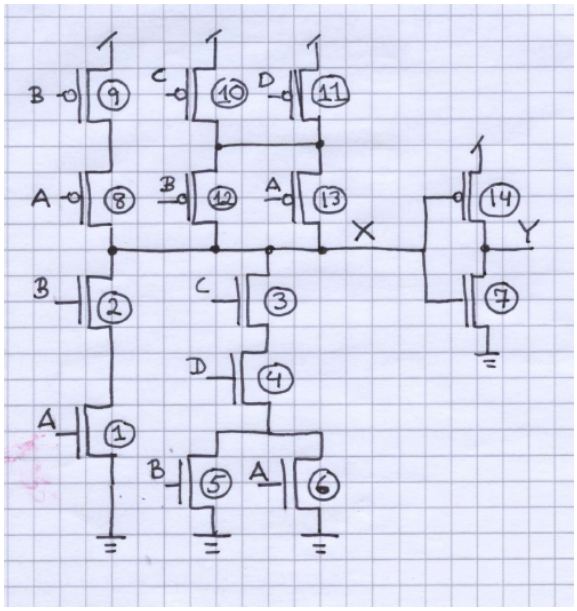
# Solution **Integrated Circuit Design MCC091** Monday January 5, 2015

## 1. Layout, static CMOS logic

a) For reference we number the nMOS transistors in the layout 1-7 from left to right and the pMOS transistors 8-14 from left to right. The output of the compound gate we name X and the output of the inverter Y. The corresponding transistor schematics is shown below; the numbers of each transistor is to the right of that transistor:



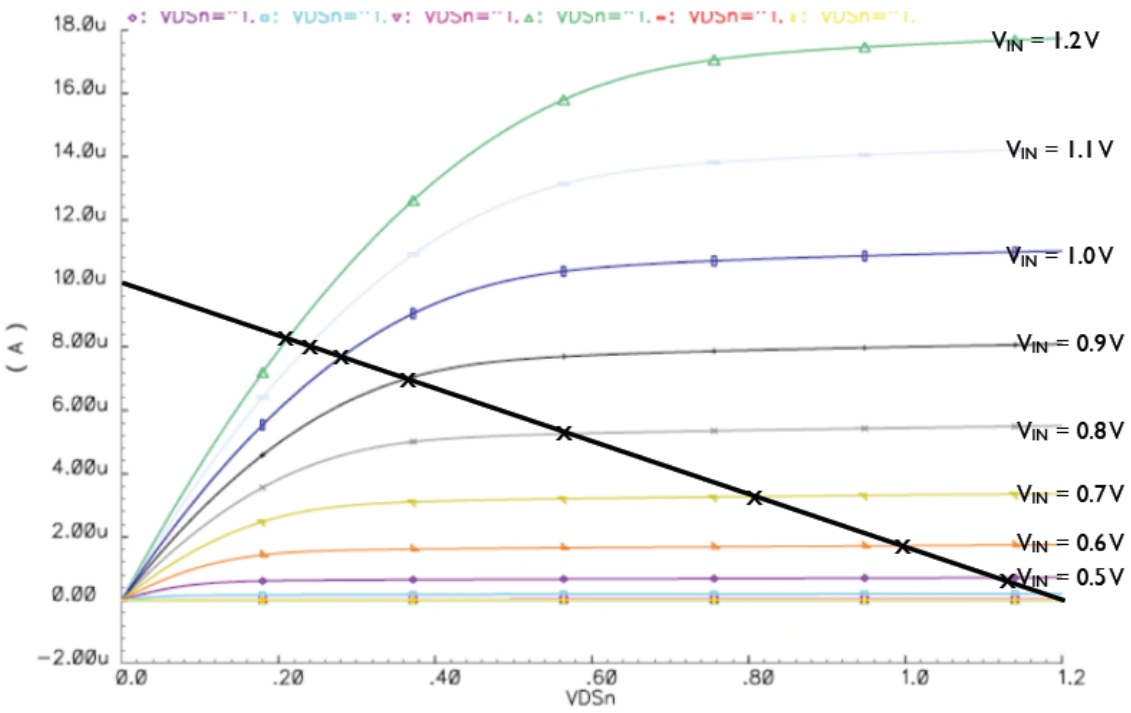

b) $Y = AB + (A+B)CD$

It is easiest to find the function from the n-net of the compound gate. Since its output, X, is inverted to form Y, the n-net gives the function for Y directly.

