

# Chapter 10

## Circuit Families

So far, we have applied logical effort primarily to analyze static CMOS circuits. High-performance integrated circuits often use other circuit families to achieve better speed at the expense of power consumption, noise margins, or design effort. This chapter computes the logical effort of gates in different circuit families and shows how to optimize such circuits. We begin by examining pseudo-NMOS logic and the closely related symmetric NOR gates. Then we delve into the design of domino circuits. Finally, we analyze transmission gate circuits by combining the transmission gates and driver into a single complex gate.

The method of logical effort does not apply to arbitrary transistor networks, but only to *logic gates*. A logic gate has one or more inputs and one output, subject to the following restrictions:

- The gate of each transistor is connected to an input, a power supply, or the output; and
- Inputs are connected only to transistor gates.

The first condition rules out multiple logic gates masquerading as one, and the second keeps inputs from being connected to transistor sources or drains, as in transmission gates without explicit drivers.

---

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

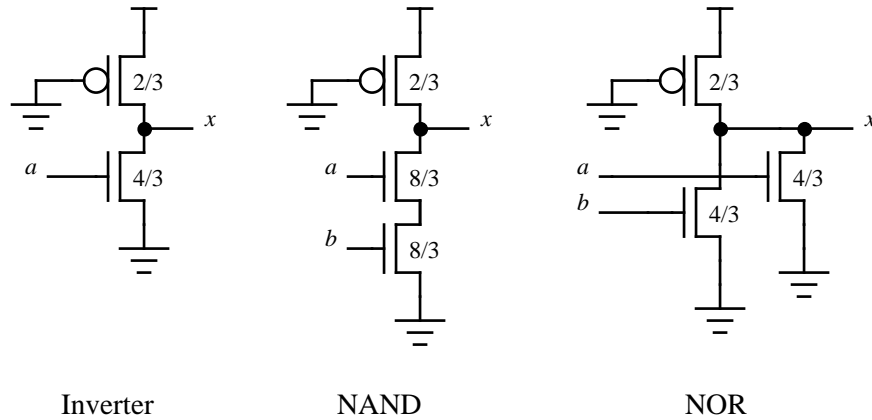


Figure 10.1: Pseudo-NMOS inverter, NAND and NOR gates, assuming  $\mu = 2$ .

## 10.1 Pseudo-NMOS circuits

Static CMOS gates are slowed because an input must drive both NMOS and PMOS transistors. In any transition, either the pullup or pulldown network is activated, meaning the input capacitance of the inactive network loads the input. Moreover, PMOS transistors have poor mobility and must be sized larger to achieve comparable rising and falling delays, further increasing input capacitance. Pseudo-NMOS and dynamic gates offer improved speed by removing the PMOS transistors from loading the input. This section analyzes pseudo-NMOS gates, while section 10.2 explores dynamic logic.

Pseudo-NMOS gates resemble static gates, but replace the slow PMOS pullup stack with a single grounded PMOS transistor which acts as a pullup resistor. The effective pullup resistance should be large enough that the NMOS transistors can pull the output to near ground, yet low enough to rapidly pull the output high. Figure 10.1 shows several pseudo-NMOS gates ratioed such that the pulldown transistors are about four times as strong as the pullup.

The analysis presented in Section 9.1 applies to pseudo-NMOS designs. The logical effort follows from considering the output current and input capacitance compared to the reference inverter from Figure 4.1. Sized as shown, the PMOS transistors produce  $1/3$  of the current of the reference inverter and the NMOS transistor stacks produce  $4/3$  of the current of the reference inverter. For falling transitions, the output current is the pulldown current minus the pullup current which is fighting the pulldown,  $4/3 - 1/3 = 1$ . For rising transitions, the output current

Gate type	Logical Effort $g$		
	Rising	Falling	Average
2-NAND	8/3	8/9	16/9
3-NAND	4	4/3	8/3
4-NAND	16/3	16/9	32/9
$n$ -NOR	4/3	4/9	8/9
$n$ -mux	8/3	8/9	16/9

Table 10.1: Logical efforts of pseudo-NMOS gates.

is just the pullup current,  $1/3$ .

