Logical Effort

Designing Fast CMOS Circuits

Ivan Sutherland
Bob Sproull
David Harris

when the load is large, as in this example, the parasitic delay is insignificant compared to the effort delay. ■

1.3 Multistage Logic Networks

The method of logical effort reveals the best number of stages in a multistage network and how to obtain the least overall delay by balancing the delay among the stages. The notions of logical and electrical effort generalize easily from individual gates to multistage paths.

The logical effort along a path compounds by multiplying the logical efforts of all the logic gates along the path. We use the uppercase symbol G to denote the *path logical effort*, so that it is distinguished from g, the logical effort of a single gate in the path. The subscript i indexes the logic stages along the path.

$$G = \prod g_i \tag{1.7}$$

The electrical effort along a path through a network is simply the ratio of the capacitance that loads the last logic gate in the path to the input capacitance of the first gate in the path. We use an uppercase symbol H to indicate the electrical effort along a path.

$$H = \frac{C_{out}}{C_{in}} \tag{1.8}$$

In this case, C_{in} and C_{out} refer to the input and output capacitances of the path as a whole, as may be inferred from context.

We need to introduce a new kind of effort, named *branching effort*, to account for fanout within a network. So far we have treated fanout as a form of electrical effort: when a logic gate drives several loads, we sum their capacitances, as in Example 1.3, to obtain an electrical effort. Treating fanout as a form of electrical effort is easy when the fanout occurs at the final output of a network. This method is less suitable when the fanout occurs within a logic network because we know that the electrical effort for the network depends only on the ratio of its output capacitance to its input capacitance.

When fanout occurs within a logic network, some of the available drive current is directed along the path we are analyzing, and some is directed off that path. We define the branching effort b at the output of a logic gate to be

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}} = \frac{C_{total}}{C_{useful}} \tag{1.9}$$

where $C_{on-path}$ is the load capacitance along the path we are analyzing and $C_{off-path}$ is the capacitance of connections that lead off the path. Note that if the path does not branch, the branching effort is one. The branching effort along an entire path B is the product of the branching effort at each of the stages along the path.

$$B = \prod b_i \tag{1.10}$$

Armed with definitions of logical, electrical, and branching effort along a path, we can define the *path effort* F. Again, we use an uppercase symbol to distinguish the path effort from the stage effort f associated with a single logic stage. The equation that defines path effort is reminiscent of Equation 1.3, which defines the effort for a single logic gate:

$$F = GBH \tag{1.11}$$

Note that the path branching and electrical efforts are related to the electrical effort of each stage:

$$BH = \frac{C_{out}}{C_{in}} \prod b_i = \prod h_i \tag{1.12}$$

The designer knows C_{in}, C_{out}, and branching efforts b_i from the path specification. Sizing the path consists of choosing appropriate electrical efforts h_i for each stage to match the total BH product.

Although it is not a direct measure of delay along the path, the path effort holds the key to minimizing the delay. Observe that the path effort depends only on the circuit topology and loading and not upon the sizes of the transistors used in logic gates embedded within the network. Moreover, the effort is unchanged if inverters are added to or removed from the path, because the logical effort of an inverter is one. The path effort is related to the minimum achievable delay along the path, and permits us to calculate that delay easily. Only a little more work yields the best number of stages and the proper transistor sizes to realize the minimum delay.

The path delay D is the sum of the delays of each of the stages of logic in the path. As in the expression for delay in a single stage (Equation 1.5), we shall distinguish the *path effort delay* D_F and the *path parasitic delay* P:

$$D = \sum d_i = D_F + P \tag{1.13}$$

The path effort delay is simply

$$D_F = \sum g_i h_i \tag{1.14}$$

and the path parasitic delay is

$$P = \sum p_i \tag{1.15}$$

Optimizing the design of an N-stage logic network proceeds from a very simple principle that we will prove in Chapter 3: *The path delay is least when each stage in the path bears the same stage effort.* This minimum delay is achieved when the stage effort is

$$\hat{f} = g_i h_i = F^{1/N} \tag{1.16}$$

We use a hat over a symbol to indicate an expression that achieves minimum delay.

Combining these equations, we obtain the principal result of the method of logical effort, which is an expression for the minimum delay achievable along a path:

$$\hat{D} = NF^{1/N} + P \tag{1.17}$$

From a simple computation of its logical, branching, and electrical efforts we can obtain an estimate of the minimum delay of a logic network. Observe that when $N = 1$, this equation reduces to Equation 1.5.

