

## Chapter 4

# Calculating the Logical Effort of Gates

The simplicity of the theory of logical effort follows from assigning to each kind of logic gate a number—its logical effort—that describes its drive capability relative to that of a reference inverter. The logical effort is independent of the actual size of the logic gate, allowing one to postpone detailed calculations of transistor sizes until after the logical effort analysis is complete.

Each logic gate is characterized by two quantities: its logical effort and its parasitic delay. These parameters may be determined in three ways:

- Using a few process parameters, one can estimate logical effort and parasitic delay as described in this chapter. The results are sufficiently accurate for most design work.
- Using test circuit simulations, the logical effort and parasitic delay can be simulated for various logic gates. This technique is explained in Chapter 5.
- Using fabricated test structures, logical effort and parasitic delay can be physically measured.

Before turning to methods of calculating logical effort, we present a discussion of different definitions and interpretations of logical effort. While these are all equivalent, in some sense, each offers a different perspective to the design task and each leads to different intuitions.

---

<sup>0</sup>Copyright ©1998, Morgan Kaufmann Publishers, Inc. This material may not be copied or distributed without permission of the publisher.

## 4.1 Definitions of logical effort

Logical effort captures enough information about a logic gate's topology—the network of transistors that connect the gate's output to the power supply and to ground—to determine the delay of the logic gate. In this section, we give three equivalent concrete definitions of logical effort.

**Definition 4.1** *The logical effort of a logic gate is defined as the number of times worse it is at delivering output current than would be an inverter with identical input capacitance.*

Any topology required to perform logic makes a logic gate less able to deliver output current than an inverter with identical input capacitance. For one thing, a logic gate must have more transistors than an inverter, and so to maintain equal input capacitance, its transistors must be narrower on average and thus less able to conduct current than those of an inverter with identical input capacitance. If its topology requires transistors in parallel, a conservative estimate of its performance will assume that not all of them conduct at once, and therefore that they will not deliver as much current as could an inverter with identical input capacitance. If its topology requires transistors in series, it cannot possibly deliver as much current as could an inverter with identical input capacitance. Whatever the topology of a simple logic gate, its ability to deliver output current must be worse than an inverter with identical input capacitance. Logical effort is a measure of how much worse.

**Definition 4.2** *The logical effort of a logic gate is defined as the ratio of its input capacitance to that of an inverter that delivers equal output current.*

This alternative definition is useful for computing the logical effort of a particular topology. To compute the logical effort of a logic gate, pick transistor sizes for it that make it as good at delivering output current as a standard inverter, and then tally up the input capacitance of each input signal. The ratio of this input capacitance to that of the standard inverter is the logical effort of that input to the logic gate. The logical effort of a logic gate will depend slightly on the mobility ratio in the fabrication process used to build it. These calculations are shown in detail later in this chapter.

**Definition 4.3** *The logical effort of a logic gate is defined as the slope of the gate's delay vs. fanout curve divided by the slope of an inverter's delay vs. fanout curve.*

This alternative definition suggests an easy way to measure the logical effort of any particular logic gate by experiments with real or simulated circuits of various fanouts.

## 4.2 Grouping input signals

Because logical effort relates the input capacitance to the output drive current available, a natural question arises: for a logic gate with multiple inputs, how many of the input signals should we consider when computing logical effort? It is useful to define several kinds of logical effort, depending on how input signals are grouped. In each case, we define an *input group* to contain the input signals that are relevant to the computation of logical effort:

- Logical effort *per input*, in which logical effort measures the effectiveness of a single input in controlling output current. The input group is the single input in question. All of the discussion in preceding chapters uses logical effort per input.
- Logical effort of a *bundle*, a group of related inputs. For example, a multiplexer requires true and complement select signals; this pair might be grouped into a bundle. Because bundles of complementary pairs of signals occur frequently in CMOS circuits, we adopt a special notation:  $s^*$  stands for a bundle containing the true signal  $s$  and the complement signal  $\bar{s}$ . The input group of a bundle contains all the signals in the bundle.
- *Total* logical effort, the logical effort of all inputs taken together. The input group contains all the input signals of the logic gate.

Terminology and context determine which kind of logical effort applies. The adjective “total” is always used when total logical effort is meant, while the other two cases are distinguished by the signals associated with them in context. “The total logical effort of a 2-input NAND gate” is the logical effort of both inputs taken together, while “the logical effort of a 2-input NAND gate” is the logical effort per input of one of its two inputs.

The logical effort of an input group is defined analogously to the logical effort per input, shown in the previous section. The analog of Definition 4.2 is: the logical effort  $g_b$  of an input group  $b$  is just

$$g_b = \frac{C_b}{C_{inv}} = \frac{\sum_b C_i}{C_{inv}} \quad (4.1)$$

where  $C_b$  is the combined input capacitance of every signal in the input group  $b$ , and  $C_{inv}$  is the input capacitance of an inverter designed to have the same drive capabilities as the logic gate whose logical effort we are calculating.

A consequence of Equation 4.1 is that the logical efforts associated with input groups sum in a straightforward way. The total logical effort is the sum of the logical effort per input of every input to the logic gate. The logical effort of a bundle is the sum of the logical effort per input of every signal in the bundle. Thus a logic gate can be viewed as having a certain total logical effort that can be allocated to its inputs according to their contribution to the gate's input capacitance.

### 4.3 Calculating logical effort

Definition 4.2 provides a convenient method for calculating the logical effort of a logic gate. We have but to design a gate that has the same current drive characteristics as a reference inverter, calculate the input capacitances of each signal, and apply Equation 4.1 to obtain the logical effort.

