Logic Effort Revisited

Mark Horowitz

This note will take another look at logical effort, first reviewing the basic idea behind logical effort, and then looking at some of the more subtle issues in sizing transistors.

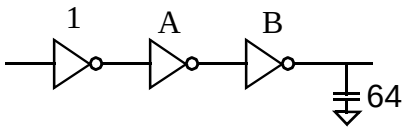
1.0 Background

As a starting point let's review basic CMOS gate delay. We model the delay of a gate to be in the form:

$$Td = Tparasitic + Tload * Cload/Cin \quad \text{(EQ 1)}$$

We can get these parameters from any gate by measuring its delay vs. fanout, and finding the slope (Tload) and the intercept (Tparasitic). We can also get the parameters from using our RC model of a MOS transistor. In the terminology of logical effort, the first term is the parasitic delay, and the second term is the effort delay. We will see that this effort delay is critical for sizing.

Given a string of gates we want to know what the optimal sizing is for these gates in a chain[1] For a given number of gates in a chain, it is relatively easy to find the sizes. If we are looking for minimum delay, then it should not be possible to change any of the device sizes and decrease the delay. If I make gate B 1% larger, the increase in delay of gate A should match the decrease in the delay of gate B. In other words changing Cload of Gate A by 1% should cause a delay change that is equal in magnitude to the delay change caused by increase Cin of gate B by 1%. Given (EQ 1), this balance will occur when the effort delay of each gate is the same. The balancing of the effort delays for minimum delay is the main result of logical effort.



1.1 Logical Effort

Rather than working in ns, we generally work in a normalize space. Since an inverter is generally the fastest gate one can build, it is used as the ruler to measure other (more complex) gates. We define the logical effort of an inverter to be '1'. The effort delay is then written as the product of three factors -- a technology constant, the logical effort of the gate, and the electrical effort of the gate (which is just the fanout Cload/Cin). The technology constant is just Tload for an inverter, and the logical effort is the ratio of Tload(gate)/Tload(inv). This sets the technology constant to be the number of picoseconds needed for

1. You should be careful about finding optimal solutions -- optimal for what? This note is mostly interested in optimal timing, but in most circuits you need to worry about power and area too. These other factors often set the transistor sizing

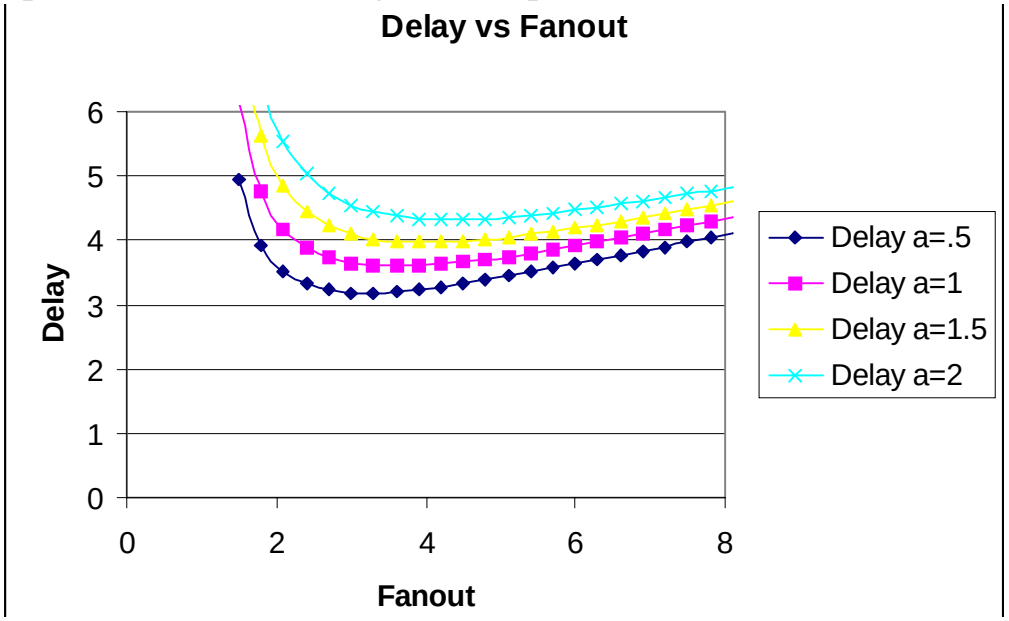
an inverter to drive an addition load equal to its input capacitance. I call the product of the electrical effort (fanout) and the logical effort the ‘effective fanout’ of the gate.

