VLSI Design EE 523
Spring 2025

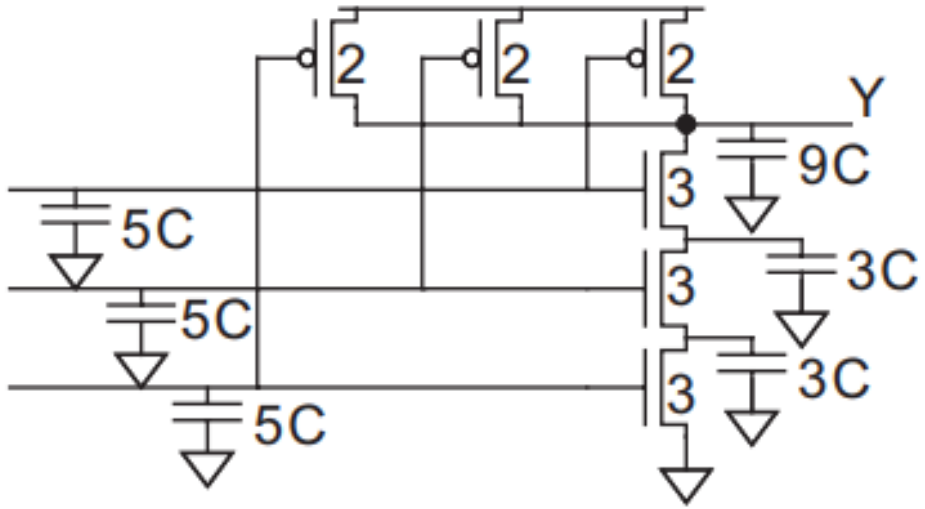
Shahid Masud

Lecture 12

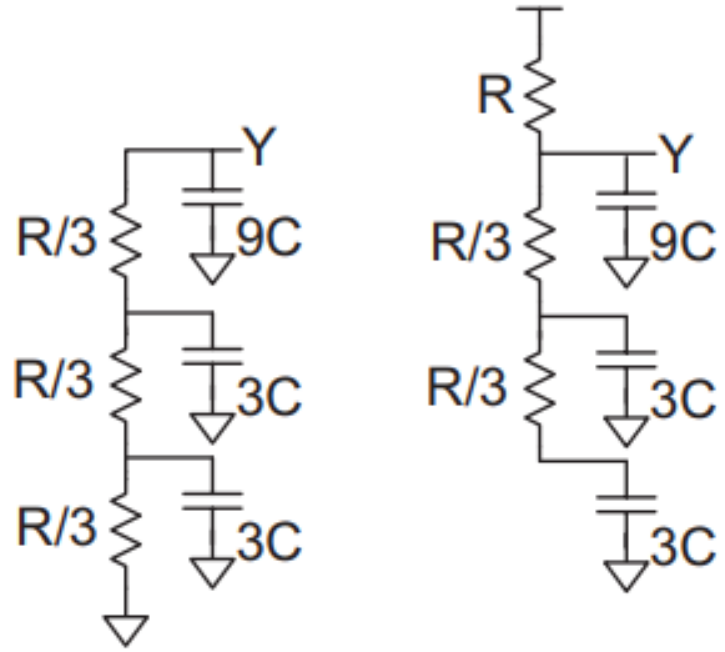
Topics for lecture 13

- Logic Effort of Unit Inverter
- Logic Effort of 2 input and 3 input NOR and Nand Gates
- Electrical Effort
- Delay calculations for no load and Fan Out of 4
- Delay calculations in a path with multiple gates
- Examine CMOS technology models to find R and C values

Lumped Capacitance to RC Model



(c)



(d) (e)

FIGURE 4.7 Equivalent circuits for a 3-input NAND gate

Real-world 74HC00

The 74HC00 quad two-input NAND gate comprises four identical circuits like the one shown in Figure 9.22. This circuit is double buffered by two inverters. Although the two inverters do not alter the overall logic function of the circuit, they do provide voltage gain (thus sharpening the voltage transfer characteristic) and current gain (thus allowing the use of smaller input transistors, with reduced input capacitance).

Table 9.3 summarizes the basic characteristics of 74HC high-speed CMOS gates. The range of supply voltages is much tighter than for the 4000 series.

TABLE 9.3

Basic Characteristics of High-Speed CMOS[a]

74HC series CMOS	
Gate material	Polysilicon
Gate length	3 µm
Oxide thickness	60 nm
Supply voltage	4.5 to 5.5 V
Propagation delay (C_L = 15 pF)	10 ns

[a] 74HCxx series.