Because we compute the logical effort as a ratio of capacitances, the units we use to measure capacitance may be arbitrary. This observation simplifies the calculations enormously. First, assume that all transistors are of minimum length, so that a transistor's size is completely captured by its width,  $w$ . The capacitance of the transistor's gate is proportional to  $w$  and its ability to produce output current, or conductance, is also proportional to  $w$ . In most CMOS processes, pullup transistors must be wider than pulldown transistors to have the same conductance.  $\mu = \mu_n/\mu_p$  is the ratio of PMOS to NMOS width in an inverter for equal conductance.  $\gamma$  is the actual ratio of PMOS to NMOS width in an inverter. For simplicity, we will often assume that  $\gamma = \mu = 2$ . Under this assumption, an inverter will have a pulldown transistor of width  $w$  and a pullup transistor of width  $2w$ , as shown in Figure 4.1a, so the total input capacitance can be said to be  $3w$ . In this chapter, we will also find general expressions for logical effort as a function of  $\gamma$ . In Chapter 9, we will consider the benefits of choosing  $\gamma \neq \mu$ .

Let us now design a 2-input NAND gate so that it has the same drive characteristics as an inverter with a pulldown of width 1 and a pullup of width 2. Figure 4.1b shows such a NAND gate. Because the two pulldown transistors of the NAND gate are in series, each must have twice the conductance of the inverter pulldown transistor so that the series connection has a conductance equal to that of the inverter pulldown transistor. Therefore, these transistors are twice as wide as the inverter pulldown transistor. This reasoning assumes that transistors in series

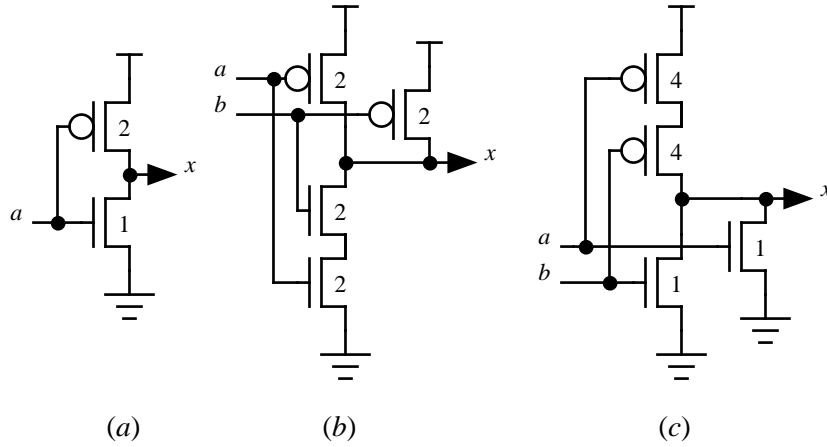


Figure 4.1: Simple gates. (a) The reference inverter. (b) A two-input NAND gate. (c) A two-input NOR gate.

obey Ohm's law for resistors in series. By contrast, each of the two pullup transistors in parallel need be only as large as the inverter pullup transistor to achieve the same drive as the reference inverter. Here we assume that if either input to the NAND gate is LOW, the output must be pulled HIGH, and so the output drive of the NAND gate must match that of the inverter even if only one of the two pullups is conducting.

We find the logical effort of the NAND gate in Figure 4.1b by extracting capacitances from the circuit schematic. The input capacitance of one input signal is the sum of the width of the pulldown transistor and the pullup transistor, or  $2 + 2 = 4$ . The input capacitance of the inverter with identical output drive is  $C_{inv} = 1 + 2 = 3$ . According to Equation 4.1, the logical effort per input of the 2-input NAND gate is therefore  $g = 4/3$ . Observe that both inputs of the NAND gate have identical logical efforts. Chapter 8 considers asymmetric gate designs favoring the logical effort of one input at the expense of another.

We designed the NOR gate in Figure 4.1c in a similar way. To obtain the same pulldown drive as the inverter, pulldown transistors one unit wide suffice. To obtain the same pullup drive, transistors four units wide are required, since two of them in series must be equivalent to one transistor two units wide in the inverter. Summing the input capacitance on one input, we find that the NOR gate has logical effort,  $g = 5/3$ . This is larger than the logical effort of the NAND gate because pullup transistors are less effective at generating output current than pulldown transistors. Were the two types of transistors similar, i.e.,  $\gamma = 1$ , both

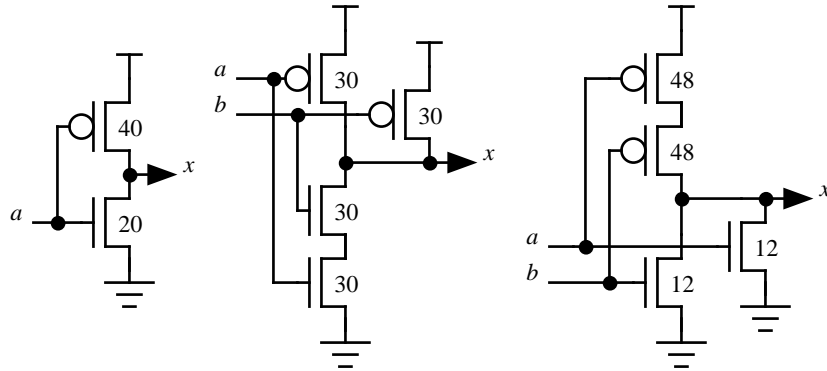


Figure 4.2: Simple gates with 60 input capacitance of 60 unit-sized transistors.

NAND and NOR gates would both have a logical effort of 1.5.

All of the sizing calculations in this monograph compute the input capacitance of gates. This capacitance is distributed among the transistors in the gate in the same proportions as are used when computing logical effort. For example, Figure 4.2 shows an inverter, NAND, and NOR gate, each with input capacitance equal to 60 unit-sized transistors.

When designing logic gates to produce the same output drive as the reference inverter, we are modeling CMOS transistors as pure resistors. If the transistor is off, the resistor has no conductance; if the transistor is on, it has a conductance proportional to its width. To determine the conductance of a transistor network, the conductances of the transistors are combined using the standard rules for calculating the conductance of a resistor network containing series and parallel resistor connections. While this model is only approximate, it characterizes logic gate performance well enough to design fast structures. More accurate values for logical effort can be obtained by simulating or measuring test circuits, as discussed in Chapter 5.