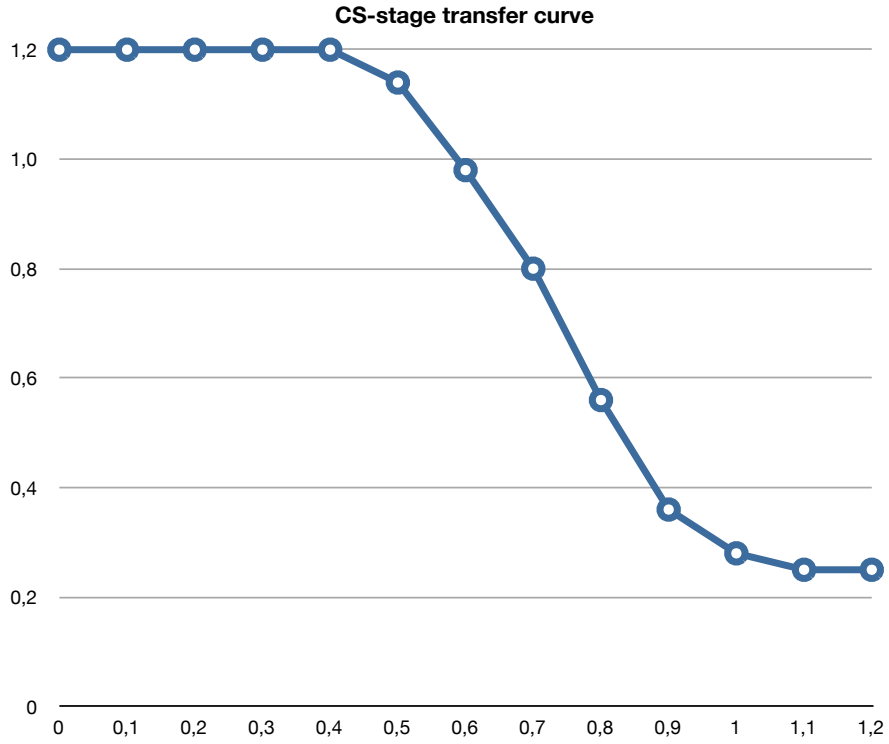
Note that this is the logic function for the 2-bit generate gate: $G_{2:1}$ where $G_{2:1} = A_2B_2 + (A_2 + B_2)A_1B_1$. See figure 10.36 in Weste & Harris.

## 2. Amplifier large and small-signal analysis

a) The resistor current we find by applying Ohm's law, U = RI, to the resistor. The two easiest points are the max and min points: U=0 and U = $V_{DD}$. $U_{RD}$=0 V gives $I_{RD}$ = 0 µA. $U_{RD}$=$V_{DD}$=1.2 V gives $I_{RD}$=$V_{DD}$/$R_D$= 1.2 V/120 kΩ = 10 µA. If all the supply voltage is across the resistor there will be 0 V over the transistor and conversely, with no voltage over the resistor all the supply voltage will be over the transistor, Thus, the two points for the load line are: $V_{DS}$=0V, $I_{DS}$=10 µA and $V_{DS}$=1.2 V, $I_{DS}$=0 µA. The points and load line are shown in the figure below:

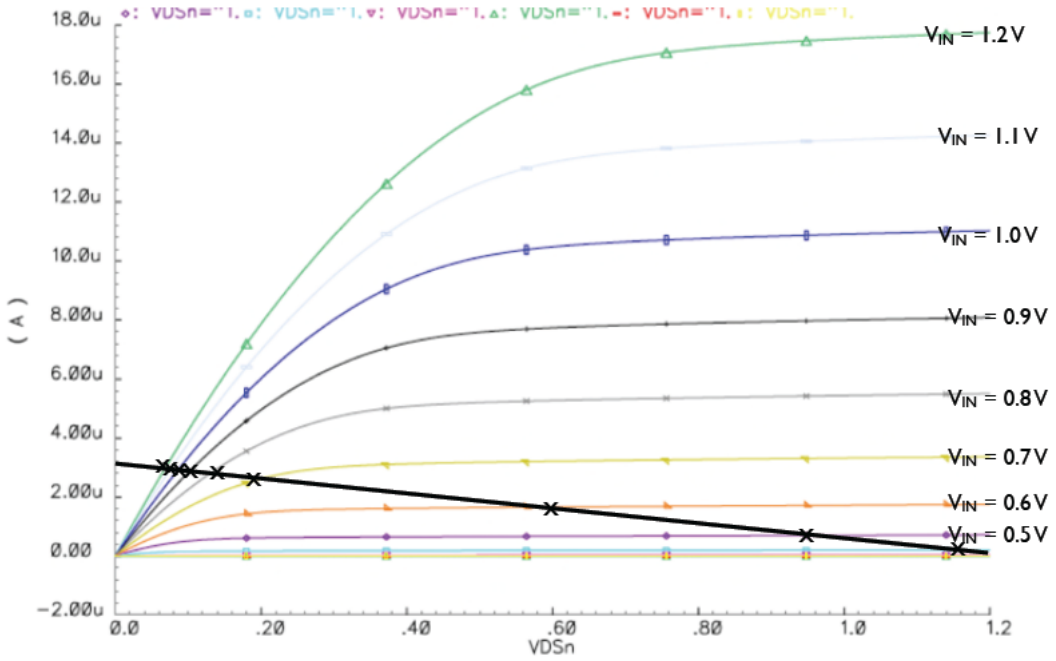



b) Where the load line intersects one of the current curves the point for the input voltage that corresponds to that input voltage. The points are marked in with crosses in the voltage diagram above. Above each point is the corresponding $V_{IN}$ (that is $V_{GS}$) value. The corresponding $V_{OUT}$ vs $V_{IN}$ diagram looks like this.




c) We can get an estimate from drawing the line in curve, or since we have so few points we can use the approximation between the two data points at $V_{IN}$ = 0.7 and 0.8 that gives us 0.55-0.8 / 0.8-0.7 = -2.5

d) The steepest part of the curve is close to the switching voltage, that is, when $V_{IN} = V_{OUT}$. Here we want $V_{SW} = 0.6$ V. Therefore, we have to find the load line that crosses the 0.6-V $I_{DS}$ curve when $V_{OUT} = 0.6$ V (the orange curve). In this case this load line corresponds to a maximum $I_{DS}$ current of around 3.2 µA. That corresponds to a resistance of $R_D = V_{DD}/I_{DSmax} = 1.2$ V / 3.2 µA = 375 kΩ. We do not have to have exactly that current so 1.2V /3.6 µA = 330 kΩ would work. Or just tripling the resistor to 360 kΩ would also work. Here is the diagram again with that load line:




The expression for the gain is $-g_m R_D$, so if we triple the resistance value, one could easily think that we should get triple the gain; that it is to around -7.5. However, the transconductance, $g_m$, is not the same for another value of $V_{GS}$. So we really should repeat the procedure from c) above, not just recalculate from the previous gain value. From the 0.5 to the 0.7 curves we get approximately: (0.21 - 0.95)/ (0.7-0.5) = -0.73/0.2 = -3.65. That is, the gain is higher but not a much as our first guess indicated.

