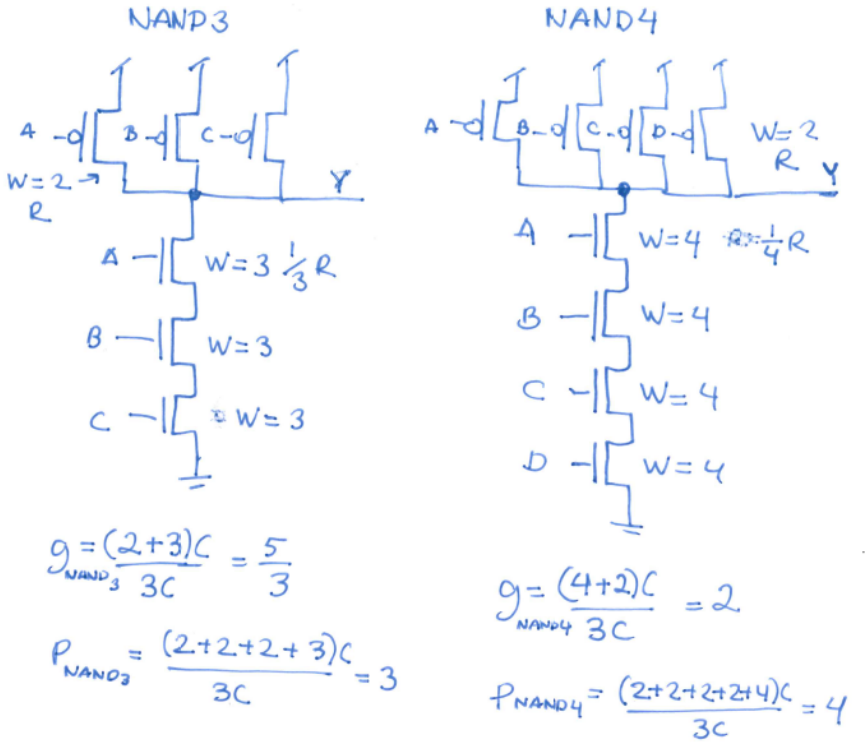
Solution to written exam in **Integrated Circuit Design MCC091**

Tuesday August 22, 2016, at 8.30-13.30 at Lecture halls, Hörsalsvägen

1) Logical effort, parasitic delay, layout

a) Here are the two circuit diagrams with scaled transistors and the corresponding g and p values. We assume the inverter has $p_{inv}=1$. There is only one logical effort, g, value per gate since all inputs are connected to two transistors that are scaled the same.



$g_{NAND3} = \frac{(2+3)C}{3C} = \frac{5}{3}$

$p_{NAND3} = \frac{(2+2+2+3)C}{3C} = 3$

$g_{NAND4} = \frac{(4+2)C}{3C} = 2$

$p_{NAND4} = \frac{(2+2+2+2+4)C}{3C} = 4$

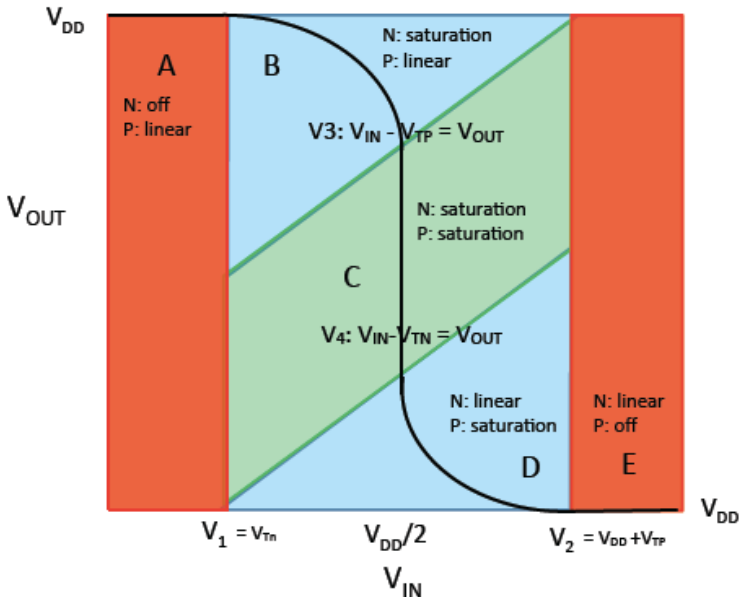
b) The layout with the minimum number (2) of p-diffusion areas connected to the output:

c) For the 4-input NAND gate, with path resistance R in both n and paths, there are two p-diffusion areas with width 2 connected to the output and 1 n-diffusion area with width 4. For a CMOS inverter with resistance R, there are diffusion areas of width 2+1 connected to the output (corresponding to pinv=1). So pNAND 4 = (2+2+4)/3 = 8/3 which is much less than 4 which we got in task a).