An important limitation of the model is that it does not account for velocity saturation. The velocity of carriers, and hence the current of a transistor, normally scales linearly with the electric field across the channel. When the field reaches a critical value, carrier velocity begins to saturate and no longer increases with field strength. The field across a single transistor is proportional to  $V_{DD}/L$ . In sub-micron processes,  $V_{DD}$  is usually scaled with  $L$  so that an NMOS transistor in an inverter is on the borderline of velocity saturation. PMOS transistors have lower mobility and thus are less prone to velocity saturation. Also, series NMOS transistors have a lower field across each transistor and therefore are less velocity

saturated. The effect of velocity saturation to remember is that series stacks of NMOS transistors in sub-micron processes tend to have less resistance than suggested by the model. Thus, structures with series NMOS transistors have slightly lower logical effort than our model predicts.

## 4.4 Asymmetric logic gates

Unlike the NAND and NOR gates, not all logic gates induce the same logical effort per input for all inputs. Equal logical effort per input is a consequence of the symmetries of the logic gates we have studied thus far. In this section, we will analyze an example in which the logical effort differs for different inputs.

Figure 4.3 shows one form of and-or-invert gate with an asymmetric configuration. The transistor widths in this gate have been chosen so that the output drive matches the reference inverter in Figure 4.1a: the pulldown structure is equivalent to a single pulldown transistor of width 1 and the pullup structure is equivalent to a single pullup transistor of width 2. The total logical effort of the gate, computed using Equation 4.1, is  $17/3$ .

The logical effort of the distinct inputs of the and-or-invert gate can be calculated individually. The logical effort per input for inputs  $a$  and  $b$  is  $6/3 = 2$ . The logical effort of the asymmetric input,  $c$ , is  $5/3$ . The  $c$  input has a slightly lower logical effort than the other inputs, reflecting the fact that the  $c$  input presents less capacitive load than the other inputs. Input  $c$  is “easier to drive” than the other two inputs.

Asymmetries in the logical effort of inputs arise in several different ways. The and-or-invert gate is topologically asymmetric, giving rise to unequal logical efforts of its inputs. Topologically symmetric gates, such as NAND and NOR, can be built with unequal transistor sizes to make them asymmetric so as to reduce the logical effort on some inputs, and thus reduce the logical effort along critical paths in a network. Other gates, such as XOR, have both asymmetric and symmetric forms, as discussed in Section 4.5.4. These techniques are explored further in Chapter 8.

## 4.5 Catalog of logic gates

The techniques for calculating logical effort are used in this section to develop Table 4.1. The expressions are slightly more general than those exhibited in earlier

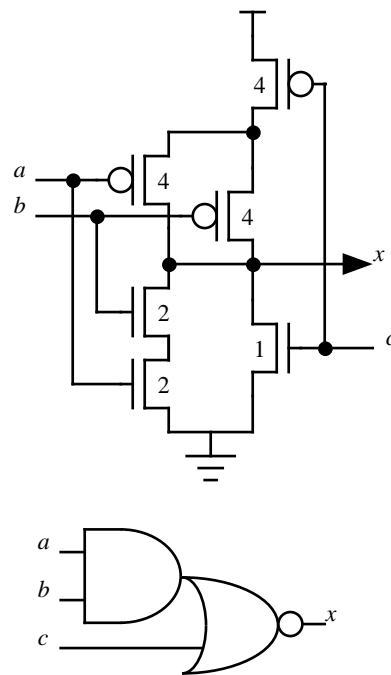


Figure 4.3: An asymmetric and-or-invert gate.



Gate type	Logical effort	Formula	$n = 2$ $\gamma = 2$	$n = 3$ $\gamma = 2$	$n = 4$ $\gamma = 2$
NAND	total per input	$\frac{n(n+\gamma)}{1+\gamma}$ $\frac{(n+\gamma)}{1+\gamma}$	8/3 4/3	5 5/3	8 2
NOR	total per input	$\frac{n(1+n\gamma)}{1+\gamma}$ $\frac{1+n\gamma}{1+\gamma}$	10/3 5/3	7 7/3	12 3
multiplexer	total $d, s^*$	$4n$ 2,2	8 2,2	12 2,2	16 2,2
XOR, XNOR, parity (symmetric)	total per bundle	$n^2 2^{n-1}$ $n 2^{n-1}$	8 4	36 12	128 32
XOR, XNOR, parity (asymmetric)	total per bundle		8 4,4	24 6,12,6	48 8,16,16,8
majority (symmetric)	total per input			12 4	
majority (asymmetric)	total per input			10 4,4,2	
C-element	total per input	$n^2$ $n$	4 2	9 3	16 4
latch (dynamic)	total $d, \phi^*$	4 2,2			
upper bounds	total per bundle	$\frac{\gamma n^2 2^n}{1+\gamma}$ $\frac{\gamma n 2^n}{1+\gamma}$	32/3 16/3	48 16	512/3 128/3

Table 4.1: Summary of calculations of the logical effort of logic gates.

sections in two ways. First, the expressions apply to logic gates with an arbitrary number of inputs,  $n$ . Second, they use a parameter for the ratio of  $p$ -type to  $n$ -type transistor widths, so as to permit calculation of logical effort for gates fabricated with various CMOS processes. Whereas the reference inverter in Figure 4.1a has a pullup-to-pulldown width ratio of 2 : 1, a ratio of  $\gamma$  : 1 is used throughout this section. Each logic gate will be designed to have a pulldown drive equivalent to an  $n$ -type transistor of width 1 and a pullup drive equivalent to a  $p$ -type transistor of width  $\gamma$ .

### 4.5.1 NAND gate

A NAND gate with  $n$  inputs, designed to have the same output drive as the reference inverter, will have a series connection of pulldown transistors, each of width  $n$ , and a parallel connection of pullup transistors, each of width  $\gamma$ . Using Equation 4.1, the total logical effort is:

$$g_{tot} = \frac{n(n + \gamma)}{1 + \gamma} \quad (4.2)$$

The logical effort per input is just  $1/n$  of this value, because the input capacitance is equally distributed among the  $n$  inputs.