The logical effort of a gate is the slope of the delay vs. fanout for the gate, divided by the slope for an inverter. It is easy to estimate this number using the simple RC model we used in EE271. Since delay of the gate is set by a sum of RC time constants, the effort delay comes from the term which is the effective resistance of the gate driving the load capacitance. For CMOS gates the resistance is inversely proportional to input capacitance, so the logical effort of a gate is:

$$LE = \frac{C_{gate}R_{gate}}{C_{inv}R_{inv}} \qquad \text{(EQ 2)}$$

One can find the LE by either making the input capacitance of the gate equal to the inverter, and looking at the resistance ratio, or by making the resistance the same and looking at the capacitance ratio.

1.2 Optimal Number of Stages and Optimal Fanout



The question of the optimal fanout and number of stages in a chain, is both simple and a little tricky. This answer comes from solving the delay equation for a string of gates:

Total delay = N * (f * Tload+Tparasitic) = ln(CL/Cin) * Tload * (f + α) / ln(f) (EQ 3)

where α is Tparasitic/Tload.

From the delay plot, it does seem like ‘4’ is the right answer, since it seems to make the delay close to the minimum. In addition, the curves are flat, so there is not a huge problem with using a range of fanouts between 2 and 8. So far this all seems to be review.

Now notice that (EQ 3) did not specify the logical effort of the gate. It turns out that the optimal fanout, where fanout is the electrical effort (Cload/Cin), for any string of **homogeneous** gates is around 4. So it would seem at first glance we have a conflict. The theory of logical effort said we should make the effective fanout (fanout * LE) 4. The reason for this confusion is that life gets more interesting when the gates are not all the same.

With different gates you have a choice on how you are going to drive your load. If the fanout per stage is too high, you could always add an inverter to the chain to drive the load. When adding buffers, we always think of using inverters since they are the fastest way to drive fanout. So the optimization problem we think about when sizing fanout is should I use this pile of gates to do my logic with n stages, or should I add a buffer (or two) and do it in n+1 (or 2) stages. Since the extra stages we are thinking about are inverters, it is this element that should have a fanout of around 4 for optimal performance. With inverters having a fanout of 4, it means that the effort delay of all other gates should be around 4 as well (they all have to be the same) and so the fanout of the logic gates will be less than ‘4’.

Understanding that the optimal effective fanout is when the best buffer element has a fanout of 4ish is important when there are gates faster than inverters, as we will see in section on domino logic. In domino logic, the effective fanout (LE * Cload/Cin) is around 3.

1.3 Parasitic Delay

The parasitic delay of a gate can be complicated to calculate, and can depend on which input causes the output to transition. In general we don’t need to calculate this precisely, but you should know that the parasitic delay of a gate grows more than linearly with the number of inputs. This is because the parasitic capacitance of parallel transistors grows linearly with the number of inputs (n transistors of the same size), while the parasitic of series devices grows quadratically (n transistors, each n times larger). The large parasitic delay makes large fanin gates rarely useful. In general, gates with more than 3 stacked devices are rarely used.

2.0 Using Logical Effort

Logical effort is a very handy tool, but there are some situations that will still require some thought. The first problem that arises in real designs is that the input and output capacitance are rarely given to you in a nice form so you can calculate logical effort. Even when this information is available there are often fixed wire capacitance that needs to be driven. All this makes formulating the problem more difficult, but not impossible. It just takes some reasonable approximations.

Once the problem is formed, you often find some circuits that don’t fit nicely into the logical effort framework, so I will talk about these next. The issues that I will discuss are side loads, logical efforts for transmission gates, and dealing with reconvergent fanout. Finally I will take a quick look at using logical effort for dynamic gates.