The inverter and NOR gate have an input capacitance of  $4/3$ . The falling logical effort is the input capacitance divided by that of an inverter with the same output current, or  $g_d = (4/3)/3 = 4/9$ . The rising logical effort is three times greater,  $g_u = 4/3$ , because the current produced on a rising transition is only one third that of a falling transition. The average logical effort is  $g = (4/9 + 4/3)/2 = 8/9$ . This is independent of the number of inputs, explaining why pseudo-NMOS is a way to build fast wide NOR gates. Table 10.1 shows the rising, falling, and average logical efforts of other pseudo-NMOS gates, assuming  $\mu = 2$  and a 4:1 pulldown to pullup strength ratio. Comparing this with Table 4.1 shows that pseudo-NMOS multiplexers are slightly better than CMOS multiplexers and that pseudo-NMOS NAND gates are worse than CMOS NAND gates. Since pseudo-NMOS logic consumes power even when not switching, it is best used for critical NOR functions where it shows greatest advantage.

Similar analysis can be used to compute the logical effort of other logic technologies, such as classic NMOS and bipolar and GaAs. The logical efforts should be normalized so that an inverter in the particular technology has an average logical effort of 1.

### 10.1.1 Symmetric NOR gates

Johnson [4] proposed a novel structure for a 2-input NOR, shown in Figure 10.2. The gate consists of two inverters with shorted outputs, ratioed such that an inverter pulling down can overpower an inverter pulling up. This ratio is exactly the same as is used for pseudo-NMOS gates. The difference is that when the output should rise, both inverters pull up in parallel, providing more current than is available from a regular pseudo-NMOS pullup.

The input capacitance of each input is 2. The worst-case pulldown current is

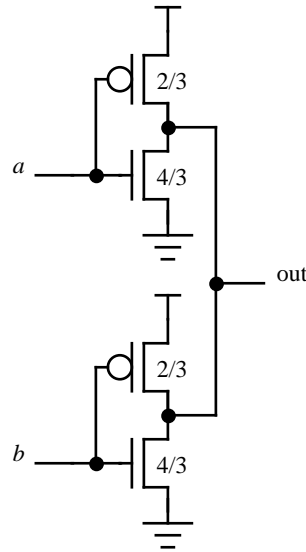


Figure 10.2: Johnson's symmetric 2-input NOR.

equal to that of a unit inverter, as we had found in the analysis of pseudo-NMOS NOR gates. The pullup current comes from two PMOS transistors in parallel and is thus  $2/3$  that of a unit inverter. Therefore, the logical effort is  $2/3$  for a falling output and 1 for a rising output. The average effort is  $g = 5/6$ , which is better than that of a pseudo-NMOS NOR and far superior to that of a static CMOS NOR!

Johnson also shows that symmetric structures can be used for wider NOR functions and even for NAND gates. Exercises 10-3 and 10-4 examine the design and logical effort of such structures.

## 10.2 Domino circuits

Pseudo-NMOS gates eliminate the bulky PMOS transistors loading the inputs, but pay the price of quiescent power dissipation and contention between the pullup and pulldown transistors. Dynamic gates offer even better logical effort and lower power consumption by using a clocked *precharge* transistor instead of a pullup that is always conducting. The dynamic gate is *precharged* HIGH then may *evaluate* LOW through an NMOS stack.

Unfortunately, if one dynamic inverter directly drives another, a race can corrupt the result. When the clock rises, both outputs have been precharged HIGH.

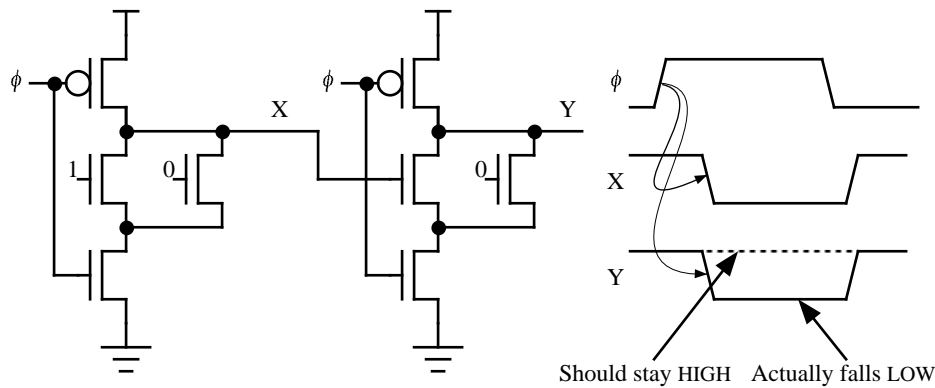


Figure 10.3: Dynamic gates cannot be cascaded directly.