Table 4.1 includes the expressions for logical effort and calculations for several common cases:  $\gamma = 2$ ,  $n = 2, 3, 4$ . Note from the equation that the logical effort changes only slightly for a wide range of  $\gamma$ : when  $\gamma$  ranges from 1 to 3, the total logical effort for  $n = 2$  ranges from 3 to 2.5.

### 4.5.2 NOR gate

The  $n$ -input NOR gate consists of a parallel connection of pulldown transistors, each of width 1, and a series connection of pullup transistors, each of width  $n\gamma$ . The total logical effort is therefore:

$$g_{tot} = \frac{n(1 + n\gamma)}{1 + \gamma} \quad (4.3)$$

Again, the logical effort per input is just  $1/n$  times this value. Table 4.1 includes examples of the logical effort of a NOR gate. For CMOS processes in which  $\gamma > 1$ , the logical effort of a NOR gate is greater than that of a NAND gate. If the CMOS fabrication process were perfectly symmetric, so that we could choose  $\gamma = 1$ , then the logical effort of NAND and NOR gates would be equal.

### 4.5.3 Multiplexers, tri-state inverters

An  $n$ -way inverting multiplexer is shown schematically in Figure 4.4. There are  $n$  data inputs,  $d_1 \dots d_n$ , and  $n$  bundles of complementary select signals,  $s*_1 \dots s*_n$ . Each data input is wired to a four-transistor *select arm*, which is in turn connected to the output  $c$ . To select input  $i$ , only bundle  $s*_i$  is driven TRUE, which enables current to flow through the pullup or pulldown structures in the select arm associated with  $d_i$ .

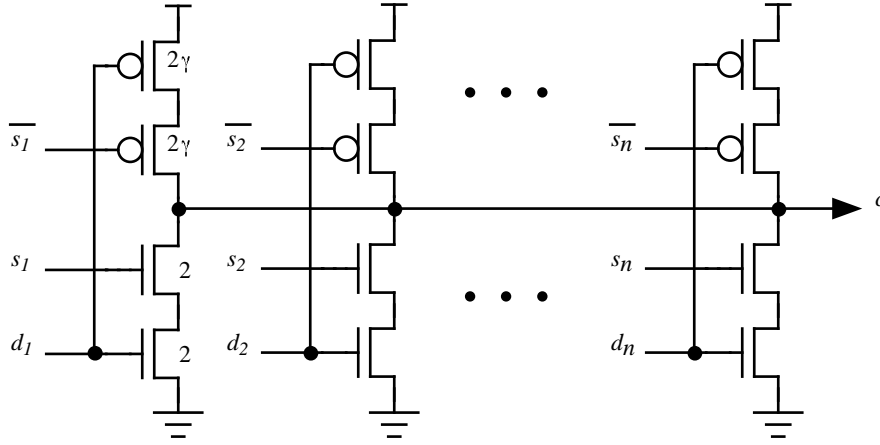


Figure 4.4: An  $n$ -way multiplexer. Each arm of the multiplexer has a data input  $d_i$  and a select bundle  $s*_i$ .

The total logical effort of a multiplexer is  $n(4 + 4\gamma)/(1 + \gamma) = 4n$ . The logical effort per data input is just  $(2 + 2\gamma)/(1 + \gamma) = 2$ , and the logical effort per select bundle is also 2. Note that the logical effort per input of a multiplexer does not depend on the number of inputs. Although this property suggests that large, fast, multiplexers could be built, stray capacitance in large multiplexers limits their growth. This problem is analyzed fully in Chapter 11. Also, increasing the number of multiplexer inputs tends to increase the logical effort of the select generation logic.

A single multiplexer arm is sometimes called a *tri-state inverter*. When a multiplexer is distributed across a bus, the individual arms are often drawn separately as tri-state inverters. Note that the logical efforts of the  $s$  and  $\bar{s}$  inputs may differ.

#### 4.5.4 XOR, XNOR, and parity gates

Figure 4.5 shows an XOR gate with two inputs,  $a*$  and  $b*$ , and output  $c$ . The gate has two bundled inputs; the  $a*$  bundle contains a complementary pair  $a$  and  $\bar{a}$ , and the  $b*$  bundle contains  $b$  and  $\bar{b}$ . The total logical effort of the gate is  $(8 + 8\gamma)/(1 + \gamma) = 8$ . The logical effort per input is just  $1/4$  this amount, or 2. The logical effort per input bundle is just the sum of the logical effort per input of the two inputs in the bundle, or 4.

The structure shown in Figure 4.5 can be generalized to compute the parity of  $n$  inputs. As an example, Figure 4.6a shows a 3-input XOR gate. The  $n$ -input gate

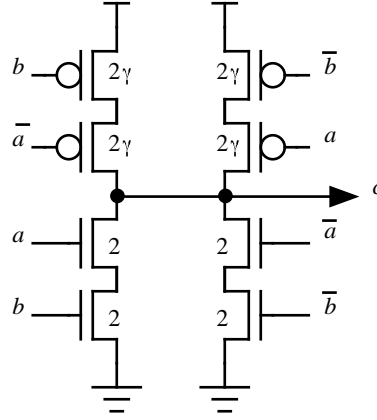


Figure 4.5: A two-input XOR gate, with input bundles  $a^*$  and  $b^*$ , and output  $c$ .

will have  $2^{n-1}$  pulldown chains, each with  $n$  transistors in series, each of width  $n$ . There will be  $2^{n-1}$  pullup chains, each with  $n$  transistors in series, each of width  $n\gamma$ . Thus the total logical effort will be  $2^{n-1}n(n + n\gamma)/(1 + \gamma) = n^2 2^{n-1}$ . The logical effort per input will be  $1/(2n)$  times this figure, or  $n2^{n-2}$ , and the logical effort per input bundle will be  $1/n$  times the total logical effort, or  $n2^{n-1}$ .

