High Speed CMOS VLSI Design

Lecture 6: Static Circuit Families

(c) 1997 David Harris

1.0 Introduction to Circuit Families

Chips can be optimized at many levels for speed, area, power, yield, etc. For years, the microarchitectural level was most important and innovations such as pipelining and superscalar execution raised processor performance. Now that increasing IPC is very difficult, circuit optimizations are more important for continuing to improve performance. Designers have looked at many circuit families in hopes of boosting performance.

Unfortunately, most circuit families suffer from critical flaws which mean the families are best relegated to old journal articles and dusty Ph.D. theses. Only a few families are relevant to general-purpose high speed circuit design. Static (a.k.a. complementary) CMOS gates are the simplest and most widely used. Pass-transistor logic, in its many flavors, also has niches where it effective. Pseudo-NMOS gates are an efficient way to implement wide NOR gates. Dynamic circuits are trickier to use, but offer the highest performance of any general-purpose circuit family.

In this lecture, we will examine the three important static logic families: static CMOS, pass-transistor logic, and pseudo-NMOS. In the next lecture, we will look at a variety of dynamic logic techniques. In the third lecture, we will consider a few special-purpose circuit styles as well as circuit pitfalls which plague many ill-conceived circuits.

The emphasis will be on high performance. Low power design is also becoming increasingly important and will be covered in a later lecture; in general, static circuits are better than dynamic circuits for low power consumption.

2.0 Static CMOS Logic

Static CMOS gates are the simplest and easiest to use circuit family. Their advantages include:

- Very robust -- “nearly idiotproof!”
- “Zero” DC power (leakage => nothing is zero in the real world)
- Low AC power
- Insensitive to process variation, if you wait long enough

- Scales well to low voltage
- Handled well by synthesis tools and simulators
- Well understood

Thus, static CMOS gates should be used by default unless there is a good reason to do otherwise. Designers have a few tricks to squeeze maximum performance out of static gates. They include topology selection, skewing gates, optimizing for critical inputs, and perhaps tapering stacks. Let’s discuss each further.

2.1 Topology Selection

The most important part of creating fast circuits is to choose a good topology of gates. The right number of stages depends on the total gain of the path and should result in a gain of about 4 per stage, though anything in the range 2-6 is not bad. Thus, if very little logic must be performed but the fanout is large, simple stages like 2-input gates and inverters should be used. If lots of logic must be performed but little fanout is necessary, complex multiple input gates should be used. CMOS logic has the advantage of supporting arbitrary AND-OR-invert structures as well as simple NAND and NOR gates.

Nevertheless, beyond a certain point, complex gates are always slower than two stages of simpler gates. This occurs because delay from internal parasitics increases quadratically with the number of series transistors. The maximum number of series transistors in a typical CMOS gate should be about 4 NMOS or 3 PMOS. An exception is that 5 series NMOS devices in domino gates are sometimes acceptable, especially since the bottom one may be a wide clocked transistor that turns on early.

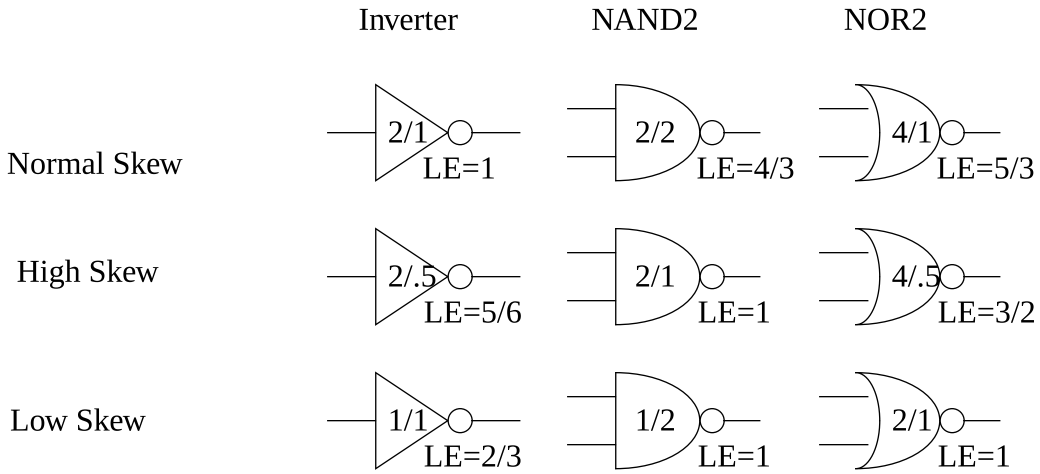
DeMorgan’s law provides a fair amount of flexibility choosing between NAND and NOR gates. In general, NAND gates are preferable because their series stacks consist of fast NMOS transistors rather than slower PMOS transistors. This is clear from the logical efforts of gates.