The HIGH input to the first gate causes its output to fall, but the second gate's output also falls in response to its initial HIGH input. The circuit therefore produces an incorrect result because the second output will never rise during evaluation, as shown in Figure 10.3. *Domino* circuits solve this problem by using inverting static gates between dynamic gates so that the input to each dynamic gate is initially LOW. The falling dynamic output and rising static output ripple through a chain of gates like a chain of toppling dominos. In summary, domino logic runs 1.5 to 2 times faster than static CMOS logic [2] because dynamic gates present a much lower input capacitance for the same output current and have a lower switching threshold, and because the inverting static gate can be skewed to favor the critical monotonically rising evaluation edges.

Figure 10.4 shows some domino gates. Each *domino* gate consists of a dynamic gate followed by an inverting static gate<sup>1</sup>. The static gate is often but not always an inverter. Since the dynamic gate's output falls monotonically during evaluation, the static gate should be skewed high to favor its monotonically rising output, as discussed in Section 9.2.1. We have calculated the logical effort of high-skew gates in Table 9.2 and will compute the logical effort of dynamic gates in the next section. The logical effort of a domino gate is then the product of the logical effort of the dynamic gate and of the high-skew gate. Remember that a domino gate counts as two stages when choosing the best number of stages.

A dynamic gate may be designed with or without a clocked evaluation transistor; the extra transistor slows the gate but eliminates any path between power

<sup>1</sup>Note that a domino “gate” actually refers to two stages, rather than a single gate. This is unfortunate, but accepted in the literature.

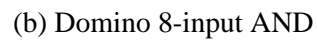
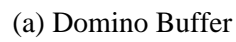


Figure 10.4: Domino buffer and 8-input AND gate.

and ground during precharge when the inputs are still high. Some dynamic gates include weak PMOS transistors called keepers so that the dynamic output will remain driven if the clock stops high.

Domino designers face a number of questions when selecting a circuit topology. How many stages should be used? Should the static gates be inverters, or should they perform logic? How should precharge transistors and keepers be sized? What is the benefit of removing the clocked evaluation transistors? We will show that domino logic should be designed with a stage effort of 2–2.75, rather than 4 that we found for static logic. Therefore, paths tend to use more stages and it is rarely beneficial to perform logic with the inverting static gates.

### 10.2.1 Logical effort of dynamic gates

The logical effort for dynamic gates can be computed just as for static gates. Figure 10.5 shows several dynamic gates with NMOS stacks sized for current equal to that of a unit inverter. Precharge is normally not a critical operation, so only the pulldown current affects logical effort. The logical efforts are shown in Table 10.2.

Logical effort partially explains why dynamic gates are faster than static gates. In static gates, much of the input capacitance is wasted on slow PMOS transistors that are not even used during a falling transition. Therefore, a dynamic inverter enjoys a logical effort only 1/3 that of a static inverter because all of the input capacitance is dedicated to the critical falling transition.

Our simple model for estimating logical effort fails to capture two other reasons that dynamic gates are fast. One is the lower switching threshold of the gate: the dynamic gate output will begin switching as soon as inputs rise to  $V_t$ , rather than all the way to  $V_{DD}/2$ . Another is the fact that velocity saturation makes the resistance of long NMOS stacks lower than our resistive model predicts. Therefore, simulations show that dynamic gates have even lower logical effort than Table 10.2 predicts.

Notice that dynamic NOR gates have less logical effort than NANDs and indeed have effort independent of the number of inputs. This is reversed from static CMOS gates and motivates designers to use wide NOR gates where possible.

