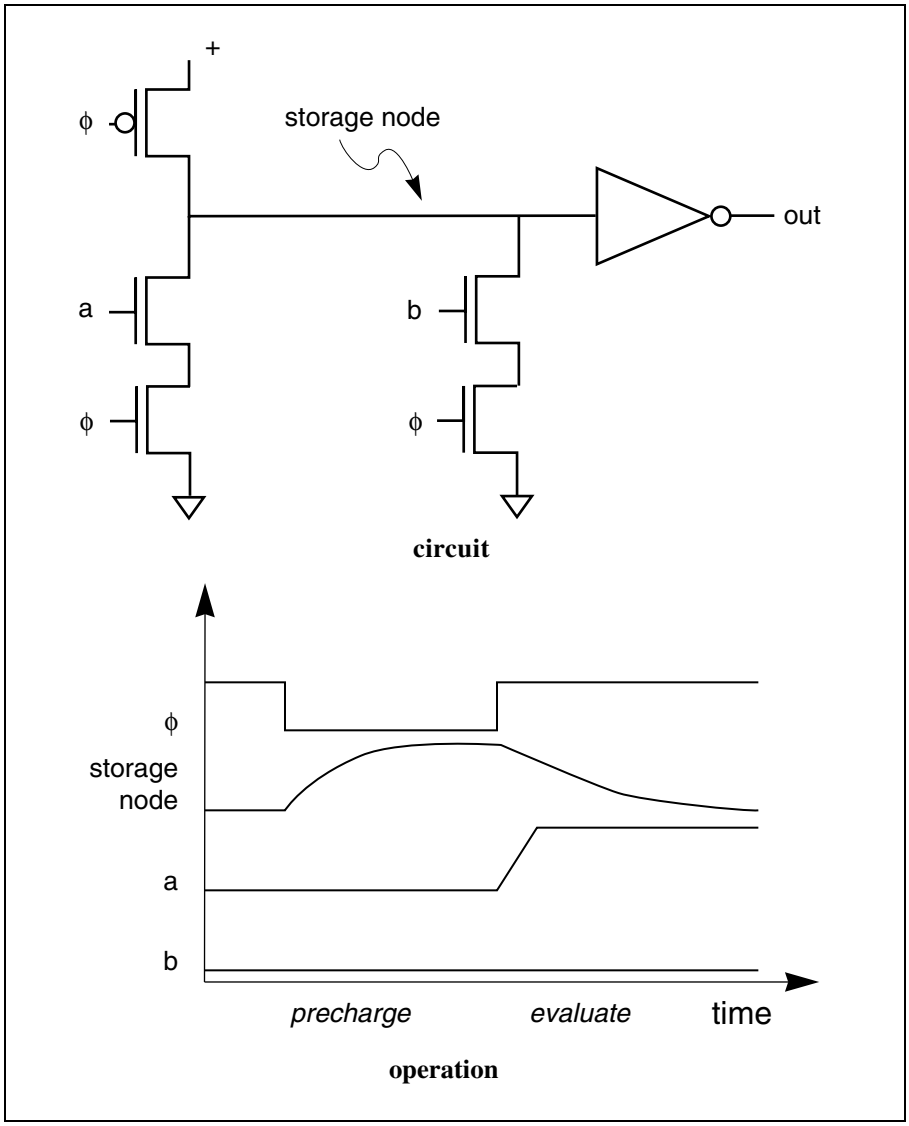


Figure 3-30: *A domino OR gate and its operation.*

change to the next output. This sequential evaluation resembles a string of falling dominos.

Why is there an inverter at the output of the domino gate? There are two reasons: logical operation and circuit behavior. To understand the logical need for an output inverter, consider the circuit of Figure 3-32, in which the output of one domino gate is

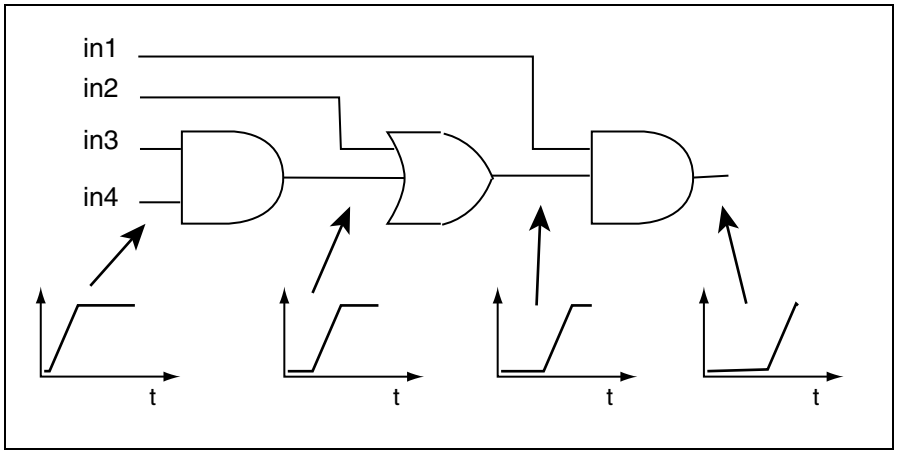


Figure 3-31: *Successive evaluations in a domino logic network.*

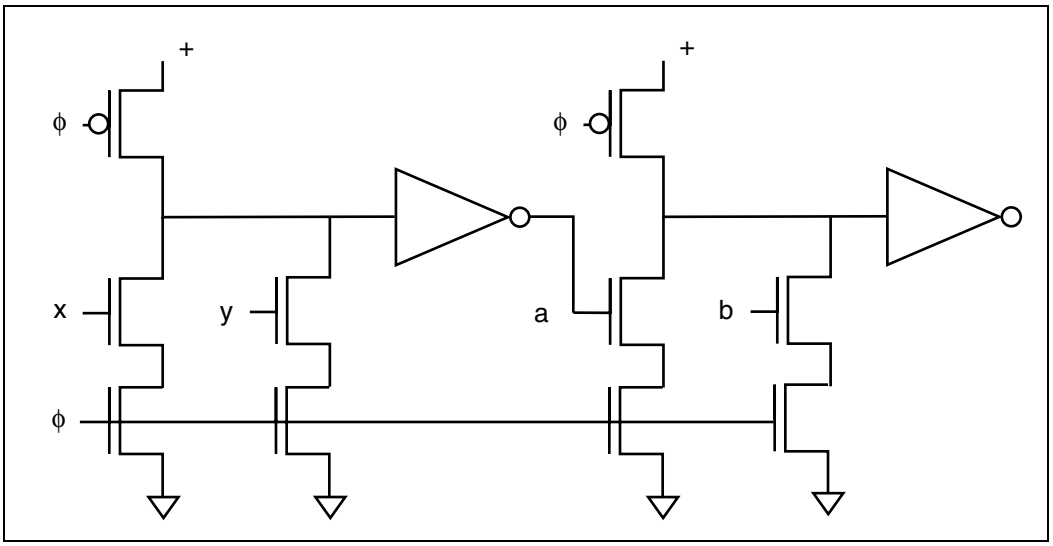


Figure 3-32: *Why domino gate input values must monotonically increase.*

fed into an input of another domino gate. During the precharge phase, if the inverter were not present, the intermediate signal would rise to 1, violating the requirement that all inputs to the second gate be 0 during precharging.

However, the more compelling reason for the output inverter is to increase the reliability of the gate. Figure 3-33 shows two circuit variations: one with the output inverter and one without. In both cases, the storage node is coupled to the output of the following gate by the gate-to-source/drain capacitances of the transistors in that

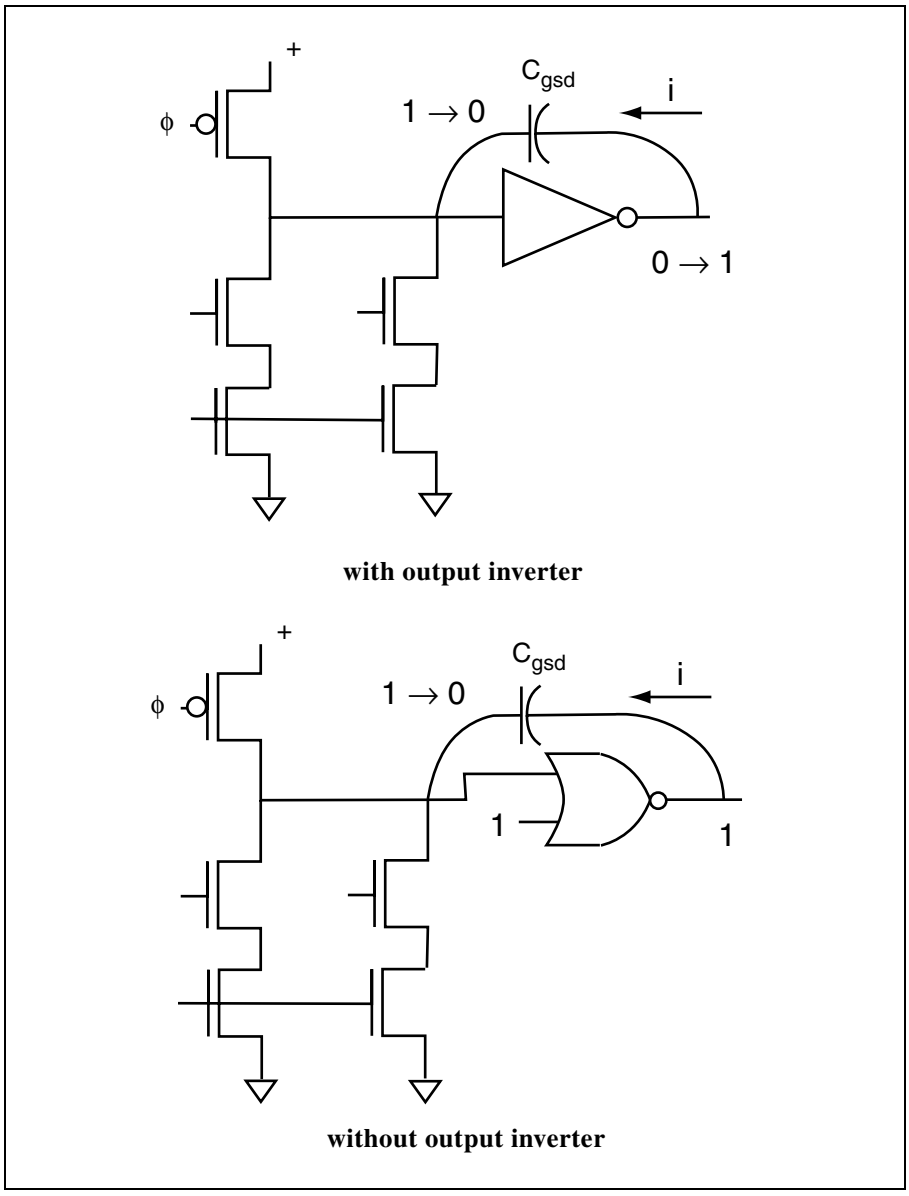


Figure 3-33: *Capacitive coupling in domino gates.*

gate. This coupling can cause current to flow into the storage node, disturbing its value. Since the coupling capacitance is across the transistor, the Miller effect magnifies its value. When the storage node is connected to the output inverter, the inverter's output is at least correlated to the voltage on the storage node and we can design the circuit to withstand the effects of the coupling capacitance. However, when the stor-

age node is connected to an arbitrary gate, that gate's output is not necessarily correlated to the storage node's behavior, making it more difficult to ensure that the storage node is not corrupted. The fact that the wire connecting the domino gate's pulldown network to the next gate (and the bulk of the storage node capacitance) may be long and subject to crosstalk generated by wire-to-wire coupling capacitances only makes this circuit less attractive.