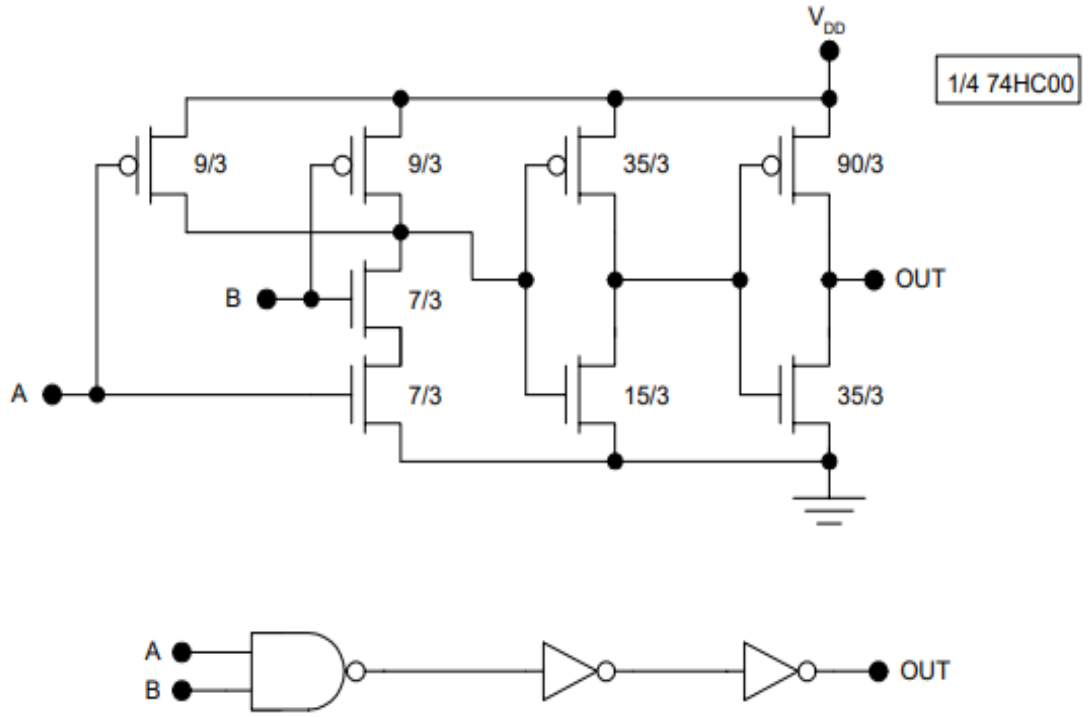


FIGURE 9.22
74HC high-speed CMOS NAND2 gate (1/4 of the 74HC00 quad two-input NAND gate).

Elmore Delay (no branches)

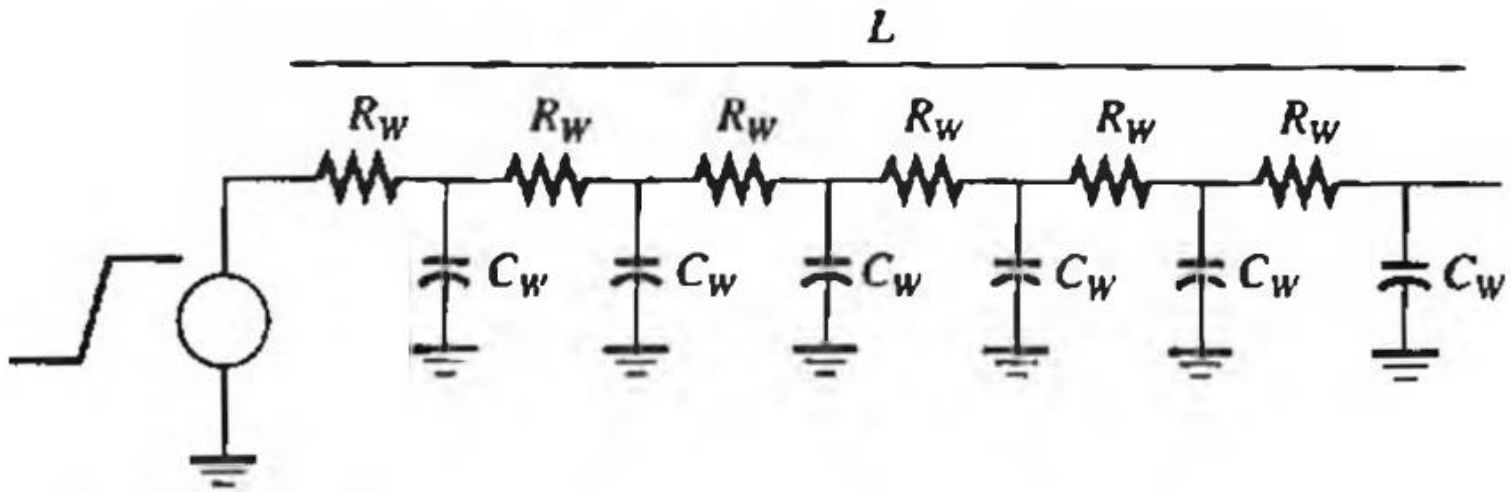


Figure 10.4
Distributed *RC* line as a lumped *RC* ladder.

For a general network, we can compute the Elmore delay as

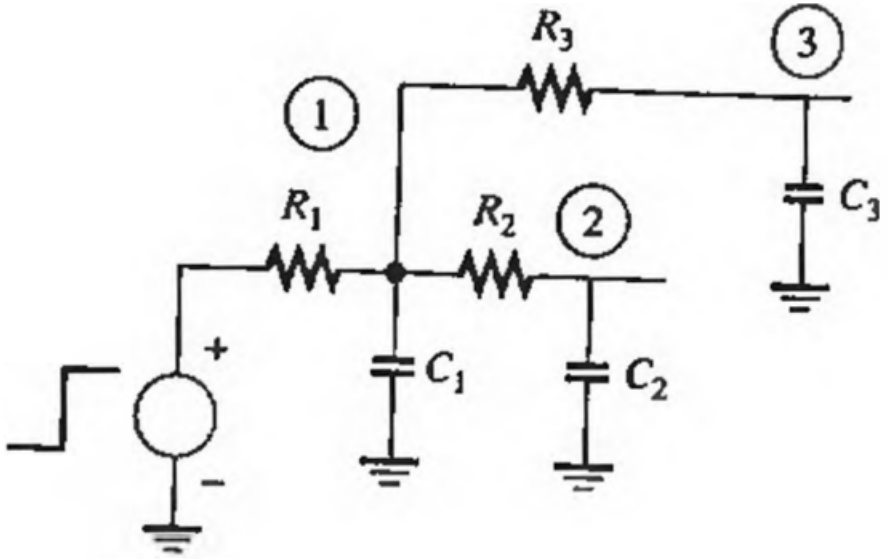
$$\tau_i = \sum_k (C_k \times R_{ik}) \qquad (10.6)$$

Elmore Delay with Branches

Elmore Delay of a Simple *RC* Tree

Problem:

Compute the Elmore delay from the input to nodes 1, 2, and 3 in the following *RC* tree. The Elmore delay is not accurate for internal nodes such as node 1. However, we are not usually interested in the delay from the input to an internal node in *RC* trees. In any case, write the expression for the time constant due to Elmore for each node.