### 10.2.2 Stage effort of domino circuits

In Section 3.4, we found that the best stage effort  $\rho$  is about 4 for static CMOS paths. This result depends on the fact that extra amplification could be provided

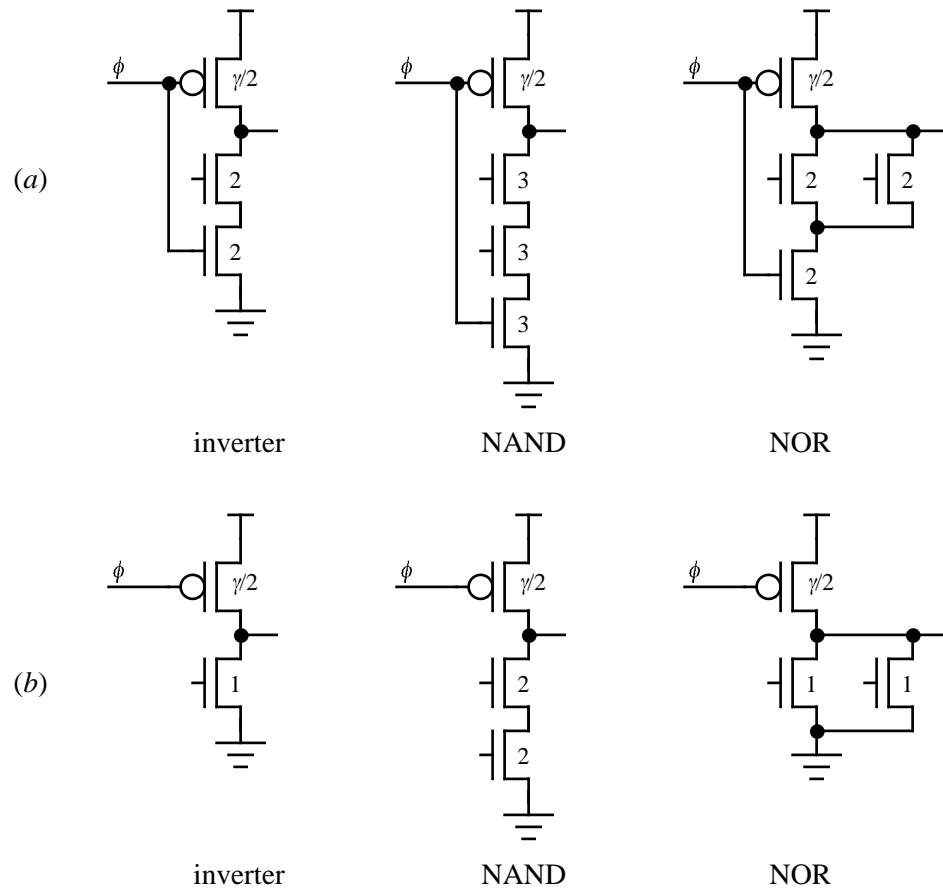


Figure 10.5: Dynamic gates (a) with and (b) without clocked evaluation transistors.



Gate type	Clocked evaluation transistor?	Formula	$n = 2$	$n = 3$	$n = 4$
inverter	yes	$2/3$			
	no	$1/3$			
NAND	yes	$(n + 1)/3$	1	$4/3$	$5/3$
	no	$n/3$	$2/3$	1	$4/3$
NOR	yes	$2/3$	$2/3$	$2/3$	$2/3$
	no	$1/3$	$1/3$	$1/3$	$1/3$
multiplexer	yes	1	1	1	1
	no	$2/3$	$2/3$	$2/3$	$2/3$

Table 10.2: Logical effort per input of dynamic gates.

by a string of inverters with logical effort 1. Domino paths are slightly different because extra amplification can be provided by domino buffers with logical effort less than 1. Adding more buffers actually reduces  $F$ , the path effort! Therefore, we would expect that domino paths would benefit from using more stages, or equivalently, that the best stage effort  $\rho$  is lower for domino paths. In this section, we will compute this best stage effort.