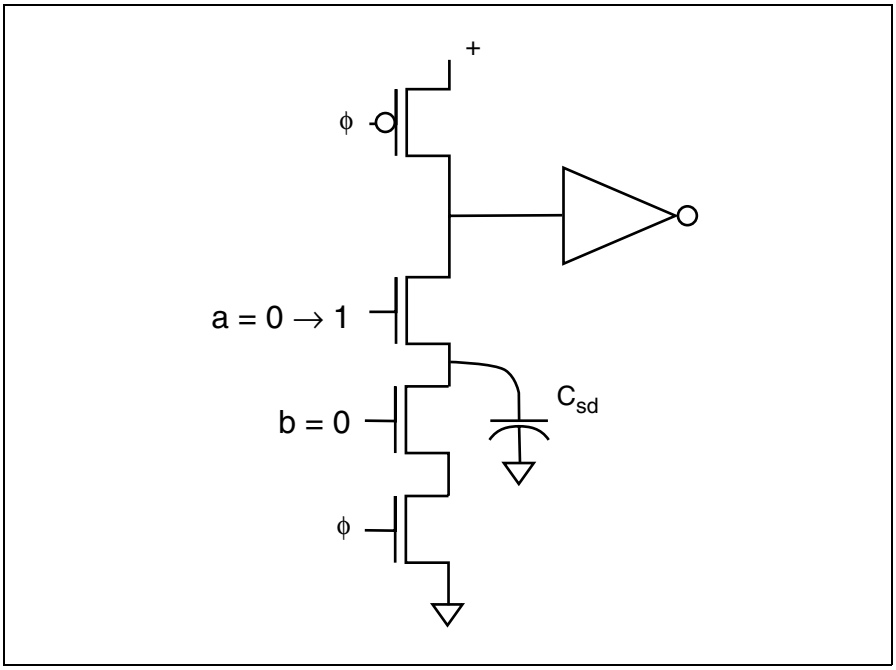


Figure 3-34: *Charge sharing in a domino circuit.*

Domino gates are also vulnerable to errors caused by **charge sharing**. Charge sharing is a problem in any network of switches, and we will cover it in more detail in Section 4.7. However, we need to understand the phenomenon in the relatively simple form in which it occurs in domino gates. Consider the example of Figure 3-34. C_{sd}, the stray capacitance on the source and drain of the two pulldown transistors, can store enough charge to cause problems. In the case when the a input is 1 and the b input is 0, the precharge node should not be discharged. However, since a is one, the pulldown connected to the storage node is turned on, draining charge from the storage node into the parasitic capacitance between the two pulldowns. In a static gate, charge stored in the intermediate pulldown capacitances does not matter because the power supply drives the output, but in the case of a dynamic gate that charge is lost to the storage node. If the gate has several pulldown transistors, the charge loss is that much more severe. The problem can be averted by precharging the internal pulldown net-

work nodes along with the precharge node itself, although at the cost of area and complexity.

Because dynamic gates rely on stored charge, they are vulnerable to charge leakage through the substrate. The primary threat comes from designs which do not evaluate some dynamic gates on every clock cycle; in these cases, the designer must verify that the gates are always re-evaluated frequently enough to ensure that the charge stored in the gates has not leaked away in sufficient quantities to destroy the gate's value.

Domino gates cannot invert, and so this logic family does not form a complete logic, as defined in Section 3.2. A domino logic network consists only of AND, OR, and complex AND/OR gates. However, any such function can be rewritten using De Morgan's laws to push all the inverters to the forward outputs or backward to the inputs; the bulk of the function can be implemented in domino gates with the inverters implemented as standard static gates. However, pushing back the inversions to the primary inputs may greatly increase the number of gates in the network.

3.6 Low-Power Gates

There are several different strategies for building low-power gates. Which one is appropriate for a given design depends on the required performance and power as well as the fabrication technology. In very deep submicron technologies leakage current has become a major consumer of power.

Of course, the simplest way to reduce the operating voltage of a gate is to connect it to a lower power supply. We saw the relationship between power supply voltage and power consumption in Section 3.3.5:

- For large V_t, Equation 3-10 tells us that delay changes linearly with power supply voltage.
- Equation 3-17 tells us that power consumption varies quadratically with power supply voltage.

This simple analysis tells us that reducing the power supply saves us much more in power consumption than it costs us in gate delay. Of course, the performance penalty incurred by reducing the power supply voltage must be taken care of somewhere in the system. One possible solution is architecture-driven voltage scaling, which we

will study in Section 8.5, which replicates logic to make up for slower operating speeds.

It is also possible to operate different gates in the circuit at different voltages: gates on the critical delay path can be run at higher voltages while gates that are not delay-critical can be run at lower voltages. However, such circuits must be designed very carefully since passing logic values between gates running at different voltages may run into noise limits.

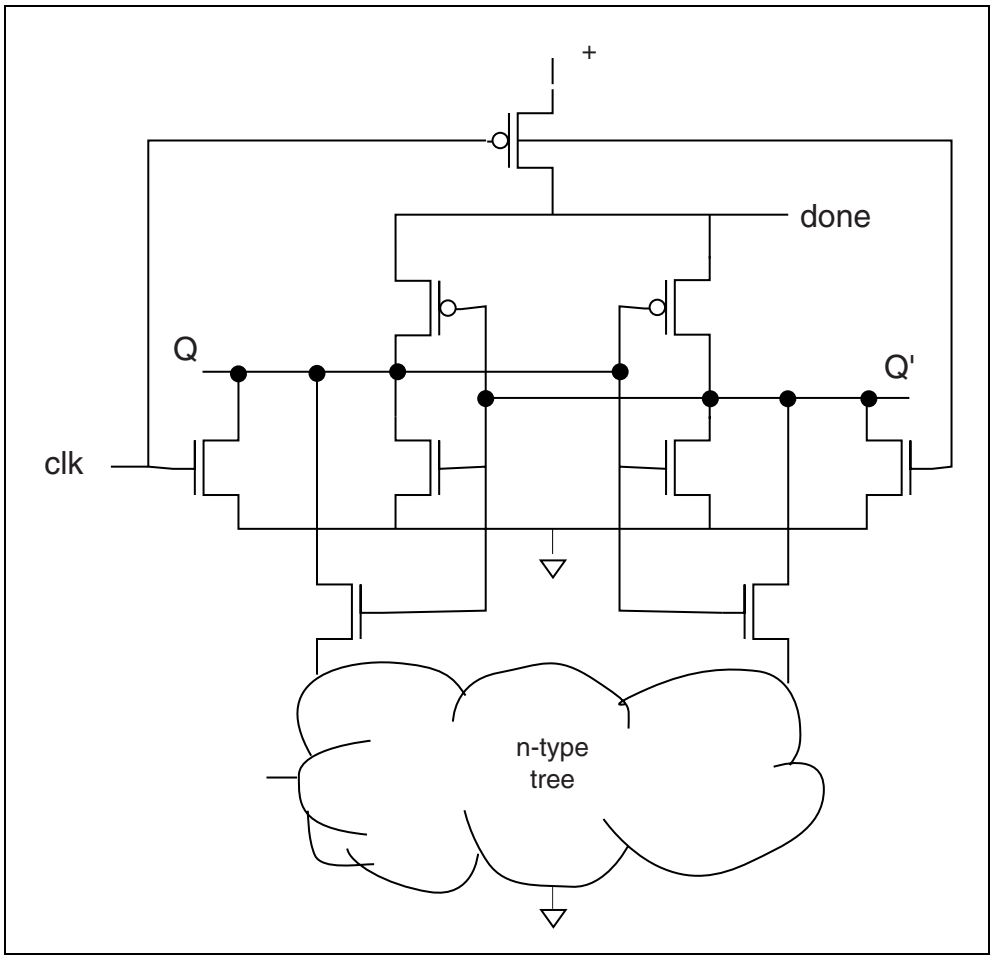


Figure 3-35: *A DCSL gate.*

After changing power supply voltages, the next step is to use different logic gate topologies. An example of this strategy is the differential current switch logic (DCSL) gate [Roy00] shown in Figure 3-35 is related to the DCVS gate of Section 3.5.2. Both use nMOS pulldown networks for both logic 0 and logic 1. However, the DCSL gate disconnects the n-type networks to reduce their power consumption. This gate is precharged with Q and Q' low. When the clock goes high, one of Q or Q' will be pulled low by the n-type evaluation tree and that value will be latched by the cross-coupled inverters.

After these techniques have been tried, two techniques can be used: reducing leakage current and turning off gates when they are not in use. Leakage current is becoming increasingly important in very deep submicron technologies. We studied leakage currents in Section 2.3.6. One simple approach to reducing leakage currents in gates is to choose, whenever possible, don't-care conditions on the inputs to reduce leakage currents. Series chains of transistors pass much lower leakage currents when both are off than when one is off and the other is on. If don't-care conditions can be used to turn off series combinations of transistors in a gate, the gate's leakage current can be greatly reduced.

The key to low leakage current is low threshold voltage. Unfortunately, there is an essential tension between low leakage and high performance. Remember from Equation 2-17 that leakage current is an exponential function of V_{gs} - V_t. As a result, increasing Vt decreases the subthreshold current when the transistor is off. However, a high threshold voltage increases the gate's delay since the transistor turns on later in the input signal's transition. One solution to this dilemma is to use transistors with different thresholds at different points in the circuit.

Turning off gates when they are not used saves even more power, particularly in technologies that exhibit significant leakage currents. Care must be used in choosing which gates to turn off, since it often takes 100 µs for the power supply to stabilize after it is turned on. We will discuss the implications of power-down modes in Section 8.5. However, turning off gates is a very useful technique that becomes increasingly important in very deep submicron technologies with high leakage currents.