Solution:

$$\tau_1 = R_1C_1 + R_1C_2 + R_1C_3$$

$$\tau_2 = C_1R_1 + C_3R_1 + C_2(R_1 + R_2)$$

$$\tau_3 = C_1R_1 + C_2R_1 + C_3(R_1 + R_3)$$

Elmore Delay example

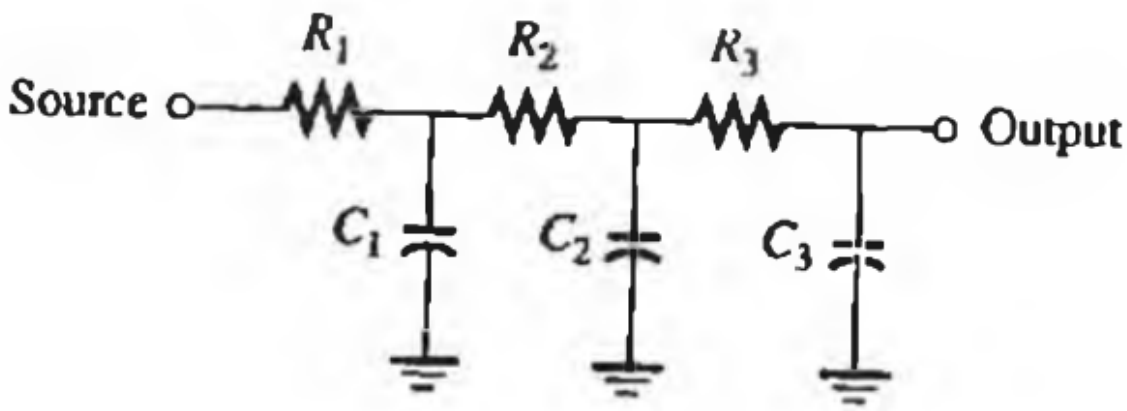


Figure 10.6
RC ladder for Elmore delay calculation.

To further illustrate this procedure, consider the circuit in Figure 10.6. It has an Elmore delay of

$$\tau = R_1C_1 + (R_1 + R_2)C_2 + (R_1 + R_2 + R_3)C_3$$

Elmore Delay on Inverter

Example 4.5

Repeat Example 4.4 if the driver is w times unit size.

SOLUTION: Figure 4.13 shows the equivalent circuit. The driver transistors are w times as wide, so the effective resistance decreases by a factor of w . The diffusion capacitance increases by a factor of w. The Elmore delay is $t_{pd} = ((3w + 3m)C)(R/w) = (3 + 3m/w)RC$.

Define the *fanout* of the gate, h, to be the ratio of the load capacitance to the input capacitance. (Diffusion capacitance is not counted in the fanout.) The load capacitance is $3mC$. The input capacitance is $3wC$. Thus, the inverter has a fanout of $h = m/w$ and the delay can be written as $(3 + 3h)RC$.

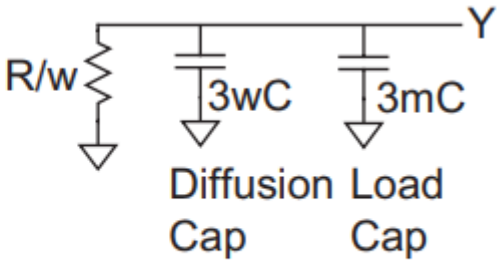


FIGURE 4.13 Equivalent circuit for wider inverter

FO4 Inverter Delay

Example 4.6

If a unit transistor has $R = 10\ k\Omega$ and $C = 0.1$ fF in a 65 nm process, compute the delay, in picoseconds, of the inverter in Figure 4.14 with a fanout of $h = 4$.

SOLUTION: The RC product in the 65 nm process is $(10\ k\Omega)(0.1\ fF) = 1$ ps. For $h = 4$, the delay is $(3 + 3h)(1\ ps) = 15$ ps. This is called the *fanout-of-4 (FO4) inverter delay* and is representative of gate delays in a typical circuit. Remember that a picosecond is a trillionth of a second. The inverter can switch about 66 billion times per second. This stunning speed partially explains the fantastic capabilities of integrated circuits.