2.2 Skewed Gates

When a transition in a particular direction is known to be most critical, CMOS gates may be skewed to favor that transition, as mentioned in a previous lecture. This speeds up the critical edge at the expense of the other edge. Care must be taken to ensure the other edge does not become a performance limiter. How do we compute the logical effort of such skewed gates?

The logical effort of a skewed gate is equal to it’s input capacitance divided by the input capacitance of an inverter that has the same drive strength on the critical transition. For example, Figure 1 shows the sizing and logical efforts of skewed inverters, NAND gates, and NOR gates.

FIGURE 1. P/N Ratios

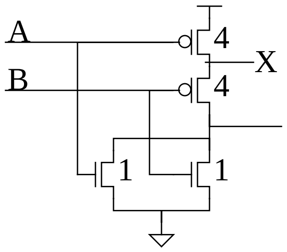


As expected, skewed gates have lower logical effort because less input capacitance is dedicated to the non-critical edge.

2.3 Critical Inputs

The delay of a gate depends on the input pattern. For example, consider a 2-input NOR gate shown in Figure 2. Input A is closest to the rail and is thus called the outside input. Input B is closest to the middle and thus is called the inside input.

FIGURE 2. 2-input NOR gate



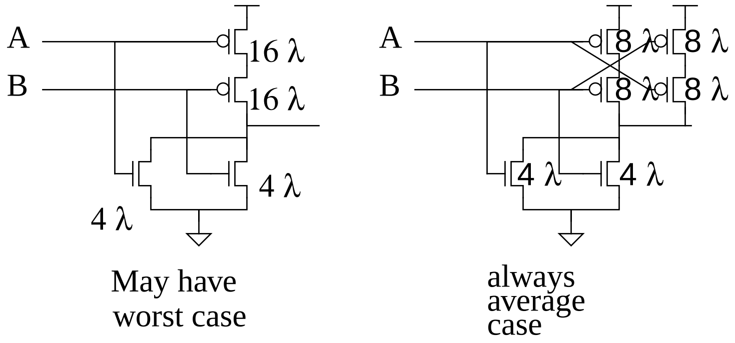
Consider the delay from A and B falling to the output rising. If A falls first, it can charge up the capacitance on node X. When B arrives, the output will respond quickly because X is already set and because charge sharing between node X and the output will further reduce the delay. On the other hand, if B falls first, X will be low. When A arrives, the output will rise more slowly because both the output capacitance and node X must be charged up. This is confirmed by the measured data in Figure 3 which shows the inner input to be substantially faster then the outer input for small loads. When both inputs switch simultaneously, the gate is even slower because neither transistor is fully turned on during the input transition time.

FIGURE 3. Curve fitting delay vs. fanout for 2-input NOR (0.8 μm process)

When fanout becomes large, the load capacitance dominates the internal capacitance, so the inner input has less of an advantage. More subtly, when the inner input switches last, it causes some negative feedback because node X will droop, reducing the V_{gs} of the inner input. This negative feedback is most important for slow input slopes because such slopes keep the inner transistor in saturation for longer periods of time, during which current depends quadratically rather than just linearly on V_{gs}. The outer input does not experience this problem because its source cannot droop. In the simulation used to produce Figure 3, the input slopes increased along with output slopes. Thus, the inside input experiences more negative feedback at higher fanouts. For very large fanouts, generally well beyond the useful loading range of a gate, the inside input can become slower than the outside input.