Our arguments parallel those in Section 3.4. We begin with a path that has  $n_1$  stages and path effort  $F$ . We contemplate adding  $n_2$  additional stages to obtain a path with a total of  $N = n_1 + n_2$  stages. This time, however, the gates we add are not static inverters. Instead, they may be dynamic buffers. In general, the additional stages have logical effort  $g$  and parasitic delay  $p$ . The extra gates therefore change the path effort. The minimum delay of the path is:

$$\hat{D} = N \left( F g^{N-n_1} \right)^{1/N} + \left( \sum_{i=1}^{n_1} p_i \right) + (N - n_1)p \quad (10.1)$$

We can differentiate and solve for  $\hat{N}$  which gives minimum delay. When parasitics are non-zero, it proves to be more convenient to compute the best stage effort  $\rho(g, p)$ , which depends on the logical effort and parasitic delay of the extra stages. The mathematics is hideous, but the conclusion is remarkably elegant:

$$\rho(g, p) = g\rho(1, p/g) \quad (10.2)$$

where  $\rho(1, p_{inv})$  is the best stage effort plotted in Figure 3.4 and fit by Equation 3.24. This result depends only on the characteristics of the stages being added, not on any properties of the original path.

Let us apply this result to domino circuits, where the extra stages are domino buffers. The logical effort of a domino buffer, like the one in Figure 10.4, is  $10/18$  with series evaluation transistors and  $5/18$  without. Estimate parasitic delays to be  $(5/6)p_{inv}$  for a high-skew static inverter,  $p_{inv}$  for a dynamic inverter with series evaluation transistor. Since the buffer with series evaluation transistor consists of two stages, each stage has  $g = \sqrt{10/18} = 0.75$  and  $p = (11/6)p_{inv}/2 = (11/12)p_{inv}$ . If  $p_{inv} = 1$ , the best stage effort is  $\rho(0.75, 0.92) = 0.75\rho(1, 0.92/0.75) = 2.76$ . The same reasoning applies to dynamic inverters with no clocked evaluation transistors having  $g = 0.52$  and  $p = 2/3$ , yielding a best stage effort of 2.0.

In summary, domino paths with clocked evaluation transistors should target a stage effort around 2.75, rather than 4 used for static paths. If the stage effort is higher, the path may be improved by using more stages. If the stage effort is lower, the path may be improved by combining logic into more complex gates. Similarly, domino paths with no clocked evaluation transistors should target a stage effort of 2.0. Since it is impractical to leave out all of the clocked evaluation transistors, many domino paths mix clocked and unclocked dynamic gates and should target an effort between 2 and 2.75. As with static logic, the delay is a weak function of path effort around the best effort, so the designer has freedom to stray from the best effort without severe performance penalty.

### 10.2.3 Building logic in static gates

When, if ever, is it beneficial to build logic into the high-skew static gates? For example, consider the two ways of building an 8-input domino AND gate shown in Figure 10.6. One design consists of (dynamic 4-NAND, inverter, dynamic 2-NAND, inverter). Another design is (dynamic 4-NAND, high-skew 2-NOR). Which is better? The logical effort of the path is always larger when high-skew gates are used. However, the high-skew gate could reduce the parasitic delay if it reduces the total number of stages. If the stage effort of the first design is very small, the path may become faster by using fewer stages. But if the stage effort is large, the first design is best. In this section, we will quantify “small” and “large” to develop guidelines on the use of logic in static gates.

We can use our results from the previous section to address this question. The topology should be chosen to obtain a stage effort  $\rho = \rho(g, p)$ , where  $g$  and  $p$  are the mean logical effort and parasitic delay of the difference between the longer and shorter paths. For stage effort below  $\rho$ , the shorter path is better, meaning it is