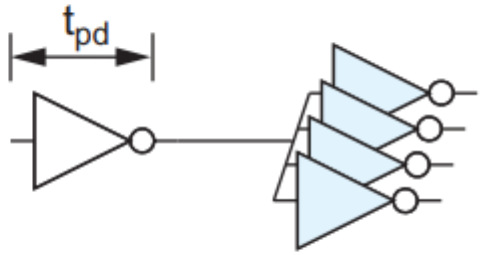


FIGURE 4.14 Fanout-of-4 (FO4) inverter

Elmore Delay in 3 Input NAND

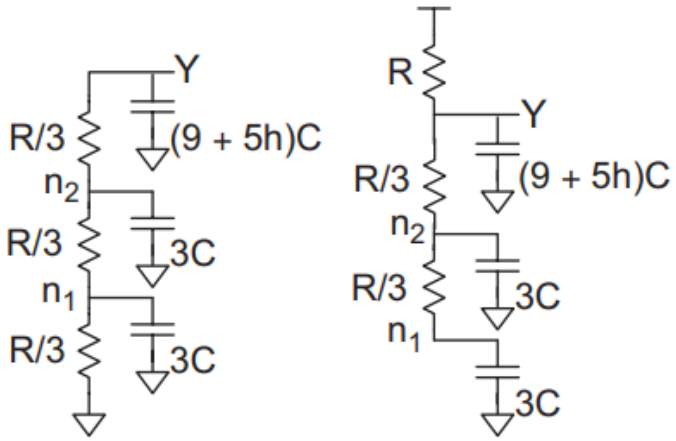


FIGURE 4.15 Equivalent circuits

Figure 4.15(b) shows the equivalent circuit for the falling transition. In the worst case, the two inner inputs are 1 and the outer input falls. Y is pulled up to V_{DD} through a single pMOS transistor. The ON nMOS transistors contribute parasitic capacitance that slows the transiton. Node Y has capacitance $(9 + 5h)C$ and resistance R to the V_{DD} supply. Node n_2 has capacitance $3C$. The relevant resistance is only R, not $(R + R/3)$, because the output is being charged only through R. This is what is meant by the resistance on the shared path from the source (V_{DD}) to the node (n_2) and the leaf (Y). Similarly, node n_1 has capacitance $3C$ and resistance R. Hence, the Elmore delay for the rising output is $t_{pdr} = (15 + 5h)RC$. The $R/3$ resistances do not contribute to this delay. Indeed, they shield the diffusion capacitances, which don't have to charge all the way up before Y rises. Hence, the Elmore delay is conservative and the actual delay is somewhat faster.

Although the gate has equal resistance pulling up and down, the delays are not quite equal because of the capacitances on the internal nodes.

Example 4.7

Estimate t_{pdf} and t_{pdr} for the 3-input NAND gate from Example 4.2 if the output is loaded with h identical NAND gates.

SOLUTION: Each NAND gate load presents 5 units of capacitance on a given input. Figure 4.15(a) shows the equivalent circuit including the load for the falling transition. Node n_1 has capacitance $3C$ and resistance of $R/3$ to ground. Node n_2 has capacitance $3C$ and resistance $(R/3 + R/3)$ to ground. Node Y has capacitance $(9 + 5h)C$ and resistance $(R/3 + R/3 + R/3)$ to ground. The Elmore delay for the falling output is the sum of these RC products, $t_{pdf} = (3C)(R/3) + (3C)(R/3 + R/3) + ((9 + 5h)C)(R/3 + R/3 + R/3) = (12 + 5h)RC$.

Calculate Contamination Delay through Elmore

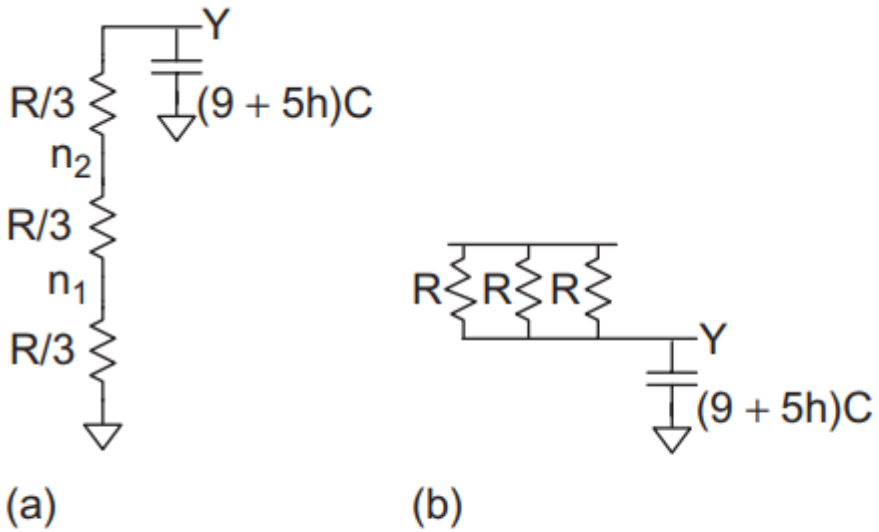


FIGURE 4.16 Equivalent circuits for contamination delay

Example 4.8

Estimate the contamination delays t_{cdf} and t_{cdr} for the 3-input NAND gate from Example 4.2 if the output is loaded with h identical NAND gates.

SOLUTION: The contamination delay is the fastest that the gate might switch. For the falling transition, the best case is that the bottom two nMOS transistors are already ON when the top one turns ON. In such a case, the diffusion capacitances on n_1 and n_2 have already been discharged and do not contribute to the delay. Figure 4.16(a) shows the equivalent circuit and the delay is $t_{cdf} = (9 + 5h)RC$.

For the rising transition, the best case is that all three pMOS transistors turn on simultaneously. The nMOS transistors turn OFF, so n_1 and n_2 are not connected to the output and do not contribute to delay. The parallel transistors deliver three times as much current, as shown in Figure 4.16(b), so the delay is $t_{cdr} = (3 + (5/3)h)RC$.

Parasitic Delay, Effort Delay and Logical Effort

In all of the Examples, the delay consists of two components. The *parasitic delay* is the time for a gate to drive its own internal diffusion capacitance. Boosting the width of the transistors decreases the resistance but increases the capacitance so the parasitic delay is ideally independent of the gate size.[3] The *effort delay* depends on the ratio h of external load capacitance to input capacitance and thus changes with transistor widths. It also depends on the complexity of the gate. The capacitance ratio is called the *fanout* or *electrical effort* and the term indicating gate complexity is called the *logical effort*. For example, an inverter has a delay of $d = h + 1$, so the parasitic delay is 1 and the logical effort is also 1. The NAND3 has a worst case delay of $d = (5/3)h + 5$. Thus, it has a parasitic delay of 5 and a logical effort of 5/3. These delay components will be explored further in Section 4.4.