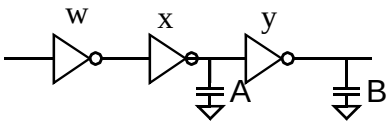
2.1 Forming The Problem

In many design problems the input or output capacitance is not known, and you need to come up with reasonable goals. In many situations focusing on the sizes that optimizes the delay is often not the correct strategy. If wire cap is present, the optimal sizing is always to make the transistor sizes so large that the wire cap is a small fraction of the total. While this often a good idea, you generally stop when the transistors are 60-70% of the total, since the return on area and power has gotten pretty poor. If you really like solving optimization problems then you need to add power or area constraints when you set the problem up.

In some designs, while the input and output capacitances are not know, the fanout is known. A multiplier is a good example of this situation. While it is possible that each row of the multiplier would have a different size, this would be a pain to layout, so a better design would use all the same sizes. Often assuming the input and output capacitances are the same is a good starting point (you can drive as much as you load your inputs). This starting point is a good base number since it is easy to correct if you find out you need to drive a larger load (if the output is 4 times the input cap, the delay will increase by one FO4 inverter).

Lastly in many designs there are large wire capacitors that act as nice breaking points for the sizing problems. While the optimal solution would have the subsequent gate size depend on the gate driving the large wire, as the next section will show, the gain is small, and the cost in power and area is large. In general breaking the problem at this point, and using a reasonable size for the subsequent gate works well.

2.2 Fixed Side Loads



In doing a sizing problem, one often runs into a situation like shown in the figure where are two fixed loads that need to be driven, ‘A’ and ‘B’ that are separated by one or more gates. These loads could be from wire capacitance, or loading of non-critical gates. Although this problem is hard to solve exactly, it is easy to solve approximately. The approximation starts by ignoring cap ‘A’ and find the sizes for the gates if it did not exist. In this process if you need to add buffer stages to get the fanout correct, it is always best to add them BEFORE cap ‘A’ since this will minimize its effect. Adding buffers before ‘A’ increases the cap of gate ‘y’ which will make the side load relatively smaller in comparison.

There are three cases that can arise: the cap of ‘A’ can be much smaller, much larger or about the same as gate ‘y’, In all scenarios the procedure is basically the same: add the cap of ‘A’ to the load on gate ‘x’, and recalculate the fanout of the buffers before the cap. Unless the number of buffers in the ‘prechain’ change, you are basically done. You could recalculate the FO of the buffers in the post chain, but this has only a very small effect on the delay, and tends to make the post cap buffers much larger for small improvements in

delay. I would recalculate the fanouts in gates after ‘A’ only if I am going to remove a buffer in this chain, or I added one in the prechain. Let’s look at each scenario in a little more detail.

In all the case where cap ‘A’ is small compared to the cap of gate ‘y’, the initial sizes are correct, and the problem is solved. Since this is the minimum delay situation, it is good to try to get the side load further down a chain of gates where its effect will be smaller.

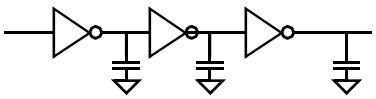
The other easy case is where ‘A’ is much larger than the gate cap. In this case resolve the problem assuming that ‘A’ is the output. Once these sizes are found, the optimal ‘y’ size is easy to find:[1]

$$\frac{x}{w} = \frac{A+y}{x} \qquad y = \sqrt{Bx} \qquad \text{(EQ 4)}$$

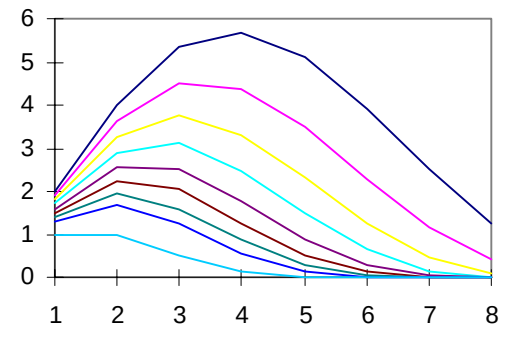
Since ‘y’ is small compared to ‘A’ the interation converges quickly. Also notice that this is going to push the post chain to using small fanouts per stage, and increase the loading on the prechain. We can guess that this change in the output fanout will have little effect on delay because the improvement in delay in going to low fanout is small (the fixed parasitic loads really hurt), and this gain is partially offset by the large load on the prechain. The net result is that doing this optimization only slightly improves performance at the expense of area.