When possible, early signals should drive transistors close to the rails for best speed. Sometimes the arrival times of signals are unknown. In such a case, it may still be possible to get an average delay instead of worst-case delay, especially when transistors are large enough that they had to be folded anyway. Such a scheme is shown in Figure 4:

FIGURE 4. Twisting stacks for average delay



2.4 Tapering Stacks

As we have seen earlier, delay of long series transistor stacks increases quadratically with the number of series transistors because both resistance and parasitic capacitance increase proportionally to the number of transistors. It would be convenient to be able to build taller stacks without parasitic delay getting out of hand. To do this, some designers have proposed tapering their transistor stacks to use wider transistors near the rails. This leads to an interesting derivation which suggests gate delay really can increase only linearly with stack height. Unfortunately, it overlooks several practical problems that overshadow the potential benefits. Let's look more closely at tapering stacks, then see why it is usually a bad idea.

Figure 5 shows a stack of three transistors with contacts between each. The delay is quadratic in the number of series transistors. Figure 6 shows a tapered stack of three series transistors. The transistors near the bottom of the stack are larger, providing more current to discharge the internal parasitics. With suitable choice of tapering, the delay can be shown to increase only linearly with the number of series transistors.

FIGURE 5. Contacted 3 transistor stack

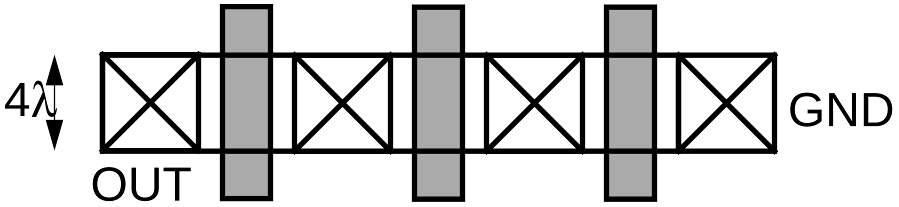
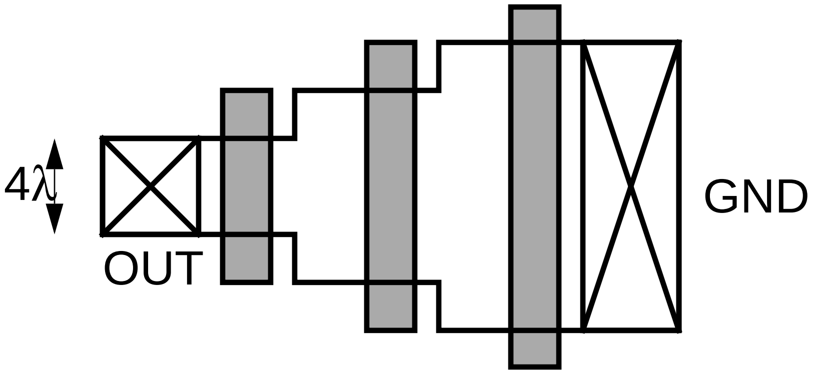


FIGURE 6. Tapered 3 transistor stack

There are two drawbacks with this tapering. One is that the transistors near the rail must be quite large. This increases the loading on previous stages. If the previous stages were also critical, the increased delay of the previous stage may swamp out savings in delay of the tapered stage.



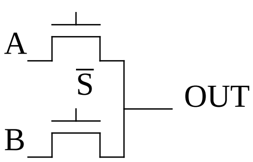
More subtly, the spacing between gates in a tapered stack is larger than the spacing between uncontacted transistors. Therefore, tapering an uncontacted tack can actually increase diffusion parasitics. Thus, it is usually a good idea only to taper stacks at contacted nodes.

Despite these limitations, tapering is occasionally useful. If non-critical early signals can drive transistors close to the rails, widening these transistors is beneficial. Since the signals were early anyway, their load can be increased without overall performance penalty. These wide transistors near the rails can now supply more current when the critical inputs arrive near the middle of the gate. A common application of this is to double the width of the clocked pulldown transistor in a dynamic gate. As long as the clock arrives well before the critical inputs, this will speed up the dynamic gate. A drawback is increased clock loading and power consumption.