Linear Delay Model of CMOS Gates

4.4 Linear Delay Model

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

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

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

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

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

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

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

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

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

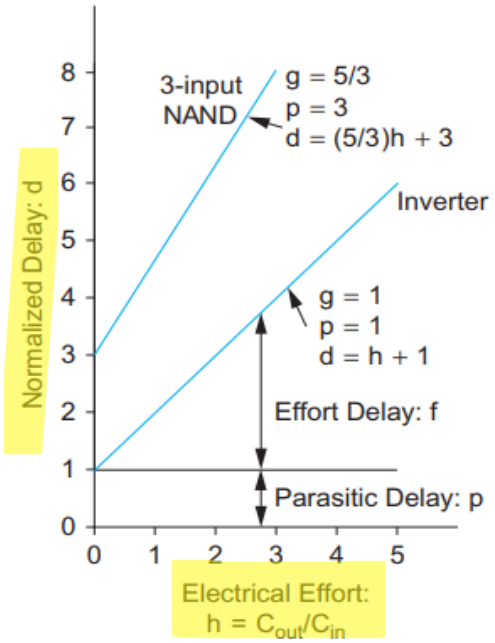
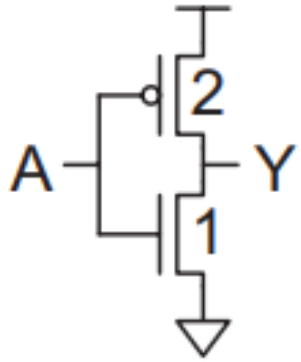


FIGURE 4.21 Normalized delay vs. fanout

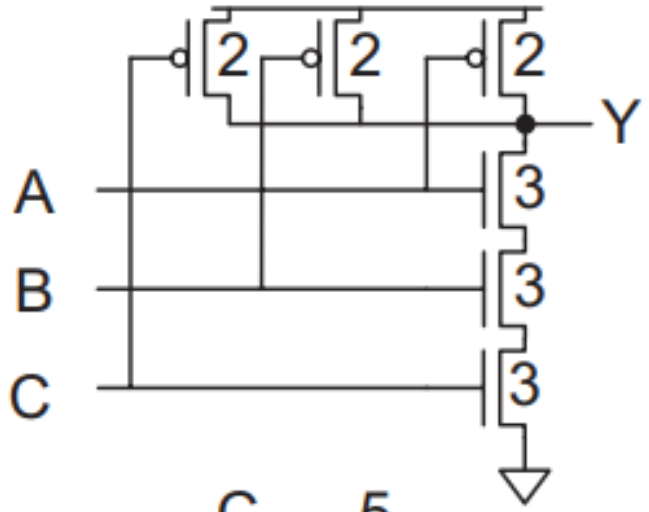
Logical Effert - Definition

4.4.1 Logical Effort

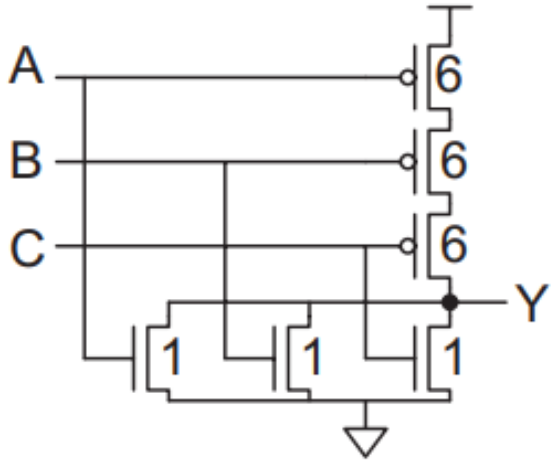
Logical effort of a gate is defined as *the ratio of the input capacitance of the gate to the input capacitance of an inverter that can deliver the same output current*. Equivalently, logical effort indicates how much worse a gate is at producing output current as compared to an inverter, given that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



(a) $C_{in} = 3$
$g = 3/3$



(b) $C_{in} = 5$
$g = 5/3$



(c) $C_{in} = 7$
$g = 7/3$

Logical Effort of Common Gates

TABLE 4.2 Logical effort of common gates

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

Parasitic Delay

4.4.2 Parasitic Delay

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

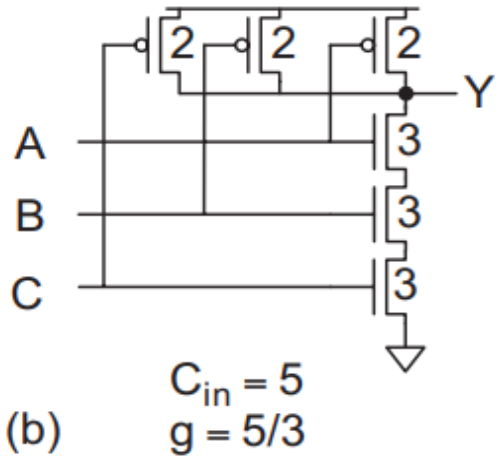


TABLE 4.3 Parasitic delay of common gates

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

Delay in Multistage Logic Network

4.5.1 Delay in Multistage Logic Networks

Figure 4.29 shows the logical and electrical efforts of each stage in a multistage path as a function of the sizes of each stage. The path of interest (the only path in this case) is marked with the dashed blue line. Observe that logical effort is independent of size, while electrical effort depends on sizes. This section develops some metrics for the path as a whole that are independent of sizing decisions.