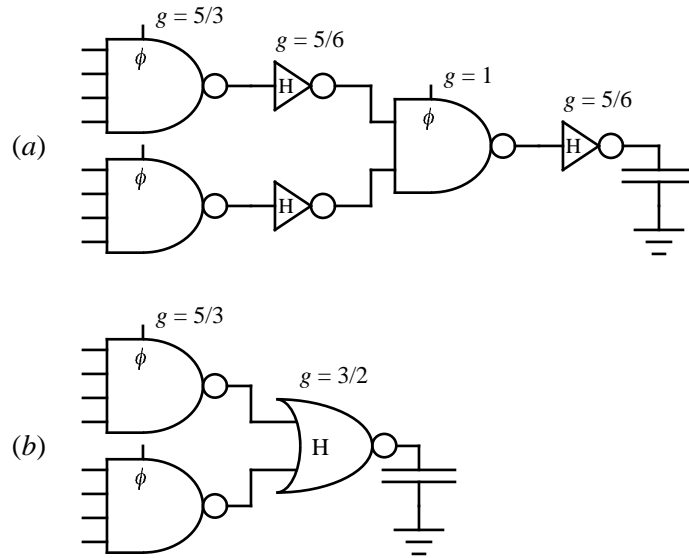


Figure 10.6: Two designs for 8-input domino AND.

advantageous to use logic in the static stage. For stage efforts above  $\rho$ , the longer path is better. An example will clarify this calculation:

**Example 10.1** Which of the designs in Figure 10.6 is best if the path has electrical effort of 1? If the path has electrical effort of 5?

We could solve this problem by computing the delays of each design and directly comparing speed. Instead, we will use the stage effort criteria derived in this section. The change in logical effort from design (b) to design (a) is  $\frac{(5/6) \times (1) \times (5/6)}{3/2} = 0.46$ . This occurs over two extra stages, so the logical effort per stage is  $g = \sqrt{0.46} = 0.68$ . We could work out the parasitic delay exactly, but we recall that over a range of parasitics,  $\rho(1, p)$  is about 4. Therefore,  $\rho(0.68, p/0.68) \approx 0.68 \times 4 = 2.72$ . If the stage effort is below 2.7, design (b) is best. If the stage effort is above 2.7, design (a) will be better.

Design (b) has a logical effort of  $(5/3) \times (3/2) = 2.5$  and thus a path effort of  $2.5H$  and stage effort of  $\sqrt{2.5H}$ . If  $H = 1$ , the stage effort is 1.6 and design (b) is best. If  $H = 5$ , the stage effort is 3.5 and design (a) would be better.

This seems like too much work for a simple example in which comparing delays is easy. The advantage of the method is the insight it gives: one should build logic into static gates only when the stage effort is below about 2.7. Moreover, it is best to first reduce the number of stages by using more complex dynamic gates. Simple calculations show that a dynamic gate with up to 4 series transistors followed by a high-skew inverter generally has lower logical effort than a smaller dynamic gate followed by a static gate. An exception is very wide dynamic NOR gates and multiplexers, which may be faster when divided into narrower chunks feeding a high-skew NAND gate to reduce parasitic delay.

In summary it is rarely beneficial to build logic into the static stages of domino gates. If a domino path has stage effort below about 2.7, the path can be improved by reducing the number of stages. The designer should first use more complex dynamic gates with up to 4 series transistors. If the stage effort is still below 2.7, only then should the designer consider replacing some of the static inverters with actual static gates.

### 10.2.4 Designing dynamic gates

In addition to the logic transistors, dynamic gates have transistors to control precharge and evaluation and to prevent the output from floating. How should each transistor be sized?

The size of the precharge transistor influences precharge time. A reasonable choice is to size it as if the dynamic gate were a low skew gate; hence the PMOS transistor can source half the current of the pulldown stack. Figure 10.5 uses such sizes.

The designer has several choices regarding the evaluation transistor. If the circuit inputs can be designed in such a way that there is no path to ground during precharge, the clocked evaluation transistor can be safely omitted. Even if there is a path to ground during part of precharge, the transistor can be removed if some extra power consumption is tolerable and the precharge transistor is strong enough to pull the output acceptably high in the time available. Frequently a clocked evaluation transistor is necessary. How big should it be?

One reasonable choice is to make the clocked evaluation transistor equal in size to the logic transistors in the dynamic gate. For higher speed, the clocked device can be made larger, just as an unbalanced gate can favor the critical inputs at the expense of the non-critical ones. For example, Figure 10.7 shows a dynamic inverter with a clocked pulldown twice as large as in Figure 10.5. The input transistor is selected so that the total pulldown resistance matches that of a normal

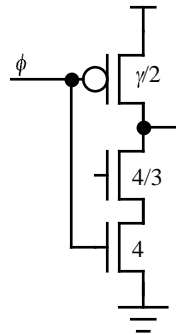


Figure 10.7: Dynamic inverter with double-sized clocked evaluation transistor.