d) We need to find and expression $d = p_{AND} + g_{AND}h_{AND}$ for the entire AND gate, where p_{AND} is the part of the delay that does no depend on the load capacitance. We have for the entire gate $d_{AND} = g_{NAND}h_{NAND}+p_{NAND} + p_{INV}h_{INV}+p_{INV}$. The electrical effort for the NAND gate, h_{NAND} is 1, because its load capacitance is the same as its input capacitance. The electrical effort for the inverter is $C_{LOAD}/6$. Thus we find = $d_{AND} = 2*1+8/3+1*(C_{LOAD}/6)+1=2+8/3+1+1*(C_{LOAD}/6) = 4.67+1*(C_{LOAD}/6)$. Due to the scaling in the layout, the input capacitance for the NAND gate is the same as the one for the inverter and thus $h_{AND} = h_{inv} = C_{LOAD}/6$. So therefore the solution is $p_{AND} = 4.67$ and $g_{AND} = 1$.

2) Inverter static charcteristics

a) See figure below:



b) The general expression for the switching voltage is

$$V_{sw} = \frac{V_{DD} + V_{tp} + V_{tn}\sqrt{\frac{kn}{kp}}}{1 + \sqrt{\frac{kn}{kp}}} = \frac{V_{DD}}{1 + \sqrt{\frac{kn}{kp}}} + \frac{V_{tp} + V_{tn}\sqrt{\frac{kn}{kp}}}{1 + \sqrt{\frac{kn}{kp}}}.$$

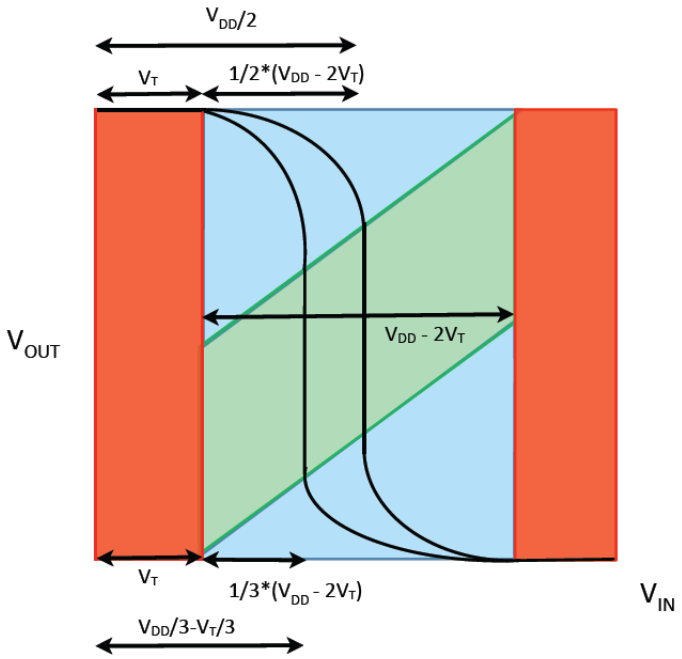
This equation one can easily derive from the quadratic MOS transistor current equations. (There are also other formulations of the equation that are equivalent). We see that with $V_{tp} = -V_{tn} =$ the expression simplifies to

$$V_{sw} = = \frac{V_{DD}}{1 + \sqrt{\frac{kn}{kp}}} + V_{tn} - V_{tn}\frac{2}{1 + \sqrt{\frac{kn}{kp}}} = V_{tn} + \frac{V_{DD} - 2V_{tn}}{1 + \sqrt{\frac{kn}{kp}}}.$$

With $k_n = k_p$ we have $V_{sw} = V_{DD}/2$ as expected. When we make the nMOS transistor four times stronger than the pMOS transistor, that is, $k_n = 4\ k_p$, we get:

$$V_{sw} = V_{tn} + \frac{V_{DD} - 2V_{tn}}{3}.$$

For a graphical interpretation of these expressions see the figure below:

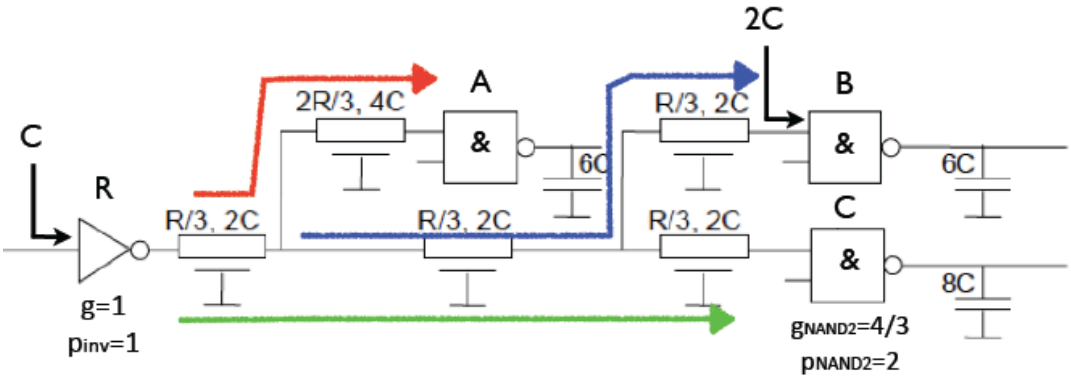


c)

The reason is that in the saturation region the transistor cannot give infinite current for a very small voltage change at the output (drain-source voltage). Or put another way, the transistors do not have infinite output resistance in the saturation region.

3) Wire and gate delay, logical effort

a) In this circuit there are three paths to the clock gaters A, B and C (red, blue and green in the figure below).



We need to model the wire segments in these paths using the pi model and calculate the wire delay using Elmore's model. However, the clock skew is the **difference** in delay. Those parts of the delays that are the same for all three paths we do not have to calculate. The Elmore delay can be calculated as the delay due to the main path plus the delay due to branches. In this circuit the main path due to the wire segments to the inputs A, B, and C all have R and 6C and the input capacitances are also all the same: 2C. Thus, we do not have to calculate the main-path delay. However, the branch delays differ. There are three branches and each path incorporates two of them:

Branch RC delay due to red branch: $bd_A = \left(\mathrm{R} + \frac{\mathrm{R}}{3}\right) \times 6\mathrm{C} = \frac{24}{3}\mathrm{RC}$

Branch RC delay due to blue, green branches: $bd_B = bd_C = \left(\mathrm{R} + \frac{2\mathrm{R}}{3}\right) \times 4\mathrm{C} = \frac{20}{3}\mathrm{RC}$

Total branch RC delay to node A is $bd_B + bd_C = 2 \times \frac{20}{3}\mathrm{RC} = \frac{40}{3}\mathrm{RC}$

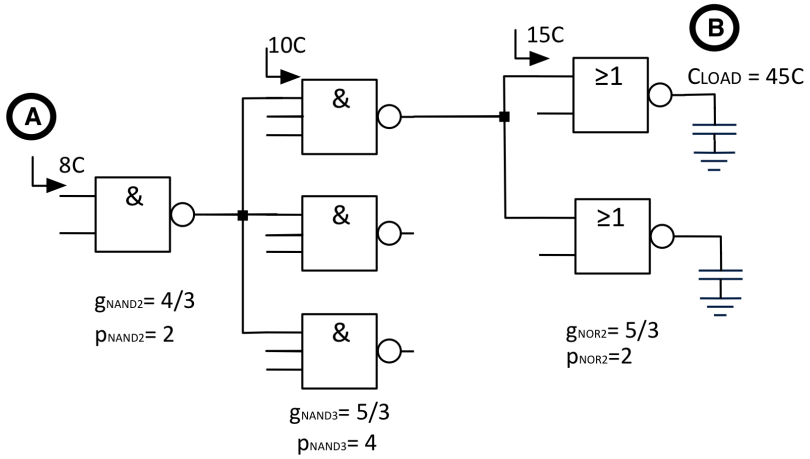
Total branch delay to nodes B and C are the same: $\left(\frac{24}{3}+\frac{20}{3}\right)\mathrm{RC}=\frac{44}{3}\mathrm{RC}$

So the **clock skew is $\mathbf{0.7\times\frac{4}{3}RC}$** and the delay to clock gaters B and C are the longest one, whereas the delay to gater A is shorter.

b) The difference in load capacitance between clock gaters B and C is what will cause clock skew between the outputs of gaters B and C. That difference is 2C. The delay to the gate inputs will not change.