To equalize the effort borne by each stage on a path, and therefore achieve the minimum delay along the path, we must choose appropriate transistor sizes for each stage of logic along the path. Equation 1.16 shows that each logic stage should be designed with electrical effort

$$\hat{h}_i = \frac{F^{1/N}}{g_i} \tag{1.18}$$

From this relationship, we can determine the transistor sizes of gates along a path. Start at the end of the path and work backward, applying the capacitance transformation:

$$C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}} \tag{1.19}$$

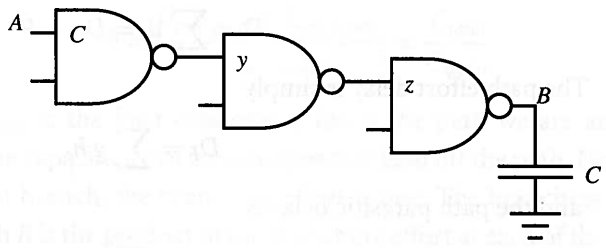

Figure 1.7 — A logic network consisting of three two-input NAND gates.

This determines the input capacitance of each gate, which can then be distributed appropriately among the transistors connected to the input. The mechanics of this process will become clear in the following examples.

Example 1.4 Consider the path from A to B involving three two-input NAND gates shown in Figure 1.7. The input capacitance of the first gate is C, and the load capacitance is also C. What is the least delay of this path, and how should the transistors be sized to achieve least delay? (The next example will use the same circuit with a different electrical effort.)

Solution To compute the path effort, we must compute the logical, branching, and electrical efforts along the path. The path logical effort is the product of the logical efforts of the three NAND gates, $G = g_0 g_1 g_2 = 4/3 \times 4/3 \times 4/3 = (4/3)^3 = 2.37$. The branching effort is $B = 1$, because all of the fanouts along the path are one, that is, there is no branching. The electrical effort is $H = C/C = 1$. Hence, the path effort is $F = GBH = 2.37$. Using Equation 1.17, we find the least delay achievable along the path to be $\hat{D} = 3(2.37)^{1/3} + 3(2p_{inv}) = 10.0$ delay units.

This minimum delay can be realized if the transistor sizes in each logic gate are chosen properly. First compute the stage effort $\hat{f} = 2.37^{1/3} = 4/3$. Starting with the output load C, apply the capacitance transformation of Equation 1.19 to compute input capacitance $z = C \times (4/3)/(4/3) = C$. Similarly, $y = z \times (4/3)/(4/3) = z = C$. Hence we find that all three NAND gates should have the same input capacitance, C. In other words, the transistor sizes in the three gates will be the same. This is not a surprising result: all stages have the same load and the same logical effort, and hence bear equal effort, which is the condition for minimizing path delay.

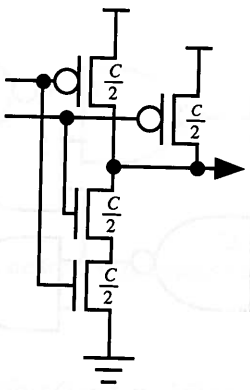

Figure 1.8—A schematic of a NAND gate from Example 1.4.

A schematic of the NAND gate is shown in Figure 1.8, assuming PMOS transistors have half the mobility of NMOS transistors. Selecting transistor sizes will be discussed further in Chapter 4. Since each input drives both a PMOS and NMOS transistor with capacitance $C/2$, the capacitance of each input is C, as desired. ■

EXAMPLE 1.5 Using the same network as in the previous example, Figure 1.7, find the least delay achievable along the path from A to B when the output capacitance is $8C$.

SOLUTION Using the result from Example 1.4 that $G = (4/3)^3$ and the new electrical effort $H = 8C/C = 8$, we compute $F = GBH = (4/3)^3 \times 8 = 18.96$, so the least path delay is $\hat{D} = 3(18.96)^{1/3} + 3(2p_{inv}) = 14.0$ delay units. Observe that although the electrical effort in this example is eight times the electrical effort in the earlier example, the delay is increased by only 40%.