3.0 Pass-Transistor Logic

Another popular static circuit family is pass-transistor logic. The idea is to use transistors as switches to efficiently implement multiplexors and related structures. The simplest 2-input multiplexor imaginable can be implemented with just two pass transistors, as shown in Figure 7. It has two major limitations: the output does not swing rail-to-rail, and the output is not buffered so the gate cannot be cascaded to drive arbitrary other gates. These limitations can be overcome by adding explicit buffers and by either employing both NMOS and PMOS transistors in parallel to pass both 0's and 1's effectively or by providing a level restoration circuit on the output. These options lead to several flavors of pass-transistor logic: CPL, DPL, SRPL, LEAP, etc. We'll look at these variations, then consider how to size pass transistors and conclude with recommended applications.

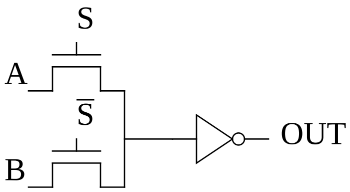
FIGURE 7. Simple Pass-transistor multiplexor



3.1 Pass-transistor Variants

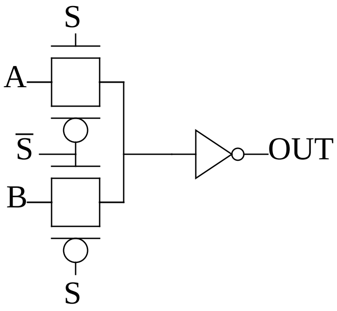
All pass-transistor circuits should end with a buffer (usually an inverter) to decouple the output from the inputs, allowing pass-transistor gates to be cascaded. If the simple pass-transistor multiplexor of Figure 7 were just buffered as shown in Figure 8, the circuit would have power-consumption problems. When the selected input is high, node X will only rise to VDD-Vt. This leaves the PMOS transistor in the inverter right at the border of ON and OFF. Thus, some DC power consumption could be expected. To avoid this, better pass-transistor families pull even the internal nodes rail-to-rail.

FIGURE 8. Buffered pass-transistor multiplexor



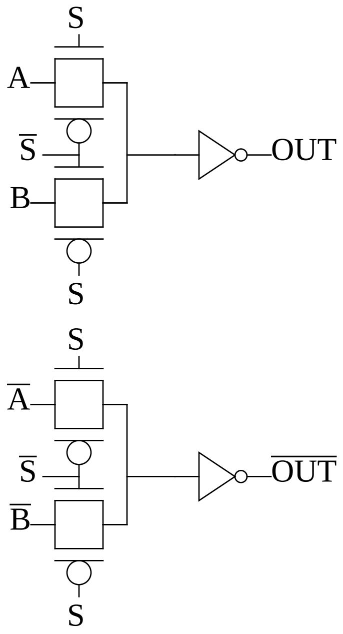
One way to avoid this problem is to gang PMOS transistors in parallel with the NMOS transistors, thus passing both 0’s and 1’s well, as shown in Figure 9. These parallel structures are called pass-gates or transmission gates.

FIGURE 9. Transmission-Gate multiplexor



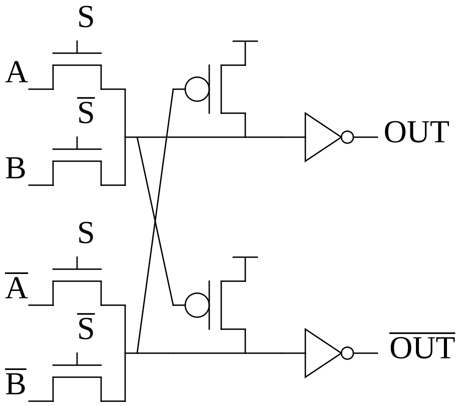
Unfortunately, to cascade such transmission gates, both true and complementary outputs must be available to drive the NMOS and PMOS gate inputs of subsequent stages. This can be done either with inverters, which add extra gates to the critical path, or by building dual-rail logic with accepts both true and complementary inputs and produces both true and complementary outputs. The second scheme is called double pass-transistor logic (DPL) and is illustrated in Figure 10.

FIGURE 10. DPL multiplexor



By now, we’ve grown from 2 transistors in the simple multiplexor to 12 transistors for DPL! Moreover, the PMOS transistors are slow and add substantial diffusion loading. For most circuits, DPL is not competitive. Since the poor performance from the PMOS transistors, it would be reasonable to remove the PMOS transistors from the transmission gates and instead use a cross-coupled level restoring circuit, as shown in Figure 11. This is known as complementary pass-transistor logic (CPL).