The leakage current through a chain of transistors in a pulldown or pullup network is lower than the leakage current through a single transistor [De01]. It also depends on whether some transistors in the stack are also on. Consider the pulldown network of a NAND gate shown in Figure 3-36. If both the *a* and *b* inputs are 0, then both transistors are off. Because a small leakage current flows through transistor M_a, the parasitic capacitance between the two transistors is charged, which in turns holds the voltage at

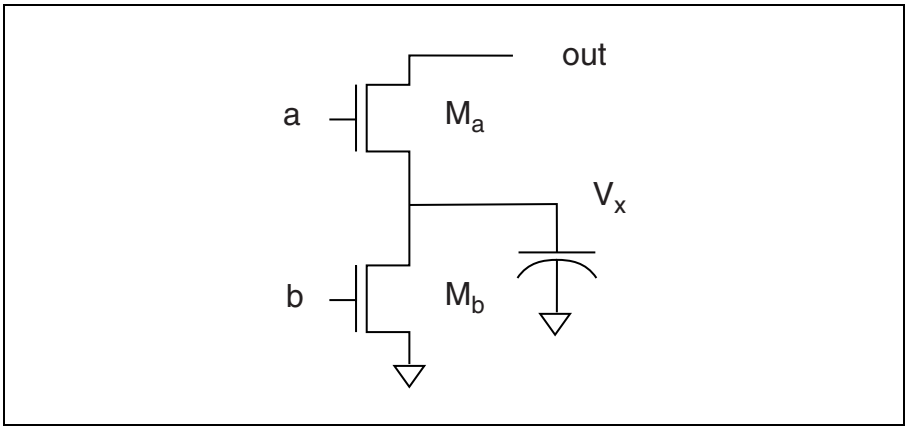


Figure 3-36: *Leakage through transistor stacks.*

that node above ground. This means that V_{gs} for is M_a is negative, thus reducing the total leakage current. The leakage current is found by simultaneously solving for the currents through the two transistors. The leakage current through the chain can be an order of magnitude lower than the leakage current through a single transistor. But the total leakage current clearly depends on the gate voltages of the transistors in the chain; if some of the gate's inputs are logic 1, then there may not be chains of transistors that are turned off and thus have reduced input voltages. Algorithms can be used to find the lowest-leakage input values for a set of gates; latches can be used to hold the gates' inputs at those values in standby mode to reduce leakage.

Figure 3-37 shows a **multiple-threshold logic** (**MTCMOS**) [Mut98] gate that can be powered down. This circuit family uses low-leakage transistors to turn off gates when they are not in use. A **sleep transistor** is used to control the gate's access to the power supply; the gated power supply is known as a **virtual V_{DD}**. The gate uses low-threshold transistors to increase the gate's delay time. However, lowering the threshold voltage also increases the transistors' leakage current, which causes us to introduce the sleep transistor. The sleep transistor has a high threshold to minimize its leakage. The fabrication process must be able to build transistors with low and high threshold voltages.

The layout of this gate must include both V_{DD} and virtual V_{DD}: virtual V_{DD} is used to power the gate but V_{DD} connects to the pullup's substrate. The layout must include The sleep transistor must be properly sized. If the sleep transistor is too small, its impedance would cause virtual V_{DD} to bounce. If the sleep transistor is too large, the sleep transistor would occupy too much area and it would use more energy when switched.

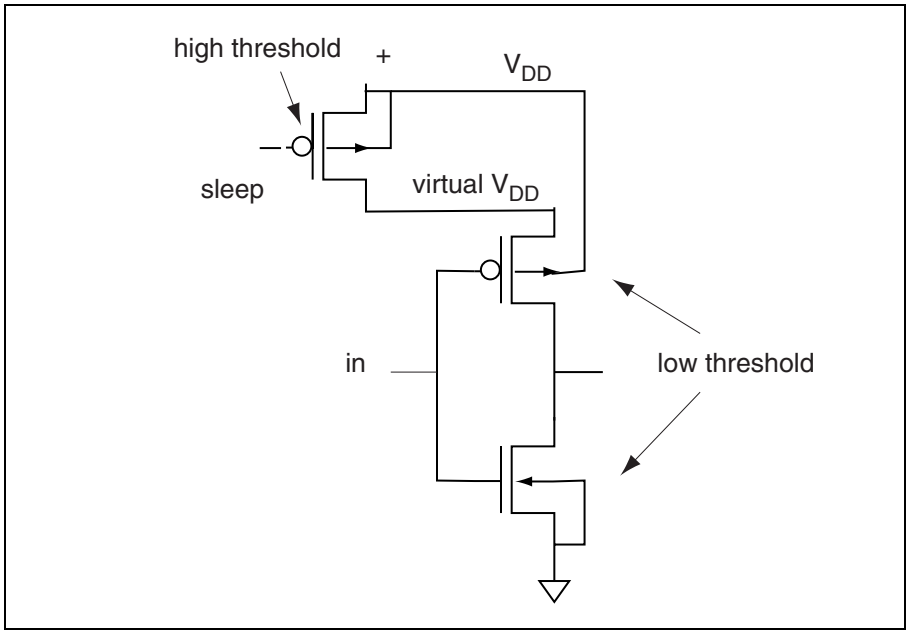


Figure 3-37: *A multiple-threshold (MTCMOS) inverter.*

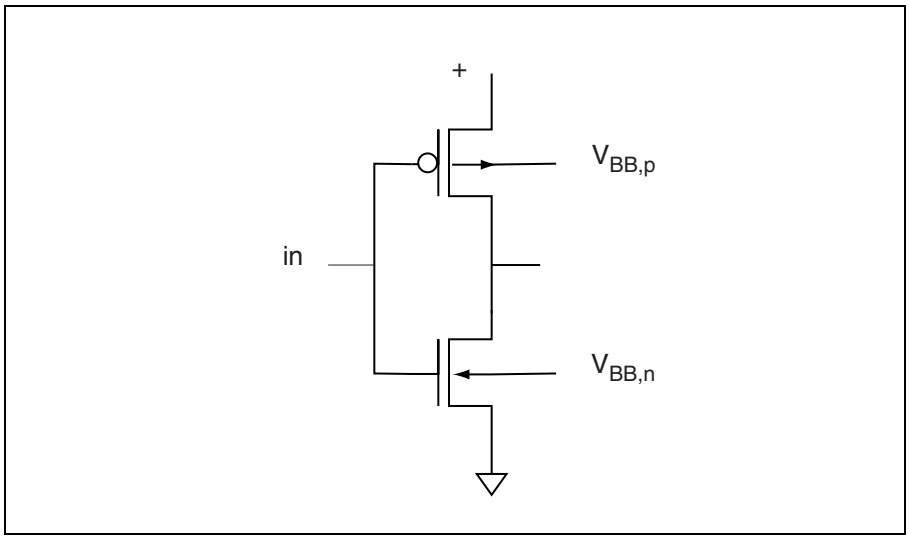


Figure 3-38: *A variable-threshold CMOS (VTCMOS) gate.*

It is important to remember that some other logic must be used to determine when a gate is not used and control the gate's power supply. This logic must be watch the

state of the chip's inputs and memory elements to know when logic can safely be turned off. It may also take more than one cycle to safely turn on a block of logic.

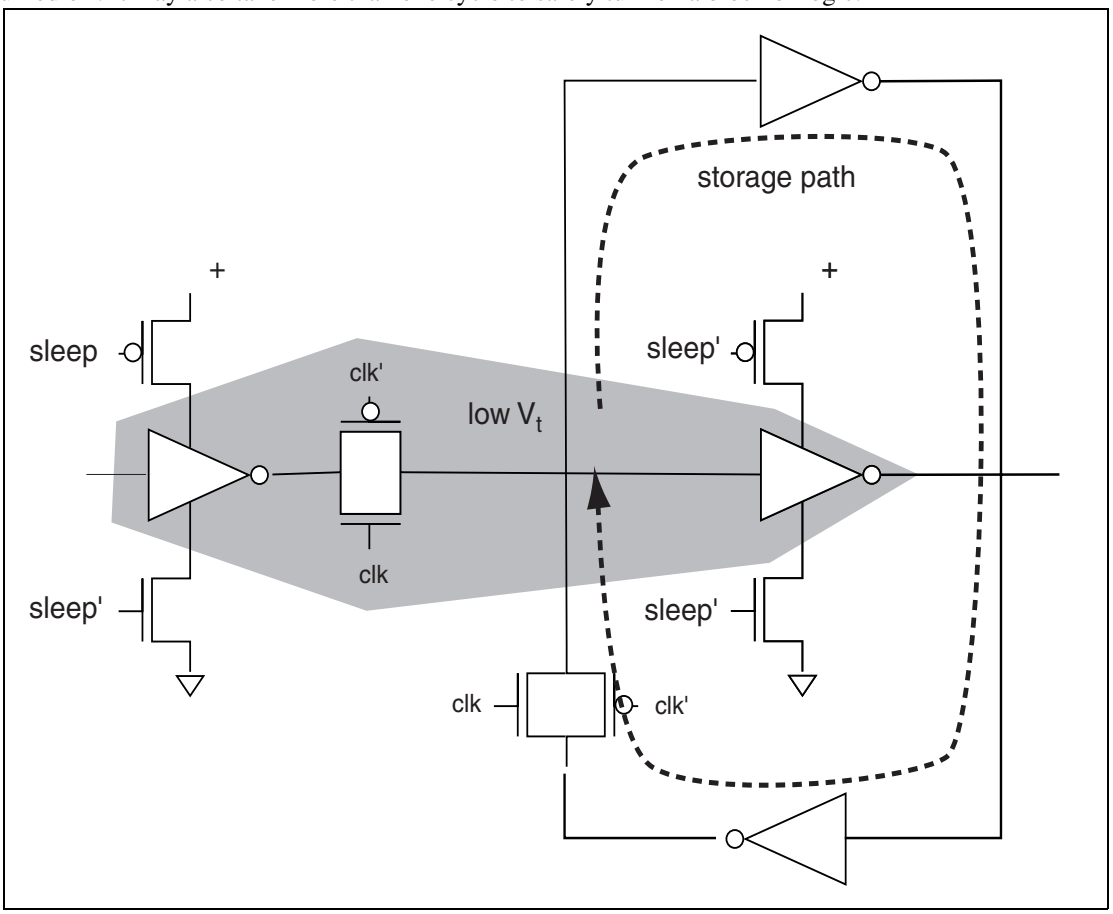


Figure 3-39: *An MTCMOS flip-flop.*

Figure 3-39 shows an MTCMOS flip-flop. The storage path is made of high V_t transistors and is always on. The signal is propagated from input to output through low V_t transistors. The sleep control transistors on the second inverter in the forward path to prevent a short-circuit path between V_{DD} and virtual V_{DD} that could flow through the storage inverter's pullup and the forward chain inverter's pullup.