Now let us compute the transistor sizes that achieve minimum delay. The stage effort $\hat{f} = 18.96^{1/3} = 8/3$. Starting with the output load $8C$, apply the capacitance transformation of Equation 1.19 to compute input capacitance $z = 8C \times (4/3)/(8/3) = 4C$. Similarly, $y = z \times (4/3)/(8/3) = z/2 = 2C$. To verify the calculation, calculate the capacitance of the first gate $y \times (4/3)/(8/3) = y/2 = C$, matching the design specification. Each successive logic gate has twice the input capacitance of its predecessor. This is achieved by making the transistors in a gate twice as wide as the corresponding

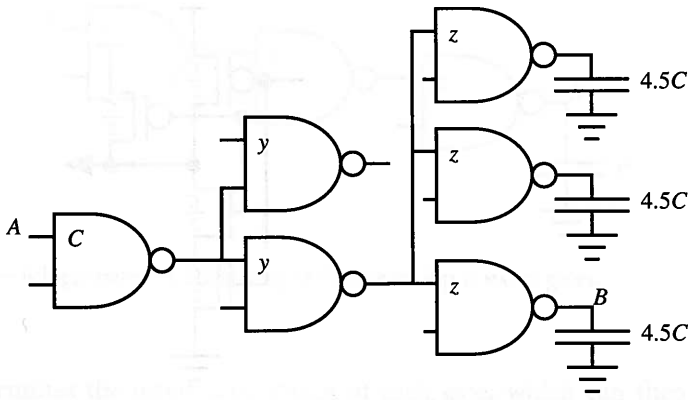

Figure 1.9—A multistage logic network with internal fanout.

transistors in its predecessor. The wider transistors in successive stages are better able to drive current into the larger loads. ■

Example 1.6 Optimize the circuit in Figure 1.9 to obtain the least delay along the path from A to B when the electrical effort of the path is 4.5.

Solution The path logical effort is $G = (4/3)^3$. The branching effort at the output of the first stage is $(y + y)/y = 2$, and at the output of the second stage it is $(z + z + z)/z = 3$. The path branching effort is therefore $B = 2 \times 3 = 6$. The electrical effort along the path is specified to be $H = 4.5$. Thus $F = GBH = 64$, and $\hat{D} = 3(64)^{1/3} + 3(2p_{inv}) = 18.0$ delay units.

To achieve this minimum delay, we must equalize the effort in each stage. Since the path effort is 64, the stage effort should be $(64)^{1/3} = 4$. Starting from the output, $z = 4.5C \times (4/3)/4 = 1.5C$. The second stage drives three copies of the third stage, so $y = 3z \times (4/3)/4 = z = 1.5C$. We can check the math by finding the size of the first stage $2y \times (4/3)/4 = (2/3)y = C$, as given in the design spec. ■

Example 1.7 Size the circuit in Figure 1.10 for minimum delay. Suppose the load is 20 microns of gate capacitance and that the inverter has 10 microns of gate capacitance.

Solution Assuming minimum-length transistors, gate capacitance is proportional to gate width. Hence, it is convenient to express capacitance in terms of microns of gate width, as given in this problem.

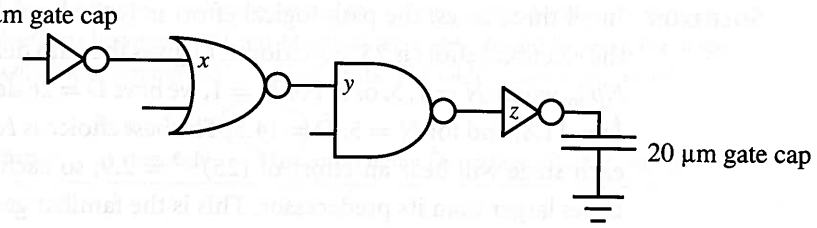

Figure 1.10—A multistage logic network with a variety of gates.

The path has logical effort $G = 1 \times (5/3) \times (4/3) \times 1 = 20/9$. The electrical effort is $H = 20/10 = 2$, and the branching effort is 1. Thus, $F = GBH = 40/9$, and $\hat{f} = (40/9)^{1/4} = 1.45$.