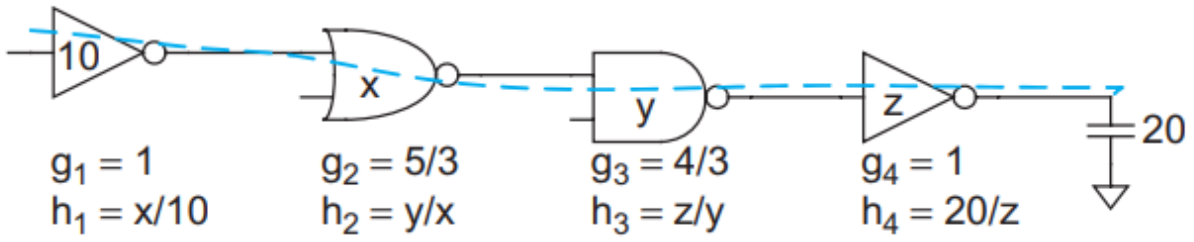


FIGURE 4.29 Multistage logic network

Path LE, Path EE, Path Effort

The *path logical effort* G can be expressed as the products of the logical efforts of each stage along the path.

$$G = \prod g_i \tag{4.32}$$

The *path electrical effort* H can be given as the ratio of the output capacitance the path must drive divided by the input capacitance presented by the path. This is more convenient than defining path electrical effort as the product of stage electrical efforts because we do not know the individual stage electrical efforts until gate sizes are selected.

$$H = \frac{C_{\text{out(path)}}}{C_{\text{in(path)}}} \tag{4.33}$$

The *path effort* F is the product of the stage efforts of each stage. Recall that the stage effort of a single stage is $f = gh$. Can we by analogy state $F = GH$ for a path?

$$F = \prod f_i = \prod g_i h_i \tag{4.34}$$

Logic Circuit with Two-way Branch

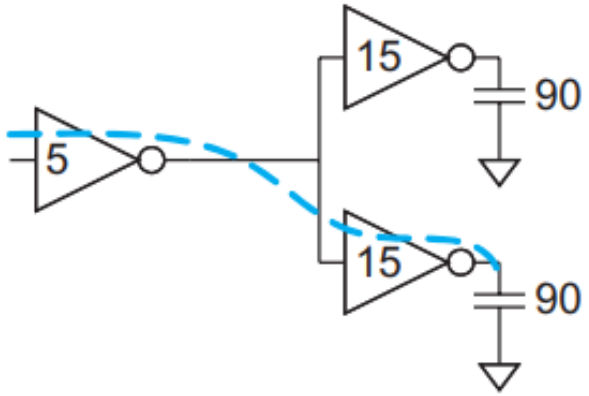


FIGURE 4.30 Circuit with two-way branch

In paths that branch, $F \neq GH$. This is illustrated in Figure 4.30, a circuit with a two-way branch. Consider a path from the primary input to one of the outputs. The path logical effort is $G = 1 \times 1 = 1$. The path electrical effort is $H = 90/5 = 18$. Thus, $GH = 18$. But $F = f_1 f_2 = g_1 h_1 g_2 h_2 = 1 \times 6 \times 1 \times 6 = 36$. In other words, $F = 2GH$ in this path on account of the two-way branch.

Path Delay Parameters

We must introduce a new kind of effort to account for branching between stages of a path. This *branching effort b* is the ratio of the total capacitance seen by a stage to the capacitance on the path; in Figure 4.30 it is (15 + 15)/15 = 2.

$$b = \frac{C_{\text{onpath}} + C_{\text{offpath}}}{C_{\text{onpath}}} \tag{4.35}$$

The *path branching effort B* is the product of the branching efforts between stages.

$$B = \prod b_i \tag{4.36}$$

Now we can define the path effort *F* as the product of the logical, electrical, and branching efforts of the path. Note that the product of the electrical efforts of the stages is actually *BH*, not just *H*.

$$F = GBH \tag{4.37}$$

We can now compute the delay of a multistage network. The *path delay D* is the sum of the delays of each stage. It can also be written as the sum of the *path effort delay* D_F and *path parasitic delay P*:

$$D = \sum d_i = D_F + P$$
$$D_F = \sum f_i \tag{4.38}$$
$$P = \sum p_i$$

The product of the stage efforts is F, independent of gate sizes. The path effort delay is the sum of the stage efforts. The sum of a set of numbers whose product is constant is minimized by choosing all the numbers to be equal. In other words, the path delay is minimized when each stage bears the same effort. If a path has N stages and each bears the same effort, that effort must be

$$\hat{f} = g_i h_i = F^{1/N} \tag{4.39}$$

Thus, the minimum possible delay of an N-stage path with path effort F and path parasitic delay P is

$$D = NF^{1/N} + P \tag{4.40}$$

This is a key result of Logical Effort. It shows that the minimum delay of the path can be estimated knowing only the number of stages, path effort, and parasitic delays without the need to assign transistor sizes. This is superior to simulation, in which delay depends on sizes and you never achieve certainty that the sizes selected are those that offer minimum delay.

Gate Sizes for Least Delay

It is also straightforward to select gate sizes to achieve this least delay. Combining EQs (4.21) and (4.22) gives us the *capacitance transformation* formula to find the best input capacitance for a gate given the output capacitance it drives.

$$C_{\text{in}_i} = \frac{C_{\text{out}_i} \times g_i}{\hat{f}} \tag{4.41}$$

Starting with the load at the end of the path, work backward applying the capacitance transformation to determine the size of each stage. Check the arithmetic by verifying that the size of the initial stage matches the specification.

Example 4.13

Estimate the minimum delay of the path from *A* to *B* in Figure 4.31 and choose transistor sizes to achieve this delay. The initial NAND2 gate may present a load of 8 λ of transistor width on the input and the output load is equivalent to 45 λ of transistor width.