A more aggressive method is **variable threshold CMOS** (**VTCMOS**) [Kur96], which actually can be implemented in several ways. Rather than fabricating fixed-

threshold voltage transistors, the threshold voltages of the transistors in the gate are controlled by changing the voltages on the substrates. Figure 3-38 shows the structure of a VTCMOS gate. The substrates for the p- and n-type transistors are each connected to their own threshold supply voltages, $V_{BB,p}$ and $V_{BB,n}$. V_{BB} is raised to put the transistor in standby mode and lowered to put it into active mode. Rather sophisticated circuitry is used to control the substrate voltages.

VTCMOS logic comes alive faster than it falls asleep. The transition time to sleep mode depends on how quickly current can be pulled out of the substrate, which typically tens to hundreds of microseconds. Returning the gate to active mode requires injecting current back into the substrate, which can be done 100 to 1000 times faster than pulling that current out of the substrate. In most applications, a short wake-up time is important—the user generally gives little warning that the system is needed.

3.7 Delay Through Resistive Interconnect

In this section we analyze the delay through resistive (non-inductive) interconnect. In many modern chips, the delay through wires is larger than the delay through gates, so studying the delay through wires is as important as studying delay through gates. We will build a suite of analytical models, starting from the relatively straightforward Elmore model for an RC transmission line through more complex wire shapes. We will also consider the problem of where to insert buffers along wires to minimize delay.

3.7.1 Delay Through an RC Transmission Line

An **RC transmission line** models a wire as infinitesimal RC sections, each representing a differential resistance and capacitance. Since we are primarily concerned with RC transmission lines, we can use the transmission line model to compute the delay through very long wires. We can model the transmission line as having unit resistance r and unit capacitance c. The standard schematic for the RC transmission line is shown in Figure 3-40. The transmission line's voltage response is modeled by a differential equation:

$$\frac{1}{r}\frac{d^2V}{dx^2} = c\frac{dV}{dt}. \qquad \textbf{(EQ 3-23)}$$

This model gives the voltage as a function of both x position along the wire and time.

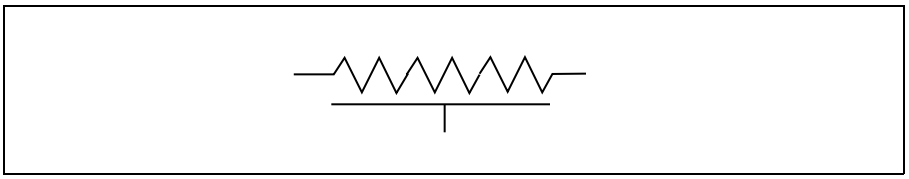


Figure 3-40: *Symbol for a distributed RC transmission line.*

The raw differential equation, however, is unwieldy for many circuit design tasks. **Elmore delay** [Elm48] is the most widely used metric for RC wire delay and has been shown to sufficiently accurately model the results of simulating RC wires on integrated circuits [Boe93]. Elmore defined the delay through a linear network as the first moment of the impulse response of the network:

$$\delta_E = \int_0^\infty t V_{out}(t) dt \ . \qquad \textbf{(EQ 3-24)}$$

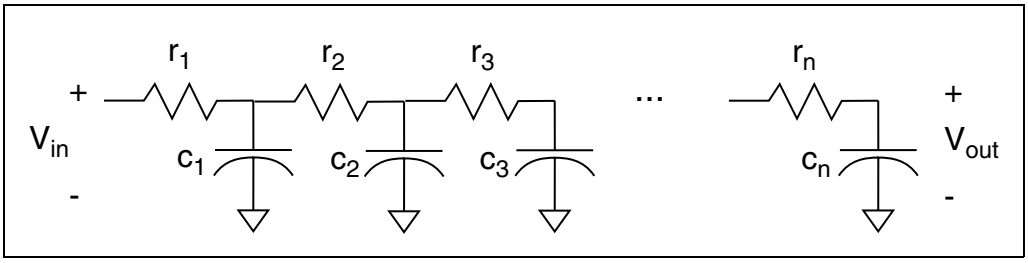


Figure 3-41: *An RC transmission line for Elmore delay calculations.*

It is because only the first moment is used that Elmore delay is not sufficiently accurate for inductive interconnect. However, in overdamped RC networks, the first moment is sufficiently accurate.

Elmore modeled the transmission line as a sequence of n sections of RC, as shown in Figure 3-41. In the case of a general RC network, the Elmore delay can be computed by taking the sum of RC products, where each resistance R is multiplied by the sum of all the downstream capacitors (a special case of the RC tree formulas we will introduce in Section 3.7.2). Since all the transmission line section resistances and capacitances in an n-section are identical, this reduces to

$$\delta_E = \sum_{i=1}^{n} r(\text{n-i})c = \frac{1}{2}rc \times n(\text{n-i}). \quad \textbf{(EQ 3-25)}$$

One consequence of this formula is that wire delay grows as the square of wire length, since n is proportional to wire length. Since the wire's delay also depends on its unit resistance and capacitance, it is imperative to use the material with the lowest RC product (which will almost always be metal) to minimize the constant factor attached to the n^2 growth rate.

Although the Elmore delay formula is widely used, we will need some results from the analysis of continuous transmission lines for our later discussion of crosstalk. The normalized voltage step response of the transmission line can be written as

$$V(t) = 1 + \sum_{k=1}^{\infty} K_k e^{-\sigma_k t/RC} \approx 1 + K_1 e^{-\sigma_1 t/RC}, \quad \textbf{(EQ 3-26)}$$

where R and C are the total resistance and capacitance of the line. We will define R_t as the internal resistance of the driving gate and C_t as the load capacitance at the opposite end of the transmission line.

Sakurai [Sak93] estimated the required values for the first-order estimate of the step response as:

$$K_1 = \frac{-1.01(R_T + C_T + 1)}{R_T + C_T + \pi/4}, \quad \textbf{(EQ 3-27)}$$

$$\sigma_1 = \frac{1.04}{R_T C_T + R_T + C_T + (2/\pi)^2}, \quad \textbf{(EQ 3-28)}$$

where R_T and C_T are R_t/R and C_t/C, respectively.

So far, we have assumed that the wire has constant width. In fact, tapered wires provide lower delay. Consider the first resistance element in the transmission line—the current required to charge all the capacitance of the wire must flow through this resistance. In contrast, the resistance at the end of the wire handles only the capacitance at the end. Therefore, if we can decrease the resistance at the head of the wire, we can

decrease the delay through the wire. Unfortunately, increasing the resistance by widening the wire also increases its capacitance, making this a non-trivial problem to solve.

Fishburn and Schevon [Fis95] proved that the optimum-shaped wire has an exponential taper. If the source resistance is R_0, the sink capacitance is C_0, and the unit resistance and capacitance are R_s and C_s, the width of the wire as a function of distance is

$$w(x) = \frac{2C_0}{C_s L} W\left(\frac{L}{2}\sqrt{\frac{R_s C_s}{R_0 C_0}}\right) e^{2W\left(\frac{L}{2}\sqrt{\frac{R_s C_s}{R_0 C_0}}\right)\frac{x}{L}}, \qquad \textbf{(EQ 3-29)}$$

where W is the function that satisfies the equality $W(x)e^{W(x)} = x$. The advantage of optimal tapering is noticeable. Fishburn and Schevon calculate that, for one example, the optimally tapered wire has a delay of 3.72 ns while the constant-width wire with minimum delay has a delay of 4.04 ns. In this example, the optimally tapered wire shrinks from 30.7 µm at the source to 7.8 µm at the sink.

Of course, exponentially-tapered wires are impossible to fabricate exactly, but it turns out that we can do nearly as well by dividing the wire into a few constant width sections. Figure 3-42 shows that a few segments of wire can be used to approximate the exponential taper reasonably well. This result also suggests that long wires which can be run on several layers should run on the lowest-resistance layer near the driver and can move to the higher-resistance layers as they move toward the signal sink.

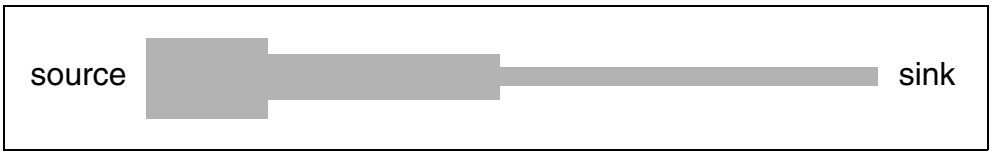


Figure 3-42: *A step-tapered wire.*

3.7.2 Delay Through RC Trees

While analyzing a straight transmission line is straightforward, analyzing more complex networks is harder. We may not always need an exact answer, either—a good approximation is often enough considering the other uncertainties in IC design and manufacturing. In the case of RC trees, as shown in Figure 3-43, we can quickly compute accurate bounds on the delay through the wire [Rub83]. The wiring can be bro-

ken into an RC tree either by representing each branch by one RC lump or by breaking a branch into several lumps.

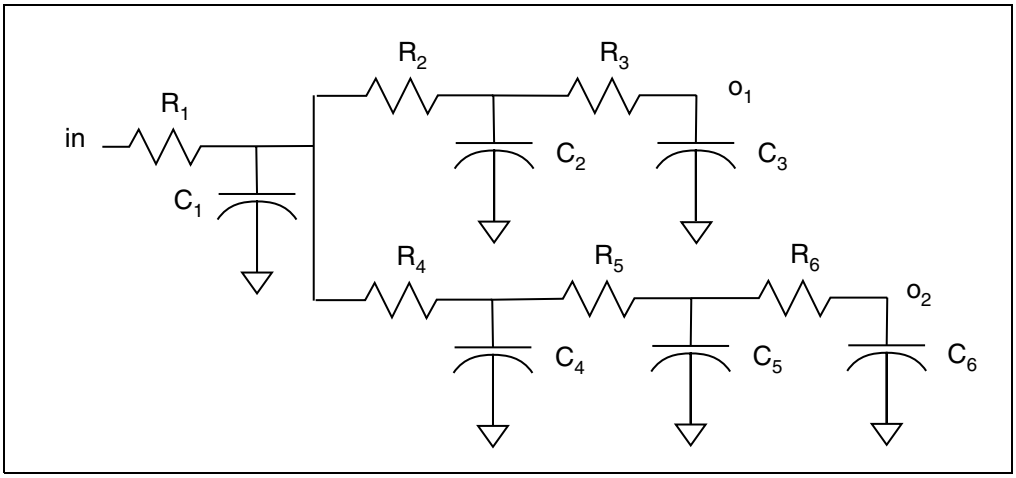


Figure 3-43: *An RC tree.*

When analyzing the RC tree, we assume the network has one input, which provides a voltage step, and several outputs. We can find the delay through the wire by analyzing the voltages at the output nodes and measuring the time between the 10% and 90% points. While an exact solution for the output voltages for an arbitrary RC network is complex, we can find accurate upper and lower bounds on the output voltage, and from those voltage bounds we can compute delay bounds. We won't perform a detailed derivation of the bounds formulas, but will only try to provide an intuitive explanation of their form.