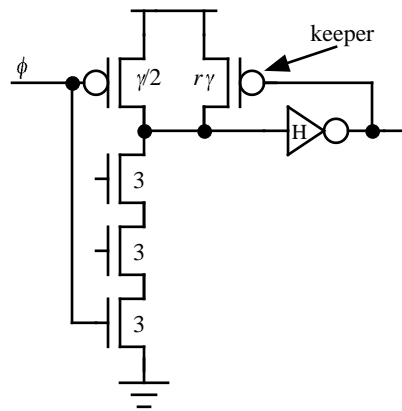


Figure 10.8: Keeper on a dynamic gate.

inverter. The logical effort is thus only  $4/9$ , much better than the effort of  $2/3$  with a normal pulldown size and nearly as good as  $1/3$  for a dynamic inverter with no clocked pulldown. The main cost of large clocked transistors is the extra clock power. Therefore, a small amount of unbalancing such as  $1.5$  or  $2\times$  is best.

Some dynamic gates use keepers to prevent the output from floating high during evaluation, as shown in Figure 10.8. The keepers also slightly improve the noise margin on the input of the dynamic gate. They have little effect on the noise margin at the output because they are usually too small to respond rapidly. The drawback of keepers is that they initially fight a falling output and slow the dynamic gate. How should keepers be sized?

The keeper current is subtracted from the pulldown stack current during evaluation. If the ratio of keeper current to pulldown stack current is  $r$ , the logical effort of the dynamic gate increases by  $1/(1 - r)$ . Therefore, a reasonable rule of

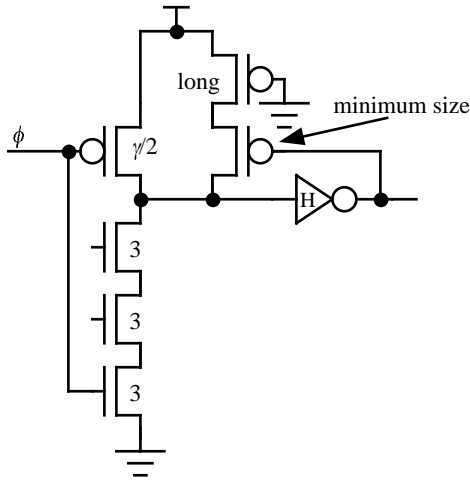


Figure 10.9: Weak keeper split into two parts.

thumb is to size keepers at  $r = 1/4$  to  $1/10$  of the strength of the pulldown stack. For small dynamic gates, this implies that keepers must be weaker than minimum sized devices. Increasing the channel length of the keepers will weaken them, but also add to the capacitive loading on the inverter. A better approach is to split the keeper into two series transistors, as shown in Figure 10.9. Such an approach minimizes the load on the inverter while reducing keeper current.

### 10.3 Transmission gates

Many transmission gate circuits can be analyzed with the method of logical effort by incorporating the transmission gate into the logic gate that drives it. Figure 10.10 shows an inverter driving a transmission gate, and then shows the same circuit redrawn. The second circuit is essentially a leg of a multiplexer (Figure 4.4).

The PMOS and NMOS transistors in the transmission gate should be equal in width because both transistors operate in parallel while driving the output. The strong NMOS transistor helps the weaker PMOS transistor during a rising output transition. We can model the two transistors in parallel as an ideal switch with resistance equal to that of an NMOS transistor for both rising and falling transitions. A larger PMOS transistor would slightly improve current drive on rising outputs, but would add significant diffusion capacitance which slows both transitions.