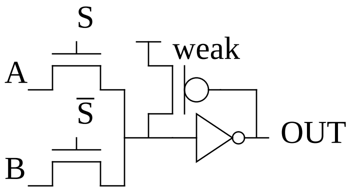
FIGURE 11. CPL multiplexor



CPL is fast because it only uses NMOS transistors in the signal path and because the cross-coupled PMOS devices are fairly fast for level restoration. The circuit is ratioless in the sense that it will eventually produce the correct answer no matter how transistors are sized, but care should be taken sizing the PMOS devices to make them strong enough to be effective, yet weak enough to not slow falling transitions too badly.

All dual-rail logic families require twice as many transistors and wires as the single-rail equivalents. Thus, sometimes, designers may prefer single-rail pass-transistor gates even if extra inverters are required when complementary results are needed. Level restoration can still be performed with feedback, this time taken from the output, as shown in Figure 12. The scheme is called LEAP and produces dense logic. Unfortunately, the feedback PMOS is slower than CPL. Moreover, the circuit is ratioed, meaning that if the PMOS device is too strong, the output may get trapped low and never be able to return high. Thus, the PMOS must be weak enough that it can be overridden even in the SF process corner. Moreover, the resistance of the gate driving inputs A and B, and the wire resistance to these inputs, must be low. Finally, performance gets much worse as VDD drops to a few multiples of the Vt and the threshold drop thus becomes more significant.

FIGURE 12. LEAP multiplexor



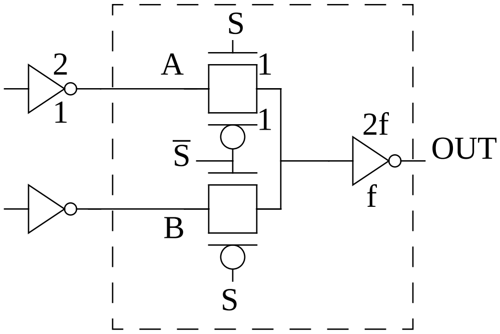
Transmission gates are susceptible to back-driving. Suppose in the circuit above, S=0 and $\overline{S}$ = 1. Let S rise to 1, but suppose that $\overline{S}$ is produced by an inverter and does not immediately switch. During the time that both select lines are 1, node A and node B are connected by a path through the two NMOS transistors. This may cause glitches that could propagate to other circuits which receive signals A and B. This is an especially serious problem if A or B are dynamic nodes or could drive dynamic gates that can't tolerate glitches. Even when there is no circuit failure, back-driving burns extra power. Thus, it is important to design the select signals to arrive at nearly the same time.

A final caveat of all pass-transistor families is that layout is more difficult and less dense than for static CMOS gates. Routing between source/drain diffusions can increase internal parasitic capacitances and slow gates more than expected.

3.2 Logical Effort of Pass Transistors

Pass-transistors may have inputs to source/drain diffusion as well as to gates. Their logical effort therefore depends not only on the pass-transistor gate itself, but also on the driver. For instance, consider the transmission gate multiplexor of Figure 13, with the A and B inputs driven by inverters.

FIGURE 13. Logical Effort of transmission-gate circuit

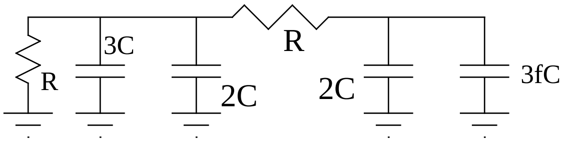


Notice how the PMOS and NMOS transistors are sized equally in the transmission gate. This is because the PMOS is only very important for pulling the output from VDD-Vt to VDD. Sizing it larger would greatly increase diffusion capacitance while only slightly reducing transmission gate resistance.

The resistance of the transmission gate depends on whether it is pulling up or down. Suppose that a transistor pulling the wrong way (i.e. an NMOS transistor pulling high or PMOS transistor pulling low) has on average about twice its usual resistance. Thus, pulling high, the transmission-gate resistance is 2R (NMOS) in parallel with 2R (PMOS) = R. Pulling low, the resistance is R (NMOS) in parallel with 4R (PMOS) = 0.8R. This is close enough that we could approximate the resistance as R in both directions.