When ‘A’ is comparable to ‘x’ the change in the fanouts in the prestage will be small, since we amortize the change over a number of stages. If they did not change at all (‘x’ did not change) the fanout in the post stage would be exactly correct, since y/x=B/y. Since ‘x’ only changes slightly, we should not be surprised that its effect on the fanout of the post cap gates is small. Said a different way, our method gets the change in fanouts from the fixed load approximately correct, so the change in delay to get it exactly correct is small.

Just to demonstrate that finding the optimal delay is often not the right approach, I solved the for optimal delay for a chain of gates, when each gate drives a fixed side load.



You might guess that the optimal sizes have each of the gates be the same, but that would not be right. Since the delay decreases if the fixed load is small compared to the capacitance of the gate, the optimal sizes ramp up in the middle of the chain and then ramp them back down (making the fanout less than 1) The curves are for different numbers in the chain, and the y axis is the size of the gate relative to



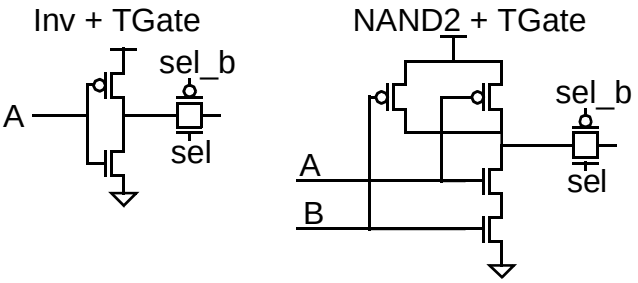
1. The equation is not much more complicated when there are chains of gates instead of the single gate ‘x’, since the fanout of all the gates in the chain remains the same

the fixed load. What is interesting is that the overall delay is only slightly better than a chain where all the gates are the same size and roughly equal to the load capacitance.

2.3 Transmission Gates and other issues with Gate sizing

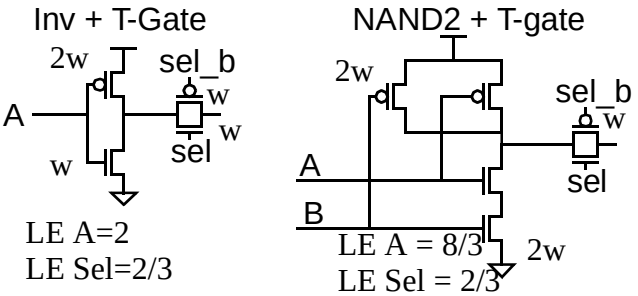
For all the gates that we have talked about in class, we make the logical effort of all the inputs the same. This strategy is a good one, since if you try to improve the LE of one input, the LE of the other inputs gets much worse. But what should you do about transmission gates?

First since we are interested in the effective resistance driving the load capacitance, we can only find the LE for a transmission gate when it is used in some context. The LE needs to be found for the supergate it is in. Now the question is how to size the devices in the inverter and the TGate.

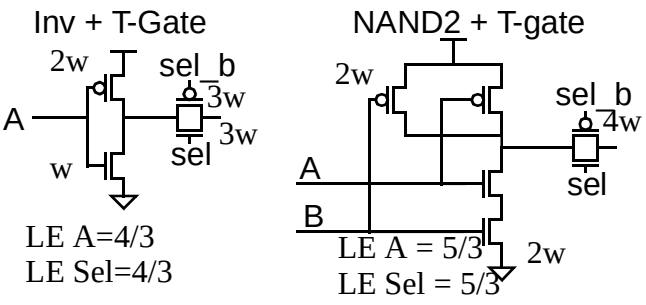


To start we need to know whether we should consider sel and sel_b two different signals, or should we add the cap together. The answer is described in more detail in the next section, and turns out that the logical effort of a splitter (creates A and A_b) is only slightly greater than one. So the logic effort should be considered by only looking at the ‘sel’ input.