The capacitance at a node k is called C_k. We are primarily concerned with resistances along paths, notably the resistances along shared paths. If o is an output node and k is an internal node, the resistance along the intersection of the paths from the input to o and to k is called R_{k0}. In Figure 3-43, $R_{1O_1} = R_1$ because R_1 is the only resistor shared by the paths to 1 and O_1. R_{00} is the total resistance from input to the output o and similarly, R_{kk} is the total resistance from input to the internal node k. The simplest time constant for the tree is

$$T_P = \sum_k R_{kk} C_k . \qquad \textbf{(EQ 3-30)}$$

Each term in the summation is the time constant of the simple *RC* circuit built from the capacitance at k and all the resistance from the input to k. Two other time constants relative to the output o are important to the bounds:

$$T_{Do} = \sum_k R_{ko} C_k ; \quad \textbf{(EQ 3-31)}$$

$$T_{Ro} = \left(\sum_k R_{ko}^2 C_k \right) / R_{oo} . \quad \textbf{(EQ 3-32)}$$

The terms of T_{D0} compute the time constant of the capacitance at each node and the resistance shared by the paths to k and o available to charge C_k. The terms of T_{R0} weight the terms of T_{D0} against the total resistance along the path to the output, squaring R_{k0} to ensure the value has units of time. Although we won't prove it here, these inequalities relate the voltage at each output, $v_0(t)$, and the voltage at an interior node, $v_k(t)$, using the path resistances:

$$R_{oo}[1\text{-}v_k(t)] \geq R_{ko}[1\text{-}v_o(t)] \quad \textbf{(EQ 3-33)}$$

$$R_{ko}[1\text{-}v_k(t)] \leq R_{kk}[1\text{-}v_o(t)] \quad \textbf{(EQ 3-34)}$$

Some intermediate steps are required to find the $v_o(t)$'s; we will skip to the resulting bounds, shown in Table 3-2. The bounds are expressed both as the voltage at a given time and as the time required for the output to assume a specified voltage; the two formulas are, of course, equivalent.

Do these bounds match our intuition about the circuit's behavior? At t=0, the upper bound for the output voltage is $v_o(0) = 1 - T_{D0}$. T_{D0} is formed by the time constants of RC sections formed by all the resistance along the path to o that are also connected to the k^{th} capacitor, such as the highlighted resistors at a in the figure. Some of the current through those resistors will go to outputs other than o, and so are not available to charge the capacitors closest to o; the upper bound assumes that *all* their current will be used to charge capacitors along the path from input to o. The lower bound is dominated by T_{R0}, which compares R_{k0} to the total resistance from the input to o; the ratio R_{k0}/R_{00} gives a minimum resistance available to charge the capacitor C_k.

	validity	bound
lower	$t \le T_{Do}\text{-}T_{Ro}$ $T_{Do}\text{-}T_{Ro} \le t \le T_p\text{-}T_{Ro}$ $t \ge T_p\text{-}T_{Ro}$	$v_o(t) \ge 0$ $v_o(t) \ge 1\text{-}[T_{Do}/(t + T_{Ro})]$ $v_o(t) \ge 1\text{-}\frac{T_{Do}}{T_p} e^{(T_p\text{-}T_{Ro})/T_P} e^{-(t/T_p)}$
upper	$t \le T_{Do}\text{-}T_{Ro}$ $t \ge T_{Do}\text{-}T_{Ro}$	$v_o(t) \le 1\text{-}((T_{Do}\text{-}t)/T_P)$ $v_o(t) \le 1\text{-}\frac{T_{Ro}}{T_p} e^{(T_{Do}\text{-}T_{Ro})/T_{Ro}} e^{-(t/T_p)}$

	validity	bound
lower	$v_o(t) \le 1\text{-}(T_{Ro}/T_p)$ $v_o(t) \ge 1\text{-}(T_{Ro}/T_p)$	$t \ge T_{Do}\text{-}T_p[1\text{-}v_o(t)]$ $t \ge T_{Do}\text{-}T_{Ro} + T_{Ro} ln(\frac{T_{Ro}}{T_p[1\text{-}v_o(t)]})$
upper	$v_o(t) \le 1\text{-}(T_{Do}/T_p)$ $v_o(t) \ge 1\text{-}(T_{Do}/T_p)$	$t \le [T_{Do}/(1\text{-}v_o(t))]\text{-}T_{Ro}$ $t \le T_P\text{-}T_{Ro} + T_P ln(\frac{T_{Do}}{T_p[1\text{-}v_o(t)]})$

Table 3-2 *Rubinstein-Penfield-Horowitz voltage and time bounds for RC trees.*

3.7.3 Buffer Insertion in RC Transmission Lines

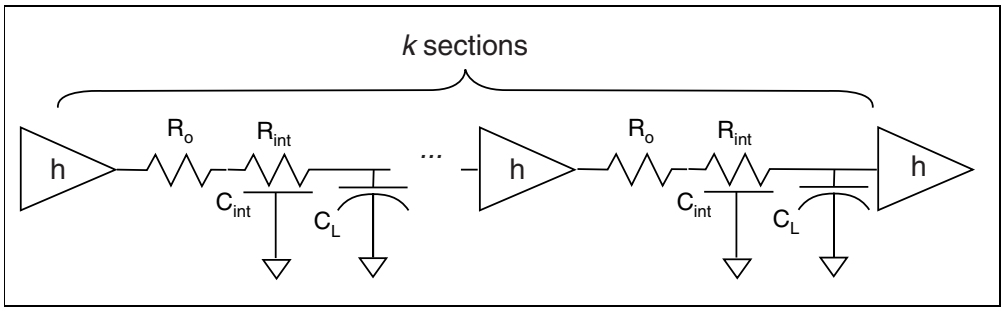


Figure 3-44: *An RC transmission line with repeaters.*

We do not obtain the minimum delay through an RC transmission line by putting a single large driver at the transmission line's source. Rather, we must put a series of buffers equally spaced through the line to restore the signal. Bakoglu [Bak90] derived the optimal number of repeaters and repeater size for an RC transmission line. As shown in Figure 3-44, we want to divide the line into k sections, each of length l. Each buffer will be of size h.

Let's first consider the case in which h=1 and the line is broken into k sections. R_{int} and C_{int} are the total resistance and capacitance of the transmission line. R_0 is the driver's equivalent resistance and C_0 its input capacitance. Then the 50% delay formula is

$$T_{50\%} = k\left[0.7R_0\left(\frac{C_{\text{int}}}{k} + C_0\right) + \frac{R_{\text{int}}}{k}\left(0.4\frac{C_{\text{int}}}{k} + 0.7C_0\right)\right] \qquad \textbf{(EQ 3-35)}$$

The various coefficients are due to the distributed nature of the transmission line. We find the minimum delay by setting $dT/dk = 0$. This gives the number of repeaters as

$$k = \sqrt{\frac{0.4R_{\text{int}}C_{\text{int}}}{0.7R_0C_0}}. \qquad \textbf{(EQ 3-36)}$$

When we free the size of the repeater to be an arbitrary value h, the delay equation becomes

$$T_{50\%} = k\left[0.7\frac{R_0}{h}\left(\frac{C_{\text{int}}}{k} + hC_0\right) + \frac{R_{\text{int}}}{k}\left(0.4\frac{C_{\text{int}}}{k} + 0.7hC_0\right)\right]. \quad \textbf{(EQ 3-37)}$$

We solve for minimum delay by setting $\frac{dT}{dk} = 0$ and $\frac{dT}{dh} = 0$. This gives the optimal values for k and h as

$$k = \sqrt{\frac{0.4R_{\text{int}}C_{\text{int}}}{0.7R_0C_0}}, \quad \textbf{(EQ 3-38)}$$

$$h = \sqrt{\frac{R_0C_{\text{int}}}{R_{\text{int}}C_0}}. \quad \textbf{(EQ 3-39)}$$

The total delay at these values is

$$T_{50\%} = 2.5\sqrt{R_0C_0R_{\text{int}}C_{\text{int}}}. \quad \textbf{(EQ 3-40)}$$

Example 3-3: *Buffer insertion in an RC line*

Let's calculate the buffers required when a minimum-size inverter drives a metal 1 wire that is 2000 λ x 3 λ. In this case, R_0 = 3.9kΩ and C_0 = 0.68 fF while R_{int} = 53.3 Ω and C_{int} = 15 fF + 90.1 fF = 105.1 fF. The optimal number of buffers is

$$k = \sqrt{\frac{0.4 \times 53.33 \times 105.1\times10^{-15}}{0.7 \times 3900 \times 0.68\times10^{-15}}} = 1.099.$$

The optimal buffer size is

$$h = \sqrt{\frac{3900 \times 105.1\times10^{-15}}{53.33 \times 0.68\times10^{-15}}} = 106.33.$$

The 50% delay is

$$T_{50\%} = 2.5\sqrt{3900 \times 0.68\times10^{-15} \times 53.33 \times 105.1\times10^{-15}} = 9.64\times10^{-12}.$$

If we increase the size of the driver by a factor of 4, reducing its resistance by 4X and increasing its capacitance by 4X, what happens? k and $T_{50\%}$ remain unchanged, but the buffer size drops by a factor of 4.

3.7.4 Crosstalk Between RC Wires

Crosstalk is important to analyze because it slows down signals—the crosstalk noise increases the signal's settling time. Crosstalk can become a major component of delay if wiring is not carefully designed.