Therefore, the equivalent circuit of the input inverter driving the output inverter through the transmission gate is:

FIGURE 14. Equivalent circuit for transmission-gate delay calculation



and the Elmore delay from input A is R(3C+2C) + (R+R)(2C+3fC) = 9+6f τ. To get drive strength equal to a unit inverter, we would have to double the transistors sizes. Thus, the logical effort from input A is 6/3 = 2. The logical effort from input S is only 4/3 because only pass-transistor gates must be driven.

Pass-transistor gates can have substantial internal diffusion capacitances. For gates which use full transmission-gates, cascading more than 2 or 3 transmission gates without buffering causes unacceptably slow switching. CPL gates using only NMOS transistors resemble NMOS series stacks and thus can have 3 or 4 series transistors before slowing greatly.

3.3 Recommendations

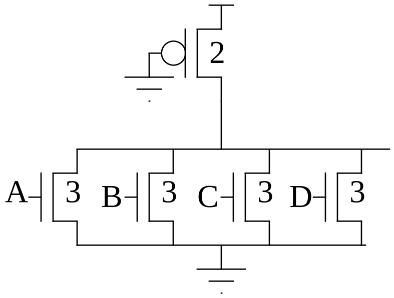
Pass-transistor logic families are primarily beneficial for multiplexors and other circuits that can be built from multiplexors. Examples include XOR gates, alone or in full adders and parity circuits. Wide multiplexors, such as those needed in superscalar execution unit bypass paths, also are efficient to implement with pass transistors. Many papers have been published purporting to prove that the author’s favorite pass-transistor logic family offers improved speed and lower power than static CMOS. Such comparisons almost always are based on the delay of full adders, which are implemented well with pass transistors and poorly with static CMOS. Moreover, the comparisons often use suboptimal static CMOS implementations! A fairer comparison finds that for most general circuits, static CMOS is better. Nevertheless, designers should keep pass-transistor logic in their book of tricks and apply it in the circumstances where it is useful.

4.0 Pseudo-NMOS Logic

Once upon a time, NMOS logic ruled the MOS world. Gates were constructed with NMOS pulldowns and resistive pullups. Since actual resistors take large amounts of area in a conventional process, the resistors were built from transistors that were always ON. The NMOS logic gates efficiently implemented NOR functions, in contrast to CMOS logic gates which favor NAND functions. The resistive pullups cause NMOS gates to dissipate DC power when their output is low. This became excessive when tens of thousands of gates were integrated into a single chip, so NMOS fell out of favor and was replaced by CMOS which dissipates almost no DC power.

Nevertheless, the idea of NMOS circuits is still useful for implementing wide NOR gates. The resistive pullup can be built from a PMOS transistor with its gate tied to ground. Such a gate is shown in Figure 15.

FIGURE 15. Pseudo-NMOS NOR4 gate

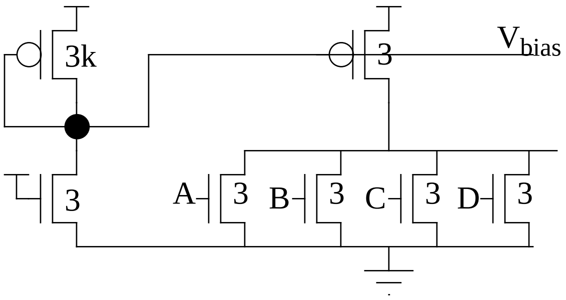


The PMOS transistor must be ratioed such that in the SF corner it is weak enough that the NMOS transistors can pull the output sufficiently low. Sufficiently low depends on the noise margins that can be tolerated. In the FS corner, the PMOS pullup is therefore very slow.

The logical effort is the ratio of input capacitance to that of an inverter with the same drive strength. This depends on whether the gate is pulling up or pulling down. The NMOS transistors produce a current proportional to 3/R. The PMOS draws off a current of roughly 1/R that would otherwise have been available to charge a load. Thus, the total output current is proportional to 3/R - 1/R = 2/R, corresponding to a resistance of R/2. An inverter must be sized twice minimum for this resistance, with an input capacitance of 6. Therefore, the logical effort for falling transitions is 3/6 = 1/2! Unfortunately, the pullup is performed through only a single PMOS transistor with an effective resistance of R. A unit-sized inverter also has resistance R and input capacitance 3. Therefore, the logical effort pulling up is 3/3 = 1. Unless the critical path only involves the pulldown, a conservative designer may size with the larger logical effort. Nevertheless, this logical effort of 1 is still quite good and does not depend on the number of inputs to the NOR gate!