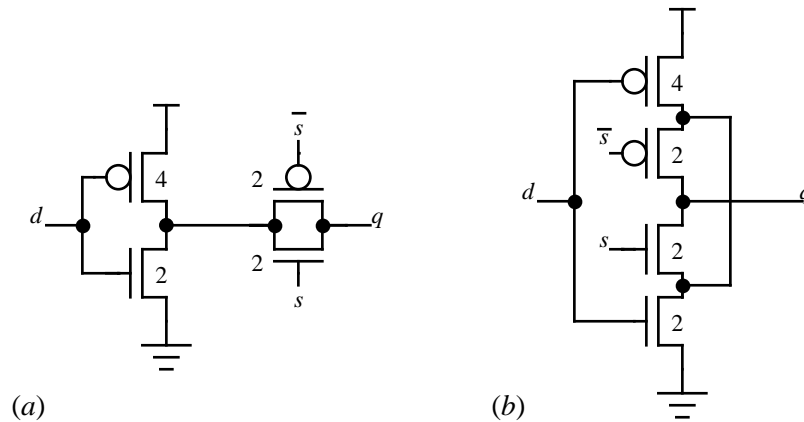


Figure 10.10: An inverter driving a transmission gate, and the same circuit redrawn so that it can be considered to be a single logic gate for the purposes of logical effort analysis.

Given this model, the circuit has drive equal to that of an inverter for both rising and falling transitions. The logical effort is 2 for input  $a$  and only  $4/3$  for  $s^*$ . This improvement in logical effort on  $s^*$  relative to a normal tri-state inverter comes at the expense of increased diffusion capacitance, leaving no great advantage for transmission gate tri-states over normal tri-states.

In general, transmission gate circuits are sized with equal PMOS and NMOS transistors and compared to an inverter with equal output current. As long as a delay equation such as Equation 3.6 describes the delay of a circuit, the method of logical effort applies. However, the parasitic capacitance increases rapidly with series transmission gates, so practical circuits are normally limited to about two series transmission gates.

A common fallacy when characterizing circuits with transmission gates is to measure the delay from the input to the output of the transmission gate. This makes transmission gate logic seem very fast, especially if the input were driven with a voltage source. As logical effort shows, the only meaningful way to characterize a transmission gate circuit is in conjunction with the logic gate that drives it.

## 10.4 Summary

This chapter used ideas of best stage effort, unbalanced gates, and unequal rise/fall delays to analyze circuit families other than static CMOS. Quantifying the logical effort of these circuit families enables us to understand better their advantages over static CMOS and to choose the most effective topologies.

We first examined ratioed circuits, such as pseudo-NMOS gates, by computing separate rising and falling logical efforts. The analysis shows that Johnson's symmetric NOR is a remarkably efficient way to implement the NOR function.

We then turned to domino circuits and found a remarkable result for the best stage effort of a path when considering adding extra stages, given in Equation 10.2. The equation tells us that the best stage effort of dynamic circuits is in the range of 2–2.75, depending on the use of clocked evaluation transistors. The equation also tells us when it is beneficial to break a path into more stages of simpler gates. We conclude that a path should incorporate logic into static gates only when the dynamic gates are already complex and the stage effort is still less than 2.7.

Finally, we explored transmission gate circuits. The logical effort of transmission gate circuits can be found by redrawing the driver and transmission gates as a single complex gate. Neglecting the driver is a common pitfall which makes transmission gates appear faster than they actually perform.

## 10.5 Exercises

**10-1** [20] Derive the logical efforts of pseudo-NMOS gates shown in Table 10.1.

**10-2** [20] Design an 8-input AND gate with an electrical effort of 12 using pseudo-NMOS logic. If the parasitic delay of an  $n$ -input pseudo-NMOS NOR gate is  $(4n + 2)/9$ , what is the path delay? How does it compare to the results from Section 2.1?

**10-3** [25] Design a 3-input symmetric NOR gate. Size the inverters so that the worst-case pulldown is four times as strong as the pullups. What is the average logical effort? How does it compare to a pseudo-NMOS NOR? To static CMOS?

**10-4** [20] Design a 2-input symmetric NAND gate. Size the inverters so that the worst-case pulldown is four times as strong as the pullups. What is the average logical effort? How does it compare to static CMOS?



**10-5** [30] Prove Equation 10.2.

**10-6** [25] Design a 4-16 decoder like the one in Section 2.2, using domino logic. You may assume you have true and complementary address inputs available.

**10-7** [25] A 4:1 multiplexer can be constructed from two levels of transmission gates. Design such a structure and compute its logical effort.