Gaters A, B and C are identical. The logical effort, *g*, is defined as "*the ratio of the input capacitance to that of an inverter that can drive the same current*" or, in other words, the same effective resistance. That is, *g*C corresponds to R. Here we have that $C_{NAND}=2\mathrm{C}=g\frac{3}{2}\mathrm{C}$ and thus $R_{NAND}=\frac{2}{3}R$ .So the resulting clock skew is 0.7 x 2/3R*2C = **is $\mathbf{0.7\times\frac{4}{3}RC}$**. We can of course also find that result directly without calculating the R_{NAND} value explicitly.

4) Path logical effort, path delay (See also example 3.13, page 123 in Weste and Harris.)



a) The path logical effort G = g_1*g_2*g_3. Here G = $\frac{4}{3}*\frac{5}{3}*\frac{5}{3}$. We have path delay F = G*H*B. So, for path delay we also need path branching effort and path electrical effort: B = 3*2 and H = $\frac{45\,\mathrm{C}}{8\,\mathrm{C}}$. We thus have

$$\mathrm{F}=\frac{4*5*5*3*2*45}{3*3*3*8}=125$$

b) F = 125 = 5*5*5. Since we have three stages in the path, the optimal stage effort f_{opt} is $\sqrt[3]{\mathrm{F}}=5$. That is, f= g*h should be 5 for each stage. From the logical efforts we calculate h_3 = 3, h_2 =3 and h_1 = 15/4. The resulting input capacitances of the gates are shown above. Remember the branching when calculating the capacitances!

c) The normalized delay is N*f_{opt} + P = 3*5 + 2 + 4 + 2 = 23. That means that the total delay is 23τ. The FO4 delay is 5τ when p_{inv} is 1, so the delay is 4.6 FO4 delays. That is around 115 ps in our 65-nm process.

5) Power

a) In general we have $f_{clk} = 1/T \sim 1/RC$ where $0.7RC$ is the time constant, τ, for the transistors. In the low-throughput (that is, *min*) case it is enough to use a clock frequency f_{clkmin} that is only ¼ of f_{clkmax}. To calculate how much we can lower V_{DD} in that case, we should find the V_{DD} that corresponds to an RC_{min} that is $4RC_{max}$ (where subscripts *max* and *min* refer the fclk cases). The transistor effective resistance, *R,* is defined as V_{DD}/I_{DSAT}. Assuming we can use the quadratic current equations, we have that I_{DSAT} is proportional to $(V_{DD}-V_T)^2$. The transistor gate capacitances are the same regardless of V_{DD} so therefore we need to solve the simpler equation: $R_{min} = 4R_{max}$ (where again *max* and *min* stands for the *fclk* cases). Thus, we have:

$$\frac{V_{DDmin}}{(V_{DDmin}-V_T)^2} = 4\frac{V_{DDmax}}{(V_{DDmax}-V_T)^2}$$

Using $V_T = 0.1V_{DDmax}$, and dividing with V_{DDmax} we can express this equation as:

$$\frac{x}{(x-0.1)^2} = 4\frac{1}{(1-0.1)^2}$$

where x is the ratio V_{DDmin}/V_{DDmax}. The solution is $x = 0.374$ which gives $V_{DDmin} \approx 0.45$ V. This low a V_{DD} value is maybe not entirely realistic, but since V_T is so low could work.

b) The dynamic power is $\alpha f C V_{DD}^2$. The ratio between the new (lower) dynamic power and the original can then be expressed as:

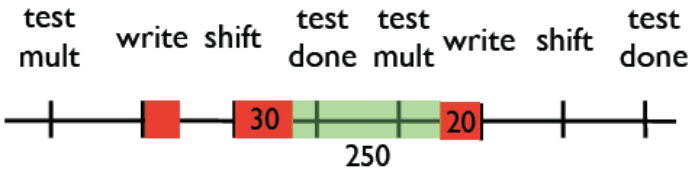
$$P_{dyn\ ratio} = \frac{1.05(0.25(V_{DDmax}^2 f_{clkmax}) + 0.75(V_{DDmin}^2 f_{clkmin}))}{V_{DDmax}^2 f_{clkmax}}$$
$$= \frac{1.05(0.25 + 0.75\times 0.374^2 \times 0.25)}{1} \approx 0.29$$

So almost all the dynamic power comes from the high-V_{DD} case.