Now how do we want the logical efforts to turn out? If the select signal is critical we would want to make the LE of this signal small, and let the LE for the inverter or NAND gate grow larger. This is exactly what happens if we make the resistance of the TGate equal to the resistance of the inverter. What is happening is that the TGate doubles the resistance of the gate without changing the input capacitance.



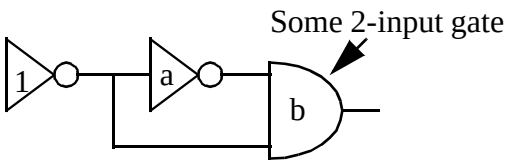
To get equal logical efforts, the resistance of the transmission gate must be made much smaller than the resistance of the inverter. Since the resistance of the supergate is the same for both inputs, to get equal logical efforts, we make the input capacitance the same. This makes the transistors in the TGate 3w for the inverter, and the LE of the both A and Sel 4/3. Obviously you can set the LE of the ‘sel’ input at any point in



between these points. The ‘best’ LE depends on which of the inputs is critical. I don’t think it would make sense to go outside these two points, since both are getting a little inefficient in terms of making use of the transistor width consumed.

2.4 Reconvergent Fanout (phase splitting)

Sizing gets a little more complex when we end up with a situation like the one shown in the figure. There are two paths from the gate ‘1’ to gate ‘b’. The path through ‘a’ is longer, but the other path requires more fanout per stage. This problem is called reconvergent fanout.



It turns out that the short path is the one you want to consider while computing gain per stage in a reconvergent fanout problem because a heavy load on the short path will also increase the delay of the long path. Let the input have 1 unit of cap and the final gate have a size of b times the input. What we want to know is what is the effective logical effort of this combination of gates. That is how does the delay of the path change as we change b. Once we know this, we can just use this ‘effective’ logical effort when we come across this situation and quickly find the fanouts at each stage. The critical path is clearly through inverter ‘a’, and the delay of this path is:

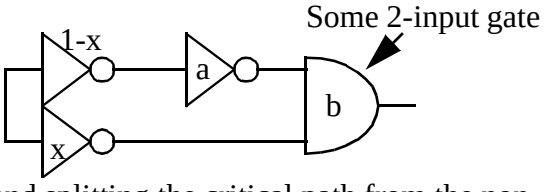
$$t = (a+b)/1 + b/a + 2^* \text{Tparasitic} \qquad \text{(EQ 5)}$$

Since the b load on the first inverter is simply a side load for the problem of sizing ‘a’, the size of ‘a’ is just the geometric mean of gate ‘1’ and ‘b’, which is sqrt(b). Thus the total delay of this path is

$$t = \frac{b+\sqrt{b}}{1} + \sqrt{b} + 2Tparasitic \qquad \frac{dt}{db} = 1 + \frac{1}{\sqrt{b}} \qquad \text{(EQ 6)}$$