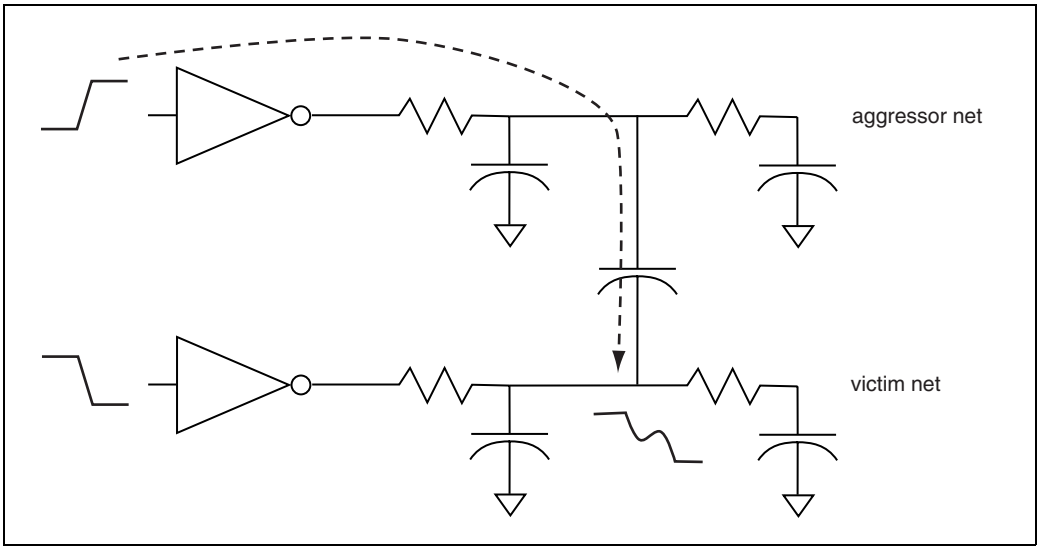


Figure 3-45: *Aggressor and victim nets.*

Figure 3-45 shows the basic situation in which crosstalk occurs. Two nets are coupled by parasitic capacitance. One net is the **aggressor net** that interferes with a **victim net** through that coupling capacitance. A transition in the aggressor net is transmitted to the victim net causing the victim to glitch. The glitch causes the victim net to take longer to settle to its final value. In static combinational logic, crosstalk increases the

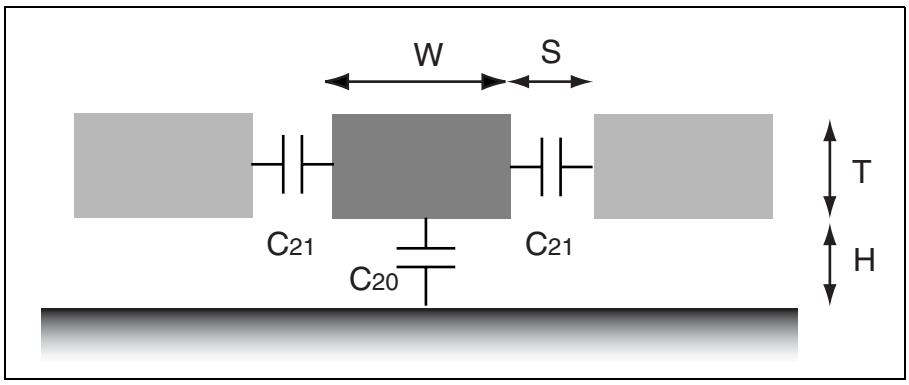


Figure 3-46: *A simple crosstalk model (after Sakurai [Sak93], © 1993 IEEE).*

delay across a net; in dynamic logic, crosstalk can cause the state of a node to flip, causing a permanent error.

In this section we will develop basic analytical models for crosstalk; in Section 4.5.4 we will learn how to minimize crosstalk through routing techniques. The simplest case to consider is a set of three wires [Sak93], as shown in Figure 3-46. The middle wire carries the signal of interest, while the other two capactively inject crosstalk noise. Each wire is of height *T* and width *W*, giving an aspect ratio of *W/T*. Each wire is height *H* above the substrate and the wires are spaced a distance S apart. We must consider three capacitances: C_{20} between the signal wire and the substrate, and two capacitances of equal value, C_{21}, to the two interfering wires. We denote the sum of these three capacitances as C_3. Sakurai estimates the RC delay through the signal wire in arbitrary time units as

$$t_r = \frac{(C_{20} + 4C_{21})}{W/H}. \qquad \textbf{(EQ 3-41)}$$

Using this simple model, Figure 3-47 shows Sakurai's calculation of relative RC delay in arbitrary units for a 0.5 µm technology for the signal wire. This plot assumes that $T/H = 1$ and that the aspect ratio varies from near 0 through 4; the delay is shown for four different spacings between the wires, as given by the P/H ratio. This plot clearly shows two important results. First, there is an optimum wire width for any given wire spacing, as shown by the U shape of each curve. Second, the optimum width increases as the spacing between wires increases.

That analysis assumes that the signals on the surrounding wires are stable, which is the best case. In general, we must assume that the surrounding wires are in transition. Consider the model of Figure 3-48, in which we have two RC transmission lines with

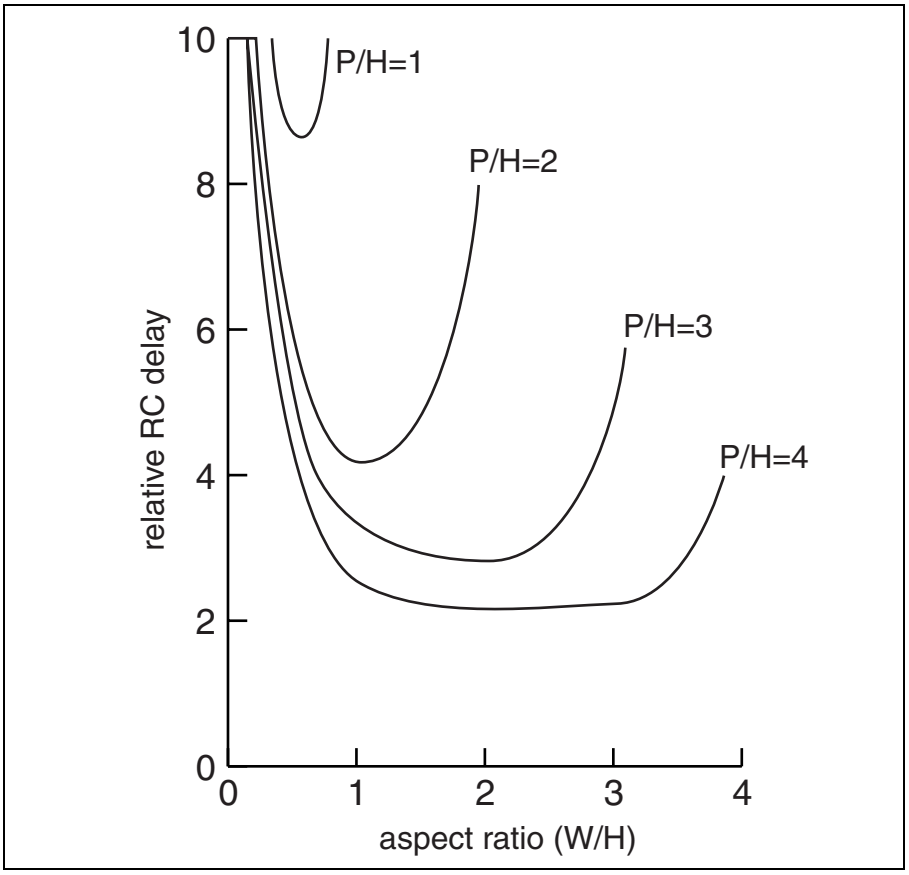


Figure 3-47: *Delay vs. wire aspect ratio and spacing (after Sakurai [Sak93] © 1993 IEEE).*

a coupling capacitance C_c between them. A step is applied to each wire at t=0, resulting in response waveforms at the opposite ends of the transmission lines [Sak93]. We assume that the unit resistances and capacitances of the two transmission lines are equal. Defining differential voltages between the two wires helps simplify the voltage response equations:

$$V_+ = \frac{V_1 + V_2}{\sqrt{2}}, V_- = \frac{V_1 - V_2}{\sqrt{2}}. \quad \textbf{(EQ 3-42)}$$

The voltage responses of the transmission lines can then be written as

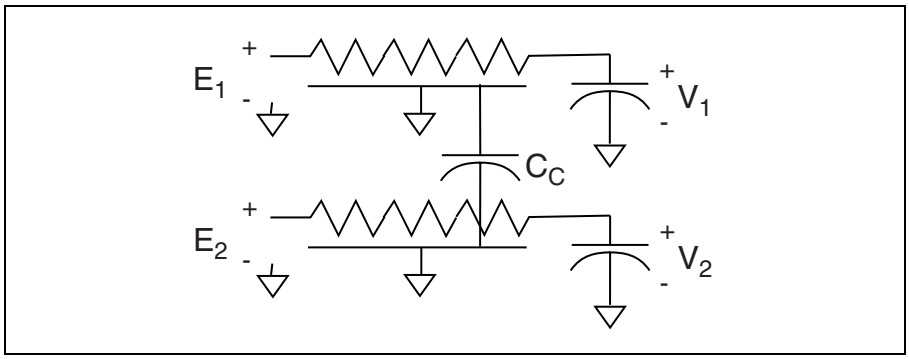


Figure 3-48: *Two coupled RC transmission lines.*

$$\frac{d^2V_+}{dx^2} = rc\frac{dV_+}{dt}, \qquad \textbf{(EQ 3-43)}$$

$$\frac{d^2V_-}{dx^2} = r(c + 2c_c)\frac{dV_-}{dt}. \qquad \textbf{(EQ 3-44)}$$

If we let $R = rl$, $C = cl$, and $C_c = c_c l$, then the voltage responses V_1 and V_2 at the ends of the transmission lines can be written as

$$V_1(t) \approx E_1 + \frac{K_1}{2}[(E_1 + E_2)e^{-\sigma_1 t/RC} + (E_1 - E_2)e^{-\sigma_1 t/RC + 2RC_c}] \quad , \qquad \textbf{(EQ 3-45)}$$

$$V_2(t) \approx E_2 + \frac{K_1}{2}[(E_1 + E_2)e^{-\sigma_1 t/RC} - (E_1 - E_2)e^{-\sigma_1 t/(RC + 2RC_c)}]. \qquad \textbf{(EQ 3-46)}$$

3.8 Delay Through Inductive Interconnect

Copper wiring provides much better performance, particularly for long wires. However, copper wires have significant inductance. Analyzing inductive wiring is more complicated than is analyzing RC transmission lines. RLC transmission lines have a more complex response that requires more subtle interpretation as well as more effort.

3.8.1 RLC Basics

First, let's review the basics of RLC circuits. A single RLC section is shown in Figure 3-49. The poles of the RLC section are at

$$\omega_0\left[\xi \pm \sqrt{\xi^2 - 1}\right] \quad \textbf{(EQ 3-47)}$$

where the damping factor ξ is defined as

$$\xi = \frac{R}{2}\sqrt{\frac{C}{L}}. \quad \textbf{(EQ 3-48)}$$

If the damping factor is greater than 1, the circuit is **overdamped** and responds to an impulse or step by monotonically approaching the final voltage. If the damping factor is less than 1, the circuit is **underdamped** and oscillates as it converges to the steady-state voltage. Underdamped circuits create a new challenge for digital circuit analysis because it is harder to find their rise times. For an underdamped circuit, we simply have to find the first time the waveform crosses the desired voltage threshold, knowing that it will always remain above that level. To determine the rise time of an underdamped circuit, we must find the last time at which the waveform falls below the threshold.