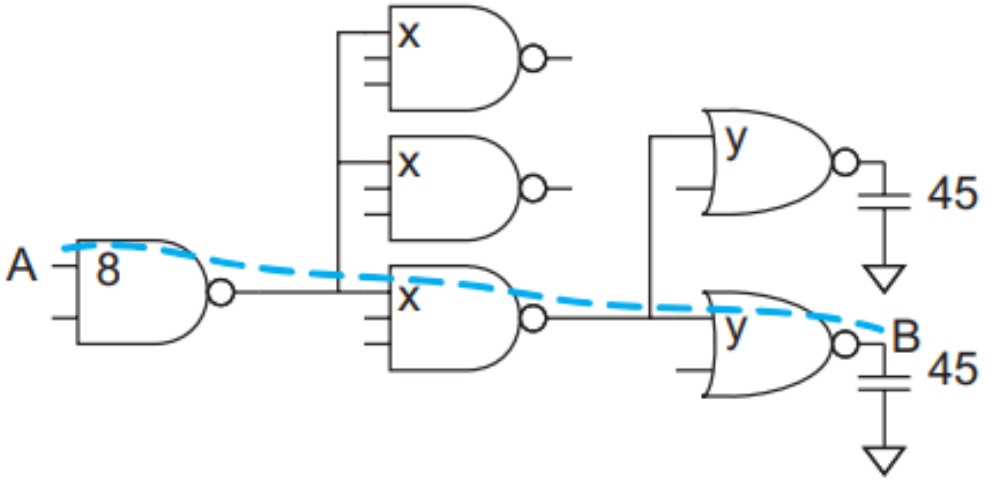


FIGURE 4.31 Example path

Solution

SOLUTION: The path logical effort is $G = (4/3) \times (5/3) \times (5/3) = 100/27$. The path electrical effort is $H = 45/8$. The path branching effort is $B = 3 \times 2 = 6$. The path effort is $F = GBH = 125$. As there are three stages, the best stage effort is $\hat{f} = \sqrt[3]{125} = 5$. The path parasitic delay is $P = 2 + 3 + 2 = 7$. Hence, the minimum path delay is $D = 3 \times 5 + 7 = 22$ in units of τ, or 4.4 FO4 inverter delays. The gate sizes are computed with the capacitance transformation from EQ (4.41) working backward along the path: $y = 45 \times (5/3)/5 = 15$. $x = (15 + 15) \times (5/3)/5 = 10$. We verify that the initial 2-input NAND gate has the specified size of $(10 + 10 + 10) \times (4/3)/5 = 8$. The transistor sizes in Figure 4.32 are chosen to give the desired amount of input capacitance while achieving equal rise and fall delays. For example, a 2-input NOR gate should have a 4:1 P/N ratio. If the total input capacitance is 15, the pMOS width must be 12 and the nMOS width must be 3 to achieve that ratio.

We can also check that our delay was achieved. The NAND2 gate delay is $d_1 = g_1h_1 + p_1 = (4/3) \times (10 + 10 + 10)/8 + 2 = 7$. The NAND3 gate delay is $d_2 = g_2h_2 + p_2 = (5/3) \times (15 + 15)/10 + 3 = 8$. The NOR2 gate delay is $d_3 = g_3h_3 + p_3 = (5/3) \times 45/15 + 2 = 7$. Hence, the path delay is 22, as predicted.

Recall that delay is expressed in units of τ. In a 65 nm process with $\tau = 3$ ps, the delay is 66 ps. Alternatively, a fanout-of-4 inverter delay is 5τ, so the path delay is 4.4 FO4s.