For  $n = 3$  and above, symmetric structures such as the one shown in Figure 4.6a fail to yield least logical effort. Figure 4.6b shows a way to share some of the transistors in separate pullup and pulldown chains to reduce the logical effort. Repeating the calculation, we see that the total logical effort is 24, which is a substantial reduction from 36, the total logical effort of the symmetric structure in Figure 4.6a. In the asymmetric version, bundles  $a^*$  and  $c^*$  have a logical effort per bundle of 6. Bundle  $b^*$  has a logical effort of 12, which is the same as in the symmetric version because no transistors connected to  $b$  or  $\bar{b}$  are shared in the asymmetric gate.

The XOR and parity gates can be altered slightly to produce an inverted output: simply interchange the  $a$  and  $\bar{a}$  connections. Note that this transformation does not change any of the logical effort calculations.

### 4.5.5 Majority gate

Figure 4.7 shows two designs for an inverting 3-input majority gate. Its output is LOW when two or more of its inputs are HIGH. The symmetric design is shown in Figure 4.7a. The total logical effort is  $(12 + 12\gamma)/(1 + \gamma) = 12$ , distributed evenly among the inputs. The logical effort per input is therefore 4. Figure 4.7b

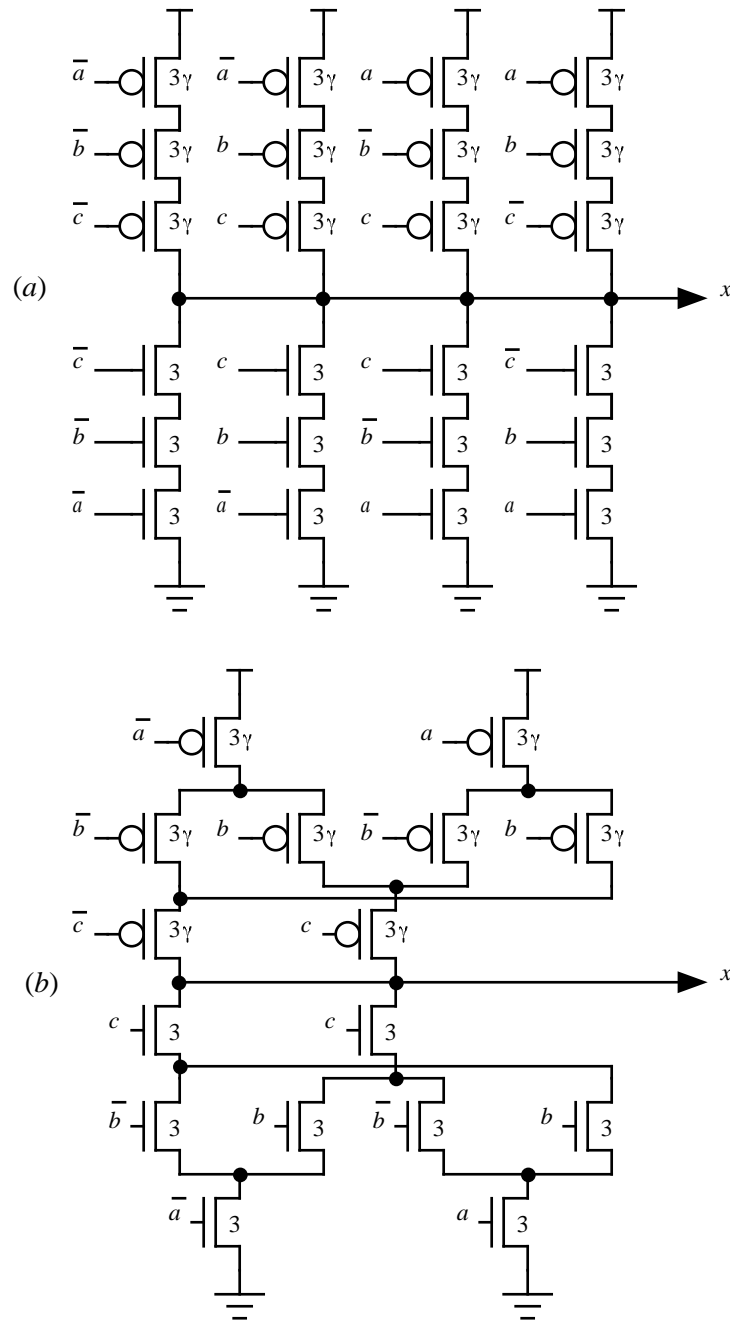


Figure 4.6: Two designs for three-input parity gates. (a) A symmetric design. (b) An asymmetric design with reduced logical effort.

shows an asymmetric design, which shares transistors as does the XOR design in Figure 4.6b. The total logical effort of this design is 10, and it is unevenly distributed among the inputs. The  $a$  input has a logical effort of 2, while the  $b$  and  $c$  inputs have logical efforts of 4 each.

#### 4.5.6 Adder carry chain

Figure 4.8 shows one stage of a ripple-carry chain in an adder. The stage accepts carry  $C_{in}$  and delivers a carry out in inverted form on  $\overline{C}_{out}$ . The inputs  $g$  and  $\overline{k}$  come from the two bits to be summed at this stage. The signal  $g$  is HIGH if this stage generates a new carry, forcing  $\overline{C}_{out} = 0$ . Similarly,  $\overline{k}$  is LOW if this stage kills incoming carries, forcing  $\overline{C}_{out} = 1$ .

The total logical effort of this gate is  $(5 + 5\gamma)/(1 + \gamma) = 5$ . The logical effort per input for  $C_{in}$  is 2; for the  $g$  input it is  $(1 + 2\gamma)/(1 + \gamma)$ ; and for the  $\overline{k}$  input it is  $(2 + \gamma)/(1 + \gamma)$ .

#### 4.5.7 Dynamic latch

Figure 4.9 shows a dynamic latch: when the clock signal  $\phi$  is HIGH, and its complement  $\overline{\phi}$  is LOW, the gate output  $q$  is set to the complement of the input  $d$ . The total logical effort of this gate is 4; the logical effort per input for  $d$  is 2, and the logical effort of the  $\phi*$  bundle is also 2. Altering the latch to make it statically stable increases its logical effort slightly (see Exercise 4-2).