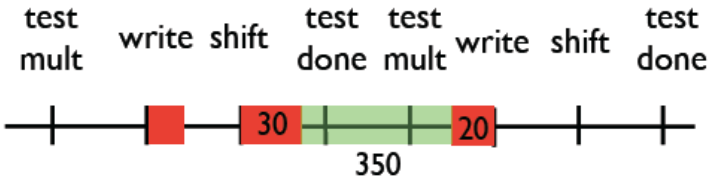
c) The static power due to subthreshold leakage is $P_{subleakage} = V_{DD}\, I_{sub.}$ In addition the subthreshold current depends on V_{DD} in that the current increases with higher V_{DD}. Both these dependencies makr the current decrease when V_{DD} decreases.

6) Adders

a) Both 16-bit adders have a worst-case delay of 250 ps. The figure below shows the unrolled steps in the control logic of the multiplier in the worst-case situation where every step results in a write to the product register. From the figure it is clear that three steps are completed in 300 ps. Thus, one step takes 100 ps and the resulting clock frequency is 1/(100ps) = 10 GHz. An 8-bit multiplication takes 1+8*4 = 33 steps which is then 3300 ps or 3.3 ns.



b) We will use the Sklansky adder because it will have a shorter worst-case delay. From the table, and our knowledge about prefix adders, we find that the expression for the worst-case delay is 50ps + 2logn * 50 ps. For n=64 we then have the worst-case delay as 50 + 6*50 = 350 ps. The ripple-carry adder has much longer delay since its delay grows linearly with n. The figure below show the unrolled steps in the control logic of the multiplier where it becomes clear that three steps are completed in 400 ps. Thus, one step takes 133 ps and the resulting clock frequency is 1/133 ps = 7.5 GHz. A 32-bit multiplication with worst-case data lasts for 1+32*4 = 129 steps which each takes 133 ps. All in all then 17.2 ns.



c) See figure below from Hennessy and Patterson Computer Organization and Design. By shifting the product rather than the multiplicand we can use a 32-bit adder rather than a 64-bit adder. The 32-bit Sklansky adder has a worst-case delay of 300 ps. The cycle time is then 350/3 = 117 ps. The time for one multiplication to complete with worst-case data is then 129*117 ps= 15.1 ns. So the gain in calculation delay is not that great. One saves space and power too though.

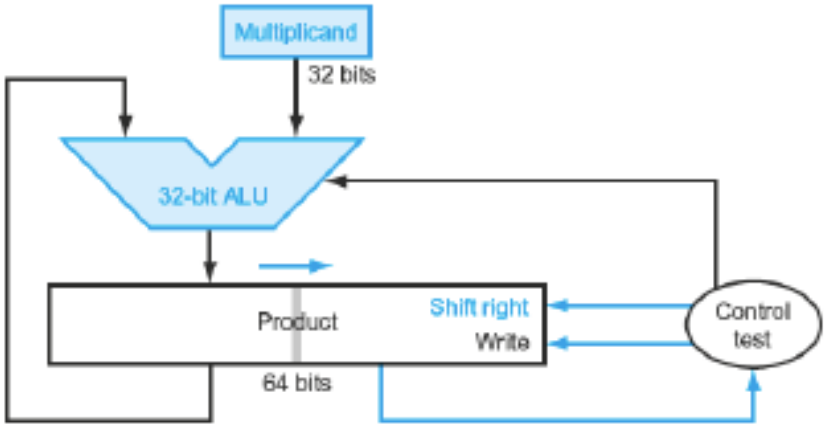


FIGURE 3.5 Refined version of the multiplication hardware. Compare with the first version in Figure 3.3. The Multiplicand register, ALU, and Multiplier register are all 32 bits wide, with only the Product register left at 64 bits. Now the product is shifted right. The separate Multiplier register also disappeared. The multiplier is placed instead in the right half of the Product register. These changes are highlighted in color. (The Product register should really be 65 bits to hold the carry out of the adder, but it's shown here as 64 bits to highlight the evolution from Figure 3.3.)