Start from the output and work backward to compute sizes: $z = 20 \times 1/1.45 = 14$, $y = 14 \times (4/3)/1.45 = 13$, and $x = 13 \times (5/3)/1.45 = 15$. These input gate widths are divided among the transistors in each gate. Notice that the inverters are assigned larger electrical efforts than the more complex gates because they are better at driving loads. Also note that these calculations do not have to be very precise. We will see in Section 3.6 that sizing a gate too large or too small by a factor of 1.5 still results in circuits within 5% of minimum delay. Therefore, it is easy to use "back-of-the-envelope" hand calculations to find gate sizes to one or two significant figures.

Note that the parasitic delay does not enter into the procedure for calculating transistor sizes to obtain minimum delay. Because the parasitic delay is fixed, independent of the size of the logic gate, adjustments to the size of logic gates cannot alter parasitic delay. In fact, we can ignore parasitic delay entirely unless we want to obtain an accurate estimate of the time required for a signal to propagate through a logic network, or if we are comparing two logic networks that contain different types of logic gates or different numbers of stages and therefore exhibit different parasitic delays. ■

Example 1.8 Consider three alternative circuits for driving a load 25 times the input capacitance of the circuit. The first design uses one inverter, the second uses three inverters in series, and the third uses five in series. All three designs compute the same logic function. Which is best, and what is the minimum delay?

Solution In all three cases, the path logical effort is 1, the branching effort is 1, and the electrical effort is 25. Equation 1.17 gives the path delay $D = N(25)^{1/N} + Np_{inv}$ where $N = 1, 3$, or 5. For $N = 1$, we have $\hat{D} = 26$ delay units; for $N = 3$, $\hat{D} = 11.8$; and for $N = 5$, $\hat{D} = 14.5$. The best choice is $N = 3$. In this design, each stage will bear an effort of $(25)^{1/3} = 2.9$, so each inverter will be 2.9 times larger than its predecessor. This is the familiar geometric progression of sizes that is found in many textbooks. ■

This example shows that the fastest speed obtainable depends on the number of stages in the circuit. Since the path delay varies markedly for different values of N, it is clear we need a method for choosing N to yield the least delay; this is the topic of the next section.

1.4 Choosing the Best Number of Stages

The delay equations of logical effort, such as Equation 1.17, can be solved to determine the number of stages, $\hat{N}$, that achieves the minimum delay. Although we will defer the solution technique until Chapter 3, Table 1.3 presents some results. The table shows, for example, that a single stage is fastest only if the path effort F is 5.83 or less. If the path effort lies between 5.83 and 22.3, a two-stage design is best. If it lies between 22.3 and 82.2, three stages are best. The table confirms that the right number of stages to use in Example 1.8, which has $F = 25$, is three. As the effort gets very large, the stage effort approaches 3.59.

If we use Table 1.3 to select the number of stages that gives the least delay, we may find that we must add stages to a network. We can always add an even number of stages by attaching pairs of inverters to the end of the path. Because we can't add an odd number of inverters without changing the logic function of the network, we may have to settle for a somewhat slower design or alter the logic network to accommodate an inverted signal. If a path uses a number of stages that is not quite optimal, the overall delay is usually not increased very much; what is disastrous is a design with half or twice the best number of stages.

The table is accurate only when we are considering increasing or decreasing the number of stages in a path by adding or removing inverters, because the table assumes that stages being added or removed have a parasitic delay equal to that of an inverter. Chapter 3 explains how other similar tables can be produced. When we are comparing logic networks that use different logic gate types or different

current because of different input gate voltages, which leads to variations in the delay of a logic gate due to different rise times of input signals. Long input rise times increase the delay of the logic gate because the pullup and pulldown networks are not switched fully on or off while the input voltage is near the switching threshold. If all rise times are equal, our simple model again holds because all logic gates will exhibit identical charging current waveforms and thus the same output voltage waveforms. Because the method of logical effort leads to nearly equal rise times by equalizing effort borne by all logic gates, we are justified in omitting rise time effects from Equation 3.8.

Further evidence to support the model is obtained from detailed circuit simulations, described in Section 5.1. Although the delay model is very simple, it is quite accurate when suitably calibrated. It is, indeed, the basis of models used by most static timing analyzers.

3.3 Minimizing Delay along a Path

The delay model for a single logic gate leads to a method for minimizing the delay in a sequence of logic gates connected in series. The key result is that path delay is minimized when the effort borne by each logic gate along the path is the same.