#### 4.5.8 Dynamic Muller C-element

Figure 4.10 shows an inverting dynamic Muller C-element with two inputs. Although this gate is rarely seen in designs for synchronous systems, it is a staple of asynchronous system design. The behavior of the gate is as follows: When both inputs are HIGH, the output goes LOW; when both inputs go LOW, the output goes HIGH. In other conditions, the output retains its previous value—the C-element thus retains state. The total logical effort of this gate is 4, equally divided between the two inputs.

An  $n$ -input C-element can be formed in the obvious way, by making series pullup and pulldown chains of  $n$  transistors each. The width of a pulldown transistor is  $n$ , and of a pullup transistor is  $n\gamma$ . The total logical effort is thus  $n(n + n\gamma)/(1 + \gamma) = n^2$ , and the logical effort per input is just  $n$ .

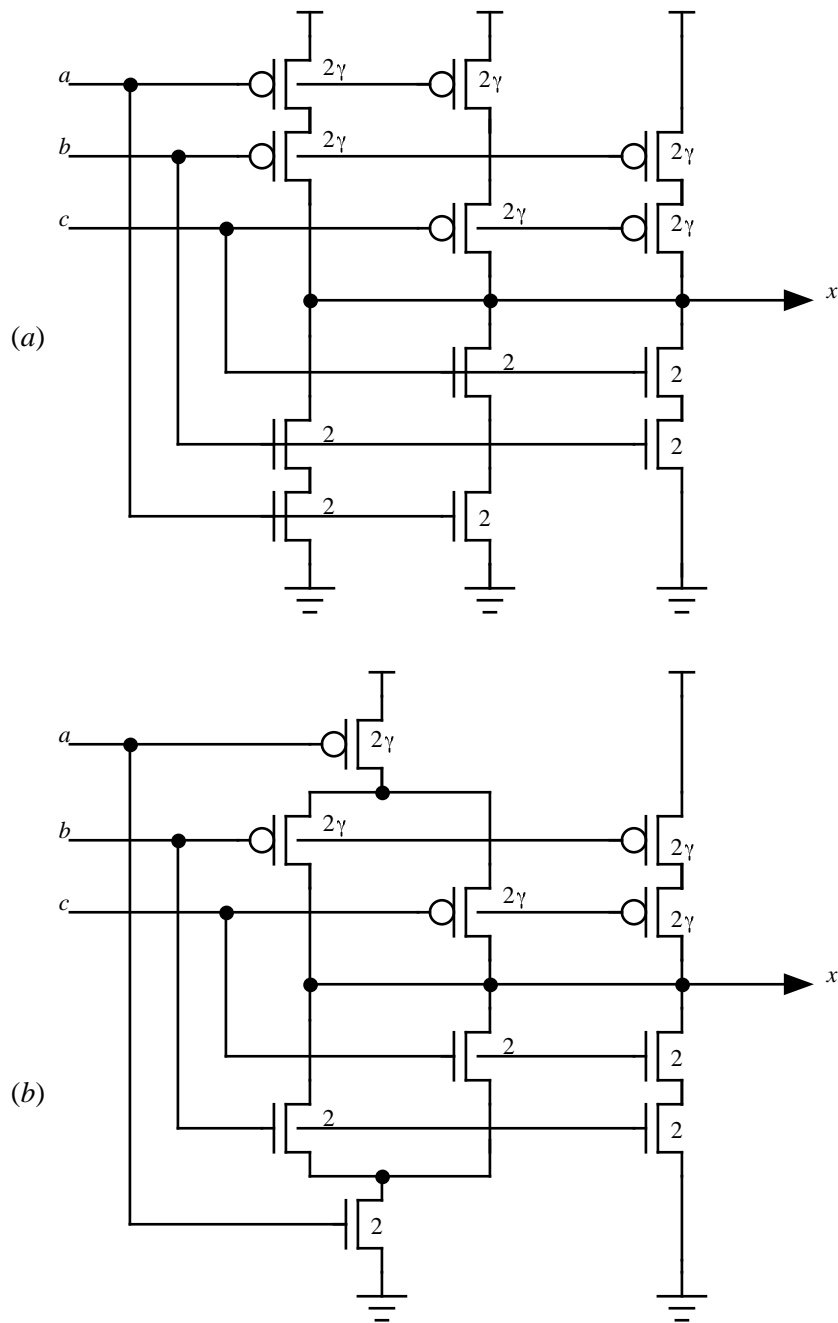


Figure 4.7: Two designs for three-input majority gates. (a) A symmetric design. (b) An asymmetric design with reduced logical effort.

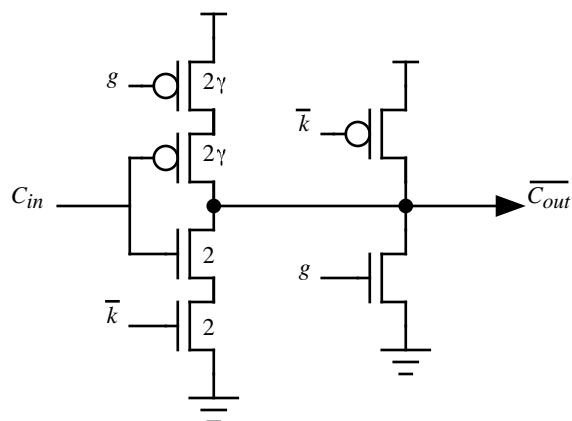


Figure 4.8: A carry-propagation gate. The carry arrives on  $C_{in}$  and leaves on  $\overline{C}_{out}$ . The  $g$  input is HIGH if a carry is generated at this stage, and the  $\overline{k}$  input is LOW if a carry is killed at this stage.

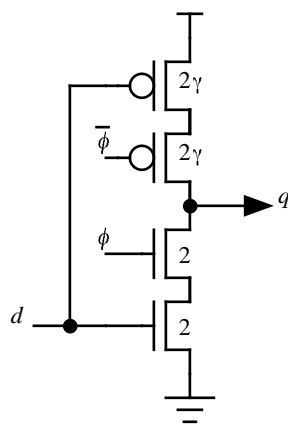


Figure 4.9: A dynamic latch with input  $d$  and output  $q$ . The clock bundle is  $\phi^*$ .