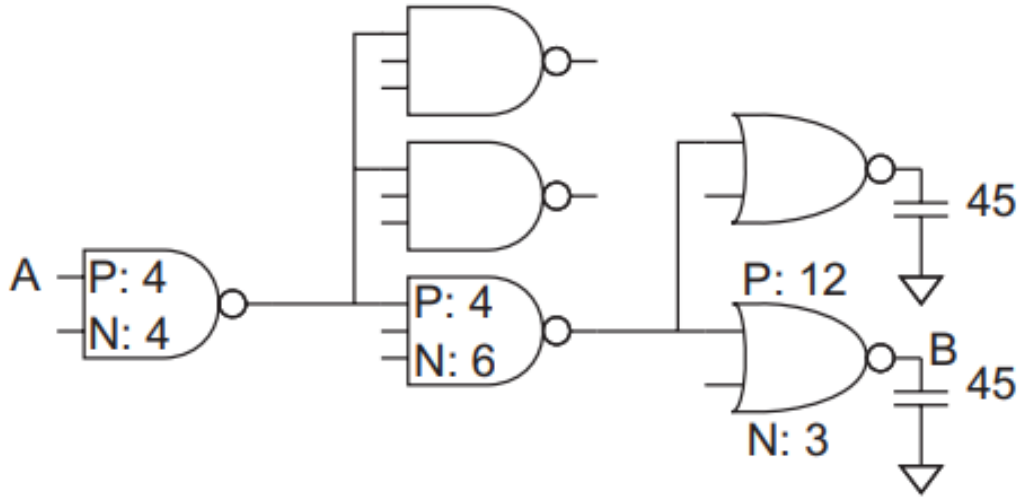


FIGURE 4.32 Example path annotated with transistor sizes

Optimum Number of Stages

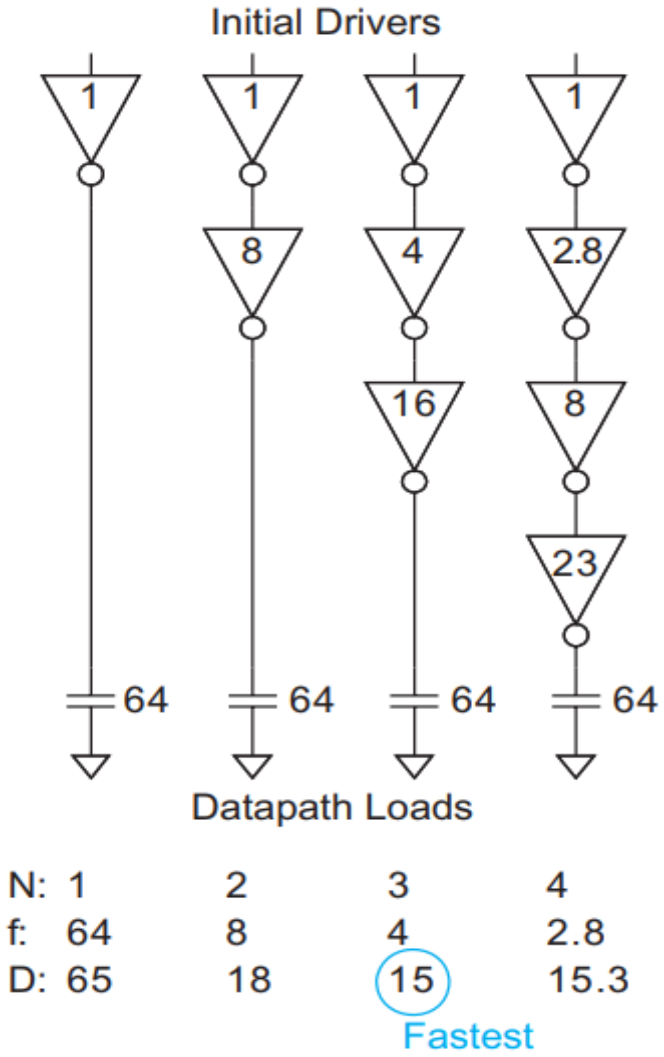


FIGURE 4.33 Comparison of different number of stages of buffers

Example 4.14

A control unit generates a signal from a unit-sized inverter. The signal must drive unit-sized loads in each bitslice of a 64-bit datapath. The designer can add inverters to buffer the signal to drive the large load. Assuming polarity of the signal does not matter, what is the best number of inverters to add and what delay can be achieved?

SOLUTION: Figure 4.33 shows the cases of adding 0, 1, 2, or 3 inverters. The path electrical effort is $H = 64$. The path logical effort is $G = 1$, independent of the number of inverters. Thus, the path effort is $F = 64$. The inverter sizes are chosen to achieve equal stage effort. The total delay is $D = N\sqrt[N]{64} + N$.

The 3-stage design is fastest and far superior to a single stage. If an even number of inversions were required, the two- or four-stage designs are promising. The four-stage design is slightly faster, but the two-stage design requires significantly less area and power.

Least Delay Path

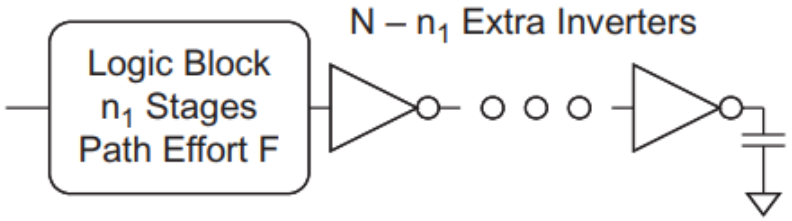


FIGURE 4.34 Logic block with additional inverters

In general, you can always add inverters to the end of a path without changing its function (save possibly for polarity). Let us compute how many should be added for least delay. The logic block shown in Figure 4.34 has n_1 stages and a path effort of F. Consider adding $N - n_1$ inverters to the end to bring the path to N stages. The extra inverters do not change the path logical effort but do add parasitic delay. The delay of the new path is

$$D = NF^{1/N} + \sum_{i=1}^{n_1} p_i + \left(N - n_1\right) p_{\text{inv}} \tag{4.42}$$

Differentiating with respect to N and setting to 0 allows us to solve for the best number of stages, which we will call $\hat{N}$. The result can be expressed more compactly by defining

$$\rho = F^{1/\hat{N}}$$

to be the best stage effort.

$$\frac{\partial D}{\partial N} = -F^{1/N} \ln F^{1/N} + F^{1/N} + p_{\text{inv}} = 0$$
$$\Rightarrow p_{\text{inv}} + \rho\left(1 - \ln \rho\right) = 0 \tag{4.43}$$

EQ (4.43) has no closed form solution. Neglecting parasitics (i.e., assuming $p_{\text{inv}} = 0$), we find the classic result that the stage effort $\rho = 2.71828$ (e) [Mead80]. In practice, the parasitic delays mean each inverter is somewhat more costly to add. As a result, it is better to use fewer stages, or equivalently a higher stage effort than e. Solving numerically, when $p_{\text{inv}} = 1$, we find $\rho = 3.59$.

A path achieves least delay by using $\hat{N} = \log_\rho F$ stages. It is important to understand

Summary of Logical Effort with Branches

4.5.4 Summary and Observations

Logical Effort provides an easy way to compare and select circuit topologies, choose the best number of stages for a path, and estimate path delay. The notation takes some time to become natural, but this author has poured through all the letters in the English and Greek alphabets without finding better notation. It may help to remember *d* for "**d**elay," *p* for "**p**arasitic," *b* for "**b**ranching," *f* for "e**f**fort," *g* for "lo**g**ical effort" (or perhaps **g**ain), and *h* as the next letter after "f" and "g." The notation is summarized in Table 4.5 for both stages and paths.

The method of Logical Effort is applied with the following steps:

1. Compute the path effort: $F = GBH$
2. Estimate the best number of stages: $\hat{N} = \log_4 F$
3. Sketch a path using: $\hat{N}$ stages
4. Estimate the minimum delay: $D = \hat{N}F^{1/\hat{N}} + P$
5. Determine the best stage effort: $\hat{f} = F^{1/\hat{N}}$
6. Starting at the end, work backward to find sizes: $C_{in_i} = \frac{C_{out_i} \times g_i}{\hat{f}}$

Table of Logical Effort Formulae

TABLE 4.5 Summary of Logical Effort notation

Term	Stage Expression	Path Expression
number of stages	1	N
logical effort	g (see Table 4.2)	$G = \prod g_i$
electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out(path)}}{C_{in(path)}}$
branching effort	$b = \frac{C_{onpath} + C_{offpath}}{C_{onpath}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
effort delay	f	$D_F = \sum f_i$
parasitic delay	p (see Table 4.3)	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

Reading and Book

- Textbook can be downloaded online “CMOS VLSI Design by Weste and Harris, 4th Edition”
- Readings from Chapter 4