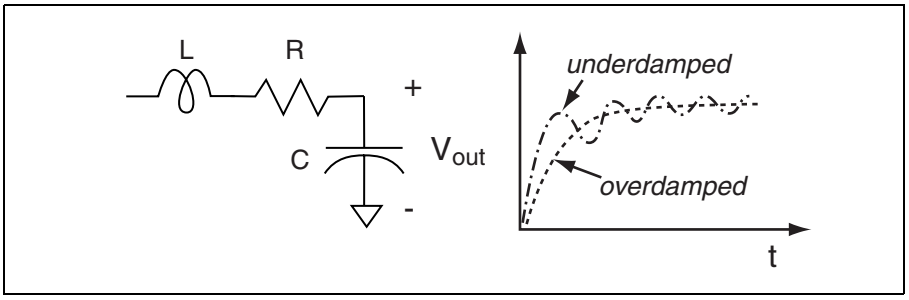


Figure 3-49: *An RLC circuit and its behavior.*

The simplest form of an RLC transmission is the lossless LC line with zero resistance. A signal propagates along an LC transmission line [Ram65] with velocity

$$v = \frac{1}{\sqrt{LC}}. \quad \textbf{(EQ 3-49)}$$

Therefore, the propagation delay through an LC transmission line of length l is $t_p = l\sqrt{LC}$. This value is a lower bound on the delay introduced by an RLC transmission line.

3.8.2 RLC Transmission Line Delay

In today's technology the resistance of the copper wiring cannot be ignored. Because we are designing digital systems, we are interested in an RLC transmission line that is being driven by a gate at one end and is connected to a receiving gate at the other end [Ism00]. We will model the driving gate as a resistance R_{tr} and the load gate as a capacitance C_L. We will use R, L and C for the unit resistance, inductance, and capacitance and R_t, L_t, and C_t for the total resistance, inductance, and capacitance of the line. The complete system is shown in Figure 3-50.

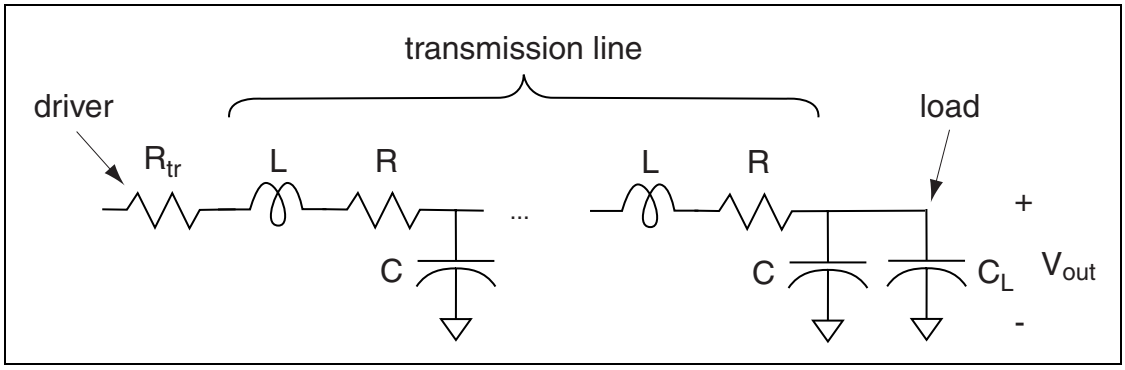


Figure 3-50: *An RLC transmission line with a driver and load.*

We can simplify our analysis by scaling time using the factor

$$\omega_n = \frac{1}{\sqrt{Ll(Cl + C_L)}}. \qquad \textbf{(EQ 3-50)}$$

We normalize time by substituting $t = t'/\omega_n$. We also need two additional values:

$$R_T = \frac{R_{\text{tr}}}{R_t}, \qquad \textbf{(EQ 3-51)}$$

$$C_T = \frac{C_L}{C_t}. \qquad \textbf{(EQ 3-52)}$$

where l is once again the length of the transmission line.

The complete derivation of the transmission line's response is rather complex, but we are most interested in the propagation delay through the wire to the load capacitance. Ismail and Friedman showed that propagation delay is primarily a function of ξ, which is defined as

$$\xi = \frac{R_t}{2}\sqrt{\frac{C_t}{L_t}}\left(\frac{Rl + Cl + RCl^2 + 0.5}{\sqrt{1 + Cl}}\right). \qquad \textbf{(EQ 3-53)}$$

They used numerical techniques to approximate the 50% propagation delay of our RLC transmission line as

$$t_{\text{pd}} = \left(e^{-2.9\xi^{1.35}} + 1.48\xi\right)/\omega_n. \qquad \textbf{(EQ 3-54)}$$

Figure 3-51 compares the response of RLC and RC wires for different values of ξ. These plots show that ignoring inductance results in very poor results for small values of ξ.

Figure 3-52 compares RC and RLC models for wires driven by inverters in a 0.25 µm technology. This figure shows that ignoring inductance results in serious errors in estimating delay for a variety of wire and driver configurations.

3.8.3 Buffer Insertion in RLC Transmission Lines

Ismail and Friedman also showed where to place buffers in an RLC transmission line [Ism00]. The circuit is shown in Figure 3-53. The transmission line is divided into k sections, each of length l/k. All the buffers are of the same size and are h times larger than a minimum-size buffer; we use R_0 and C_0 to represent the source resistance and load capacitance of a minimum-size buffer.

We can define

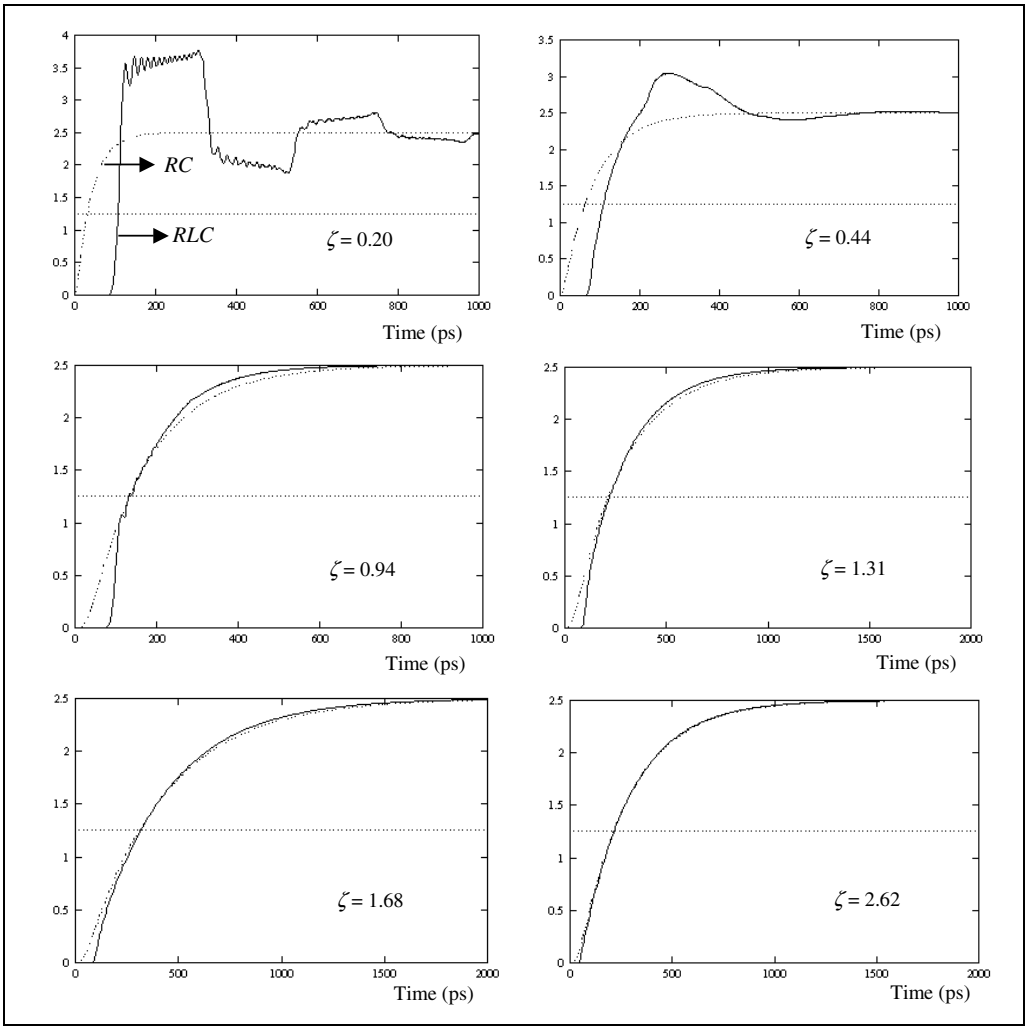


Figure 3-51: *RC vs. RLC models for interconnect for various values of ξ (from Ismail and Friedman [Ism00]). © 2000 IEEE.*

$$T_{L/R} = \sqrt{\frac{L_t/R_t}{R_0 C_0}}. \qquad \textbf{(EQ 3-55)}$$

As in the RC case, we are interested in determining the optimum drive per stage h_{opt} and the optimum length of each stage's wire k_{opt}. This optimization problem cannot

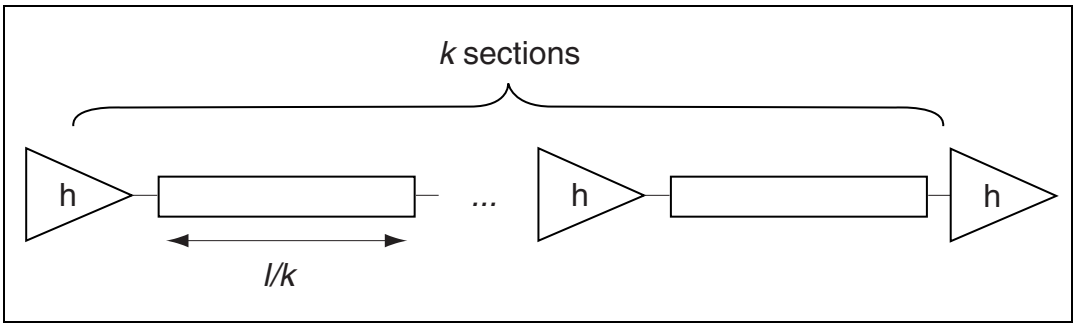


Figure 3-53: *Repeaters in an RLC transmission line.*

be solved analytically, but Ismail and Friedman fitted curves to the functions to provide these formula:

$$h_{\text{opt}} = \sqrt{\frac{R_0 C_t}{R_t C_0}} \frac{1}{[1 + 0.16(T_{\text{L/R}})^3]^{0.3}}, \qquad \textbf{(EQ 3-56)}$$

$$k_{\text{opt}} = \sqrt{\frac{R_t C_t}{2R_0 C_0}} \frac{1}{[1 + 0.18(T_{\text{L/R}})^3]^{0.3}}. \qquad \textbf{(EQ 3-57)}$$

3.9 References

Claude Shannon first described the relationship between Boolean logic and switching functions for his Master's thesis; his paper [Sha38] is still interesting reading. Hodges and Jackson [Hod83] give an excellent introduction to device characteristics and digital circuit design, showing how to analyze CMOS logic gates as well as design more complex digital circuits. Books by Rabaey [Rab96] and Uyemura [Uye92] are detailed presentations of digital logic circuits; Rabaey's book also covers bipolar circuits in detail. Geiger, Allen, and Strader [Gei90] give a good introduction to circuit simulation as well as a number of important topics in circuit and logic design. Shoji [Sho88] gives a very thorough analysis of delay through CMOS gates. Domino logic was introduced by Krambeck, Lee, and Law [Kra82]. De et al [De01] concentrate on leakage currents in CMOS logic.