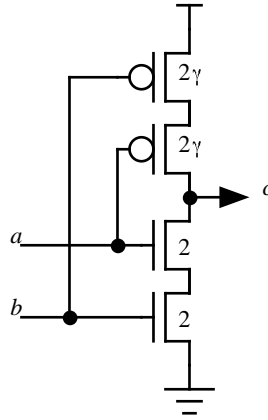


Figure 4.10: A two-input inverting dynamic Muller C-element. The inputs are  $a$  and  $b$ , and the output is  $c$ .

#### 4.5.9 Upper bounds on logical effort

It is easy to establish an upper bound for the logical effort of a gate with  $n$  inputs. For any truth table, construct a gate with  $2^n$  arms, each consisting of a series connection of  $n$  transistors, each of which receives the true or complement form of a different input. For entries in the truth table that require a LOW output, the series transistors in the corresponding arm are all  $n$ -type pulldown transistors and the series string bridges ground and the logic gate output. The transistor gates in the string receive inputs in such a way that the series connection conducts current when the input conditions for the truth table entry are met. For entries in the truth table that require a HIGH output, the series transistors are all  $p$ -type pullups and the series string spans the positive power supply and the logic gate output. The transistor gates receive the complement of the appropriate input. To design such a gate to have the same output drive as the reference inverter, each  $n$ -type transistor must have width  $n$ , and each  $p$ -type transistor must have width  $\gamma n$ . To compute the worst-case logical effort, assume that  $\gamma \geq 1$  and inputs are connected only to  $p$ -type transistors, which are larger than  $n$ -type transistors and so offer more load. Thus the worst-case input capacitance is  $\gamma n^2 2^n$ , and the worst-case logical effort is therefore  $\gamma n^2 2^n / (1 + \gamma)$ .

This result shows that in the worst case, the logical effort of a logic gate grows exponentially with the number of inputs. These bounds are not particularly tight, and may perhaps be improved. Any improvement will hinge on reducing the number of transistors in a gate by sharing.

## 4.6 Estimating parasitic delay

Calculating the parasitic delay of logic gates is not as easy as calculating logical effort. The principal contribution to the parasitic capacitance is the capacitance of the diffused regions of transistors connected to the output signal. The capacitance of these regions will depend on their layout geometry and on process parameters. However, a crude approximation can be obtained by imagining that a transistor of width  $w$  has a diffused region of capacitance equal to  $wC_d$  associated with its source and an identical region associated with its drain. The constant  $C_d$  is a property of the fabrication process and the inverter layout.

This model allows us to compute the parasitic delay of an inverter. The output signal is connected to two diffused regions: the one associated with the pulldown of width 1 will have capacitance  $C_d$ , and the one associated with the pullup of width  $\gamma$  will have capacitance  $\gamma C_d$ . The input capacitance of the inverter is likewise proportional to the transistor widths, but with a different constant of proportionality characteristic of transistor gate capacitance. Thus the input capacitance is  $(1 + \gamma)C_g$ . The parasitic delay is the ratio of the parasitic capacitance to the input capacitance of the inverter, which is just  $p_{inv} = C_d/C_g$ . The two constants of proportionality can be determined from layout geometry and process parameters (see Exercise 4-10). We shall adopt a nominal value of  $p_{inv} = 1.0$ , which is representative of inverter designs. This quantity can be measured from test circuits, as shown in Section 5.1.

We can estimate the parasitic delay of logic gates from the inverter parameters. The delay will be greater than that of an inverter by the ratio of the total width of diffused regions connected to the output signal to the corresponding width of an inverter, provided the logic gate is designed to have the same output drive as the inverter. Thus we have

$$p = \left( \frac{\sum w_d}{1 + \gamma} \right) p_{inv} \quad (4.4)$$

where  $w_d$  is the width of transistors connected to the logic gate's output. For this estimate to apply, we assume that transistor layouts in the logic gates are similar to those in the inverter. Note that this estimate ignores other stray capacitances in a logic gate, such as contributions from wiring and from diffused regions that lie between transistors that are connected in series.

This approximation can be applied to an  $n$ -input NAND gate, which has one pulldown transistor of width  $n$  and  $n$  pullup transistors of width  $\gamma$  connected to the output signal, so  $p = np_{inv}$ . An  $n$ -input NOR gate likewise has  $p = np_{inv}$ .

Gate type	Formula	Parasitic delay when $p_{inv} = 1.0$			
		$n = 1$	$n = 2$	$n = 3$	$n = 4$
inverter	$p_{inv}$	1			
NAND	$np_{inv}$		2	3	4
NOR	$np_{inv}$		2	3	4
multiplexer	$2np_{inv}$		4	6	8
XOR, XNOR, parity	$n2^{n-1}p_{inv}$		4	12	
majority	$6p_{inv}$			6	
C-element	$np_{inv}$		2	3	4
latch	$2p_{inv}$	2			

Table 4.2: Estimates of the parasitic delay of logic gates.

An  $n$ -way multiplexer has  $n$  pulldowns of width 2 and  $n$  pullups of width  $2\gamma$ , so  $p = 2np_{inv}$ . Table 4.2 summarizes some of these results.

The parasitic estimation has a serious limitation in that it predicts linear scaling of delay with number of inputs. In actuality, the parasitic delay of a series stack of transistors increases quadratically with stack height because of internal diffusion and gate-source capacitances. The Elmore delay model [9] handles distributed RC networks and shows that stacks of more than about four series transistors are best broken up into multiple stages of shorter stacks. Since parasitics are so geometry-dependent, the best way to find parasitic delay is to simulate circuits with extracted layout data.

## 4.7 Properties of logical effort

The calculation of logical effort for a logic gate is a straightforward process:

- Design the logic gate, picking transistor sizes that make it as good a driver of output current as the reference inverter.
- The logical effort per input for a particular input is the ratio of the capacitance of that input to the total input capacitance of the reference inverter.
- The total logical effort of the gate is the sum of the logical efforts of all of its inputs.