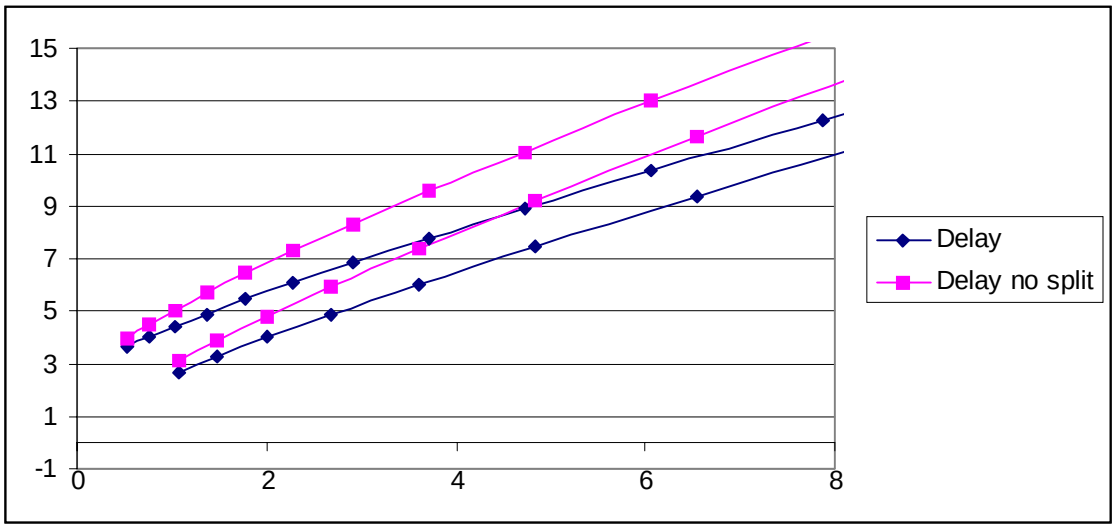
So the logical effort (the slope of the delay with load) is not really a constant, but for reasonable total effort (around 4), the fanout would be around 2.5, and the effective logical effort is 1.6. This is a reasonable number to use for stage efforts from 3 to 8.

Actually this is not the best solution for this problem since we have not matched the delay of the two paths. We have loaded the critical path with directly driving the large gate. This kind of problem often occurs, and you can speed up the circuit by duplicating some logic and splitting the critical path from the non-critical path. So what we should have done is to split the first inverter into two inverters, where the sum of the inverter sizes is equal to 1. In this problem, we have to find both x and a which minimize the total delay of the circuit. This is a little harder to solve, but it is easy to write the equations to solve. At the optimal point the delay of the two paths will be the same, and a will be the geometric mean of b and (1-x). This gives



$$t = \frac{b}{x} + Tpar = 2\left(\sqrt{\frac{b}{1-x}} + Tpar\right) \qquad \text{(EQ 7)}$$

It is easy to solve for ‘b’ as a function of x, so that is what I did. Given x and b, I can find the delay and this gives the plot shown below.



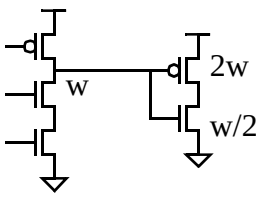
The blue curves (with diamonds) are the result of splitting the first inverter. If the inverter is not split (the first technique) you get the pink (squares) curves. The lower curves are the results you get if you ignore the parasitic delay, the upper curves are the results if the self loading is equal to its gate capacitance.

Splitting the gate is always better than not splitting the gate. When the gate drive is split, the circuit behaves like a single stage gate with logical effort of about 1 (1.1 if you want to be precise) and a large effective parasitic delay (around 4).

2.5 Domino Gates

Working with domino gates is nearly the same as working with static gates. The one issue you need to be careful about is setting the correct effective fanout per stage. For static gates the best buffer is an inverter, so the optimal effective fanout per stage is around 4. If the logical effort of a gate is larger than 1, the fanout of the stage should be reduced.

In domino gates, the LE is often less than 1, so to make the effective fanout 4, the true fanout would need to be greater than 4. This is not correct, since in domino circuits there are better buffers than inverters. The best buffer would look like a domino stage followed by a skewed inverter. The logical effort for the inverter is 5/6, and the domino stage is somewhere between 1/3 and 2/3 depending on the size of the clock transistor (let us assume 2/



3). The LE for the two stages is .55, and the average LE per stage is 0.75. If we used a chain of these gates and tried to find the best fan out per stage, the answer would be that each stage should have an electrical fanout of about 4, (actually 4.4 for dynamic stage and 3.6 for the static gate). At this fanout, the effective fanout (LE * Fanout) of the domino gate is 3, not 4.

Thus in dynamic circuits you should aim for smaller effective fanout per stage than for static logic, which makes the electrical fanout more similar to what you would do with static gates.