Consider the two-stage path in Figure 3.3. The path's input capacitance is C_1, the input capacitance of the first stage. Capacitance C_3 loads the second stage. According to Equation 3.8, the total delay, measured in units of τ, is

$$D = (g_1 h_1 + p_1) + (g_2 h_2 + p_2) \tag{3.13}$$

While the logical efforts g_1 and g_2 and parasitic delays p_1 and p_2 in this equation are fixed, the electrical efforts in each stage can be adjusted to minimize the

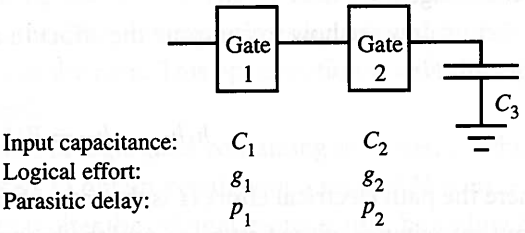

Figure 3.3 — Generic two-stage path.

delay. The electrical efforts are constrained, however, by the input capacitance C_1 and the load capacitance C_3, which are fixed:

$$h_1 = \frac{C_2}{C_1}$$

$$h_2 = \frac{C_3}{C_2}$$

and because the branching effort is 1:

$$h_1 h_2 = \frac{C_3}{C_1} = H$$

The path electrical effort H is a given constant that we cannot adjust. Substituting $h_2 = H/h_1$ into Equation 3.13, we obtain

$$D = (g_1 h_1 + p_1) + \left(\frac{g_2 H}{h_1} + p_2\right) \tag{3.14}$$

To minimize D, we take the partial derivative with respect to the only variable h_1, set the result equal to zero, and solve for h_1:

$$\frac{\partial D}{\partial h_1} = g_1 - \frac{g_2 H}{h_1^2} = 0 \tag{3.15}$$

$$g_1 h_1 = g_2 h_2 \tag{3.16}$$

Thus, delay is minimized when each stage bears the same effort, which is the product of the logical effort and the electrical effort. This result is independent of the scale of the circuits and of the parasitic delays. It does not say that the delays in the two stages will be equal—the delays will differ if the parasitic delays differ.

This result generalizes to paths with any number of stages (Exercise 3-3) and to paths that include branching effort. The fastest design always equalizes effort in each stage.

Let us now see how to compute the effort in each stage. We have for a path of length N:

$$h_1 h_2 \cdots h_N = BH \tag{3.17}$$

where the path electrical effort H is the ratio of the load on the last stage to the input capacitance of the first stage, and the branching effort B is the product of the branching efforts at each stage. Define the path logical effort to be

$$g_1 g_2 \cdots g_N = G \tag{3.18}$$

Multiplying these two equations together, we obtain the path effort F:

$$(g_1 h_1)(g_2 h_2) \cdots (g_N h_N) = GBH = F \tag{3.19}$$

To obtain minimum delay, the N factors on the left must be equal, so that each stage bears the same effort $\hat{f} = gh$. Thus the equation can be rewritten as:

$$\hat{f}^N = F \tag{3.20}$$

or

$$\hat{f} = F^{1/N} \tag{3.21}$$

Given G, B, H, and N for the path, we can compute F and therefore the stage effort $\hat{f}$ that achieves least delay. (Recall that our notation places a hat over a quantity chosen to achieve least path delay.) Now we can solve for the electrical effort h_i of each stage: $h_i = \hat{f}/g_i$. To calculate transistor sizes, we work backward or forward along the path, choosing transistor sizes to obtain the required electrical effort in each stage. This is the procedure outlined in Section 1.3.

The path delay obtained by this optimization procedure is

$$\hat{D} = \sum (g_i h_i + p_i) = NF^{1/N} + P \tag{3.22}$$

Although the parasitic delays do not affect the procedure for designing the path to obtain least delay, they do affect the actual delay obtained. We will see in the next section that parasitic delay also influences the best number of stages in a path.

3.4 Choosing the Length of a Path

Although equalizing the effort borne by each stage in a path minimizes delay for a given path, the delay can sometimes be reduced further by adjusting the number of stages in the path. This optimization is also a straightforward result of our delay model.

Consider a path of logic gates containing n_1 stages, to which we append n_2 additional inverters to obtain a path with a total of $N = n_1 + n_2$ stages. Let us assume that we may alter the original n_1 stages only by scaling because they perform necessary logic functions, but we may alter the number n_2 of inverters if