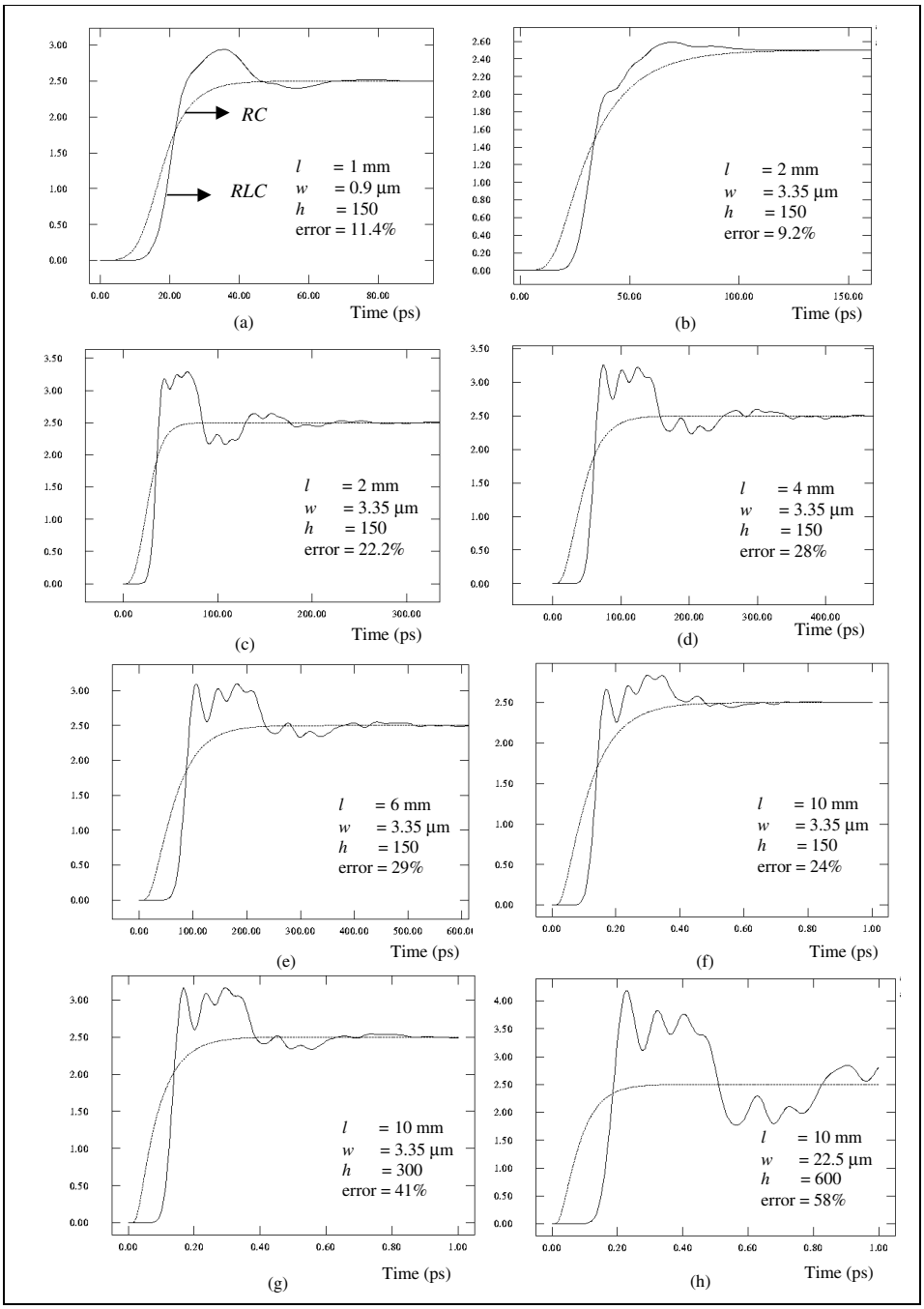


Figure 3-52: *CMOS gate driving a copper wire, using RC and RLC models (from Ismail and Friedman [Ism00]).*

3.10 Problems

Use the parameters for the 0.5 μ*m* process of Table 2-4 whenever process parameters are required, unless otherwise noted.

3-1. Design the static complementary pullup and pulldown networks for these logic expressions:

a) $(a + b + c)'$.

b) $[(a + b)c]'$.

c) $(a + b)(c + d)$.

3-2. Write the defining logic equation and transistor topology for each complex gate below:

a) AOI-22.

b) OAI-22.

c) AOI-212.

d) OAI-321.

e) AOI-2222.

3-3. Draw a layout for a two-input NOR gate using a static complementary circuit. The gate should have its inputs in metal on the cell's left hand side and its output in metal on the right hand side.

3-4. Size the transistors in a three-input, static complementary NOR gate such that the circuit's rise and fall times are approximately equal.

3-5. What is the difference in fall time of a two-input, static complementary NOR gate (assuming a minimum-size load capacitance) when one pulldown and when two pulldowns are activated?

3-6. Compute the low-to-high transition time and delay (at a power supply voltage of 3.3V) using the τ model through a two-input NAND gate which drives one input of a three-input NOR gate (both static complementary gates):

a) Compute the load capacitance on the NAND gate, assuming the NOR gate's transistors all have W=6λ, L=2λ.

b) Compute the equivalence resistance of appropriate transistors for the low-to-high transition, assuming the pulldown transistors have W=6λ, L=2λ and the pullups have W=6λ, L=2λ.

c) Compute the transition time and delay.

3-7. Compute, using the τ model, the high-to-low transition time and delay (at a power supply voltage of 3.3V) of a two-input, static complementary NOR gate with minimum-sized transistors driving these loads:

a) An inverter with minimum-sized pullup and pulldown.

b) An inverter whose pullup and pulldown are both of size W=10λ, L=10λ.

c) A 2000$\lambda \times 2\lambda$ poly wire connected to an inverter with minimum-sized pullup and pulldown (lump the wire and inverter parasitics into a single *RC* section).

d) Size the transistors in a two-input NOR gate such that the gate's rise time and fall time are approximately equal.

3-8. Redesign the layout of Figure 3-8 so that the inverter's output displays roughly symmetric rise and fall times.

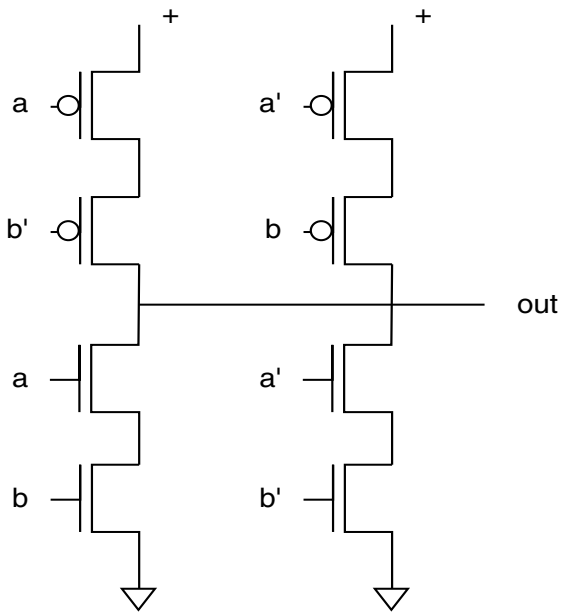
3-9. Design a three-input, static complementary NAND gate, which implements this function:

a	b	c	NAND(a,b,c)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

a) Draw a switch-level schematic.

b) Draw a stick diagram.

3-10. Here is a partial schematic for a two-input XOR gate:



The gate's output is 1 when exactly one of its inputs is 1. The schematic is partial because transistors in the diagram require both the true (a) and complement (a') form of the inputs, but the inverters which generate a' and b' from a and b are not shown.

a) Write the truth table for the two-input XOR.

b) Complete the schematic for this gate—compute the complement of the inputs using inverters.

c) Draw a stick diagram for the partial XOR schematic.

d) Draw a stick diagram for the complete XOR schematic.

3-11. Draw a stick diagram for a three-input NOR gate.

3-12. Draw the stick diagram for a one-bit multiplexer cell built from static, complementary gates. The stick diagram should show the detail of the component NAND gates—don't draw the NAND gates as cells.

3-13. A two-input NAND gate drives a set of parallel inverters; all the logic gates use minimum-sized transistors. Compare the NAND gate driving *n* inverters to the delay

through a two-stage network, in which two inverters each drive *n*/2 fanout inverters. What is the smallest value of *n* for which the buffered network has smaller delay than the unbuffered network?

3-14. At what length does a minimum-width poly wire present a capacitive load equal to a minimum-sized inverter? A minimum-width metal-1 wire?

3-15. Draw the circuit topology of a three-input NOR gate designed in pseudo-nMOS.

3-16. Design a two-input AND gate in domino logic:

a) Draw a transistor schematic.

b) Draw a stick diagram.

3-17. For a pseudo-nMOS inverter:

a) What V_{OL} must be chosen such that the pulldown transistor of a subsequent pseudo-nMOS gate will be off when the inverter's output is logic 1?

b) What is the ratio $W_p/L_p/W_n/L_n$ required to achieve this output voltage?

3-18. Draw a schematic for a 2-input NAND in MTCMOS logic.

3-19. Draw a schematic for these gates in DCVSL:

a) 2 input NAND;

b) 2 input NOR;

c) 3 input NAND.

3-20. Draw schematic for these gates in DCSL:

a) 2 input NAND;

b) 2 input NOR;

c) 3 input NAND;

3-21. Plot the Elmore delay for a metal 1 wire of size $2000\lambda \times 3\lambda$ using

a) 2 sections;

b) 4 sections;

c) 8 sections.

3-22. Plot the Elmore delay for a metal 2 wire of size $3000\lambda \times 4\lambda$ using

a) 2 sections;

b) 4 sections;

c) 8 sections.

3-23. Compute the optimal number of buffers and buffer sizes for these RC (non-inductive) wires when driven by a minimum-size inverter:

a) metal 1 $3000\lambda \times 3\lambda$.

b) metal 1 $5000\lambda \times 3\lambda$.

c) metal 2 $3000\lambda \times 4\lambda$.

d) metal 2 $4000\lambda \times 4\lambda$.