Table 4.1 reveals a number of interesting properties. The effect of circuit topology on logical effort is generally more pronounced than the effect of fabrication

technology. For CMOS with  $\gamma = 2$ , the total logical effort for 2-input NAND and NOR gates is nearly, but not quite, three. If CMOS were exactly symmetric ( $\gamma = 1$ ), the total logical effort for both NAND and NOR would be exactly three; the asymmetry of practical CMOS processes favors NAND gates over NOR gates.

In contrast to the weak dependence on  $\gamma$ , the logical effort of a gate depends strongly on the number of inputs. For example, the logical effort per input of an  $n$ -input NAND gate is  $(n + \gamma)/(1 + \gamma)$ , which clearly increases with  $n$ . When an additional input is added to a NAND gate, the logical effort of each of the existing inputs increases through no fault of its own. Thus the total logical effort of a logic gate includes a term that increases as the square of the number of inputs; and in the worst case, logical effort may increase exponentially with the number of inputs. When many inputs must be combined, this non-linear behavior forces the designer to choose carefully between single-stage logic gates with many inputs and multiple-stage trees of logic gates with fewer inputs per gate. Surprisingly, one logic gate escapes super-linear growth in logical effort—the multiplexer. This property makes it attractive for high fan-in selectors, which are analyzed in greater detail in Chapter 11.

The logical effort of gates covers a wide range. A two-input XOR gate has a total logical effort of 8, which is very large compared to the effort of NAND and NOR of about 3. The XOR circuit is also messy to lay out because the gates of its transistors interconnect with a criss-cross pattern. Are the large logical effort and the difficulty of layout related in some fundamental way? Whereas the output of most other logic functions changes only for certain transitions of the inputs, the XOR output changes for every input change. Is its large logical effort related in some way to this property?

The designs for logic gates we have shown in this chapter do not exhaust the possibilities. In Chapter 8, logic gates are designed with reduced logical effort for certain inputs that can lower the overall delay of a particular path through a network. In Chapter 9, we consider designs in which the rising and falling delays of logic gates differ, which saves space in CMOS and permits analysis of ratioed NMOS designs with the method of logical effort.

## 4.8 Exercises

**4-1** [20] Show that Equation 4.1 corresponds to the definition of logical effort given in Equation 3.8.

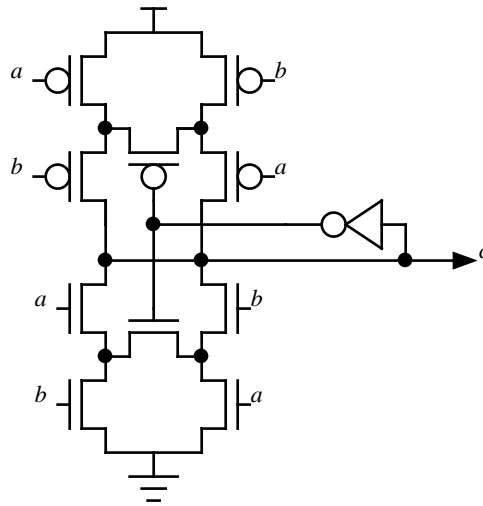


Figure 4.11: A static Muller C-element.

**4-2** [20] Modify the latch shown in Figure 4.9 so that its output is statically stable, even when the clock is LOW. How big should the transistors be? What is the logical effort of the new circuit?

**4-3** [20] In a fashion similar to Exercise 4-2, modify the dynamic C-element so that its output is static. How big should the transistors be? What is the logical effort of the new circuit?

**4-4** [20] Another way to construct a static C-element is shown in Figure 4.11. What relative transistor sizes should be used? What is the logical effort of the gate?

**4-5** [20] Figure 4.8 shows an adder element that inverts the polarity of the carry signal. A different design will be required for stages that accept a complemented carry input and generate a true carry output. Design such a circuit and calculate the logical effort of each input.

**4-6** [10] In many CMOS processes the ratio of pullup to pulldown conductance,  $\gamma$ , is greater than 2. How high does  $\gamma$  have to be before the logical effort of NOR is twice that of NAND?

**4-7** [20] The choice of transistor sizes for the inverter of Figure 4.1 was influ-

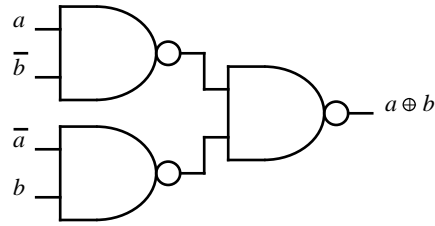


Figure 4.12: A two-stage XOR circuit.

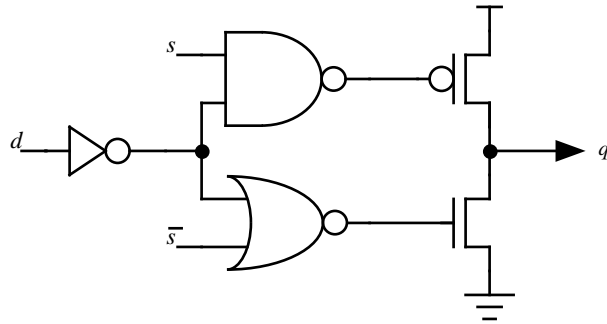


Figure 4.13: An inverting bus driver circuit, equivalent to the function of a tri-state inverter.

enced by the value of  $\gamma$ . Express the best pullup and pulldown transistor sizes in an inverter as a function of  $\gamma$  to obtain minimum delay in a two-inverter pair. Consider rising and falling delays separately.

**4-8** [20] Compare the logical effort of a two-stage XOR circuit such as shown in Figure 4.12 with that of the single stage XOR of Figure 4.4. Under what circumstances is each preferable?

**4-9** [20] Figure 4.13 shows a design for an inverting bus driver that achieves the same effect as a tri-state inverter. Compare the logical effort of the two circuits. Under what circumstances is each preferable?

**4-10** [25] Measure the gate and diffusion capacitances of your process. From these values, estimate  $p_{inv}$ .