One way to improve the tracking of the P/N ratio across process corners and thus avoid the weak pullup performance is to supply a bias to the PMOS transistor that depends on the processing. This can be done with a current mirror. For example, the current mirror shown in Figure 16 sets the PMOS current to be 1/k that of an NMOS transistor, independent of process. k is usually chosen to be 2-3, depending on desired output low levels. Thus, noise margins will be better in the SF corner and performance will be better in the FS corner.

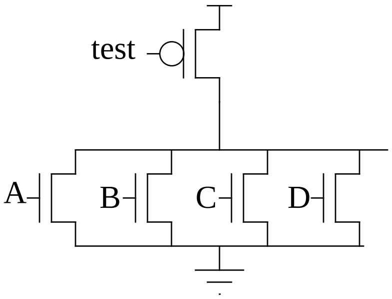
FIGURE 16. Current mirror control of pseudo-NMOS gates



Care must be taken with the bias generator. If a single bias were generated centrally and routed to all pseudo-NMOS gates, the bias would be very susceptible to IR drops, power supply variation, and noise coupled on to the routing. Hence, care must be taken to keep the analog bias voltage stable and the bias must usually be generated locally to avoid power supply noise across the chip.

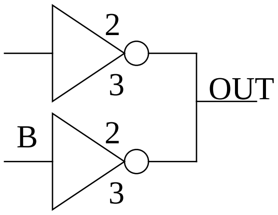
Pseudo-NMOS gates also suffer from DC power consumption. This is a problem for IDDQ test methods which measure the DC current to detect short circuits. One solution is to connect the PMOS gate to a special test signal instead of ground. During IDDQ test, the test signal is asserted high to turn off the PMOS pullups. During normal operation, the test signal stays low.

FIGURE 17. Pseudo-NMOS gate with power-down test mode



A variant on pseudo-NMOS gates is Johnson's symmetric NOR. Such gates consist of two or more inverters with their outputs tied together. When ratioed appropriately, the NMOS transistors can overpower the PMOS transistors. Consider the symmetric 2-input NOR gate in Figure 18. When both inputs are high, the output is low and no power is consumed. When one input is high, the NMOS transistor in one inverter overpowers the PMOS transistor in the other inverter to keep the output nearly low. Some power is consumed. When both outputs fall low, both PMOS transistors turn on, pulling the output high faster than a regular pseudo-NMOS gate could operate. No power is consumed in the high state either.

FIGURE 18. Symmetric 2-input NOR gate



As usual, the logical effort of the gate is the ratio of the input capacitance of the gate to the input capacitance of an inverter with the same drive strength. The worst case is when one input is high and the other is low, causing contention. The effective resistance can be estimated by subtracting the current drawn by the PMOS from the current pulled by the NMOS. The current is thus 3/R - 1/R = 2/R, corresponding to a resistance of R/2. Just to check, the case of both PMOS devices pulling up in parallel also has an effective resistance of R/2. This is the same resistance as a double-sized inverter, with input capacitance 6. Therefore, the logical effort of this NOR gate is 5/6! This is quite good.

It is interesting to observe that with the opposite ratioing such that PMOS transistors always override NMOS transistors, a symmetric NAND gate could be constructed. Symmetric NANDs use such large PMOS transistors that they are not very efficient and should be avoided.

5.0 Summary of Static Circuit Families

In summary, conventional static CMOS gates are the most generally useful circuit family. They can be optimized by choosing good topologies, by placing critical inputs near the center, and by skewing gates where appropriate. Pass-transistors offer advantages for multiplexors and related circuits, such as XOR gates and full adders. Level restoration is the major challenge in using pass-transistors. Pseudo-NMOS gates are good for wide NOR structures where they eliminate the PMOS stack, but suffer from DC power consumption and ratio challenges.