## 3. Design and tapering

a) We know that for minimum delay each stage should have the same effort and that a stage effort, f, of 4 is good to minimize the delay. 1024 is $2^{10}$, which is also 4*4*4*4*4, so 5 inverters in all and thus four inverters in the box is a good solution.

b) The delay is p+gh or p+f. We have f=4 (see a) above) and $p_{inv} = 0.5$ (from problem statement). So the normalized delay, d, is 5*(4+ 0.5) = 22.5 and with $\tau = 4$ ps we have a delay of $d*\tau = 22.5*4 = 90$ ps.

c) The dynamic power consumption due to recharging of the capacitances is $P_{dyn} = \alpha f C_{tot} V^2$ where α and f are given in the problem statement. V in the equation is $V_{DD}$. The remaining factor is $C_{tot}$, the total switched capacitance, which we have to determine. $C_{in}$ for the inverter is not given in the problem but we can express the dynamic power in $C_{in}$. How many times $C_{in}$ is the switched capacitance?

$$C_{tot} = C\,(1 + 4.5 + 4.5f + 4.5f^2 + 4.5f^3 + 4.5f^4)$$

if we count the also the input capacitance of the first inverter. We can also write the capacitance as

$$C_{tot} = C_{parastictot} + C_{inputtot} = 1/2 + 2 + 32 + 128 + (1 + 4 + 16 + 64 + 256 + 1024)$$

Either way the dynamic power is 0.25 * 20*$10^6$ * $1^2$* 1527.5 *$C_{in}$ where $C_{in}$ should be a few fF. Even though we do not have the exact number for the input capacitance we can check that the numbers are reasonable using our previous knowledge of the 65-nm process. If we assume that $C_{in}$ for the first inverter is a little less than 4 fF we can approximate 1527.5 *$C_{in}$ with 6000 fF = 6 pF. With this capacitance we arrive at

$P_{dyn} = 50 * 10^6 * 6 * 10^{-12} = 300\ \mu W$

which seems an entirely reasonable result. Either reply is regarded as correct.

d) It is incorrect because we neglect to take into account that if we have very large fanout for a gate the switching will be very slow at the gate output because its current is too small to charge the capacitance quickly. Then the n-net and p-net transistors in the gate will both be conducting at the same time and quite a large short-circuit current will flow during the switching.

e) **Bonus question** It would be better to remove one inverter than to add one inverter since the dynamic power will be lower while the delay would only be negligibly longer.

**4.** **Wire delay** The clock skew is the difference in delay. Since the problem only asks for these differences it is not necessary to calculate the entire delays in detail. The capacitance differs only at the leaves; the wires are the same for all four leaves. We will have to take the delay for the branches into account but we can calculate the main paths and the branches separately and then add them.

**Main-path delay**

Main-path delay = delay due to wire capacitances + total path resistance * $C_{leaf}$. The total path resistance to all leaves is (1+16+32)/16 R = 49/16 R. We then identify the common part as the shortest delay of the four, which is the one to terminal B:

**A:** Main delay = Wire part + 49/16 R * 18C = *cd12* + RC/16 (49*18) = c*d12p* + RC/16 (49*4)
**B:** Main delay = Wire part + 49/16 R * 14C = *cd12* + RC/16 (49*14) = *cd12p*
**C**: Main delay = Wire part + 49/16 R * 22C = *cd12* + RC/16 (49*22) = *cd12p* + RC/16 (49*8)
**D:** Main delay = Wire part + 49/16 R * 18C = *cd12* + RC/16 (49*18) = *cd12p* + RC/16 (49*4)

**Branch capacitances**

To the main delays we have to add the branch delays. Also here it is only the leaf capacitances that differ. Again we identify the smallest capacitance as the common part.

The entire AB sub tree has the capacitance = *cap12* + 32C = *cap12p*
The entire CD sub tree has the capacitance = *cap12* + 40C = *cap12p* + 8C
The A subtree has the capacitance = *cap2*+ 14C + 4C = *cap2p* + 4C
The B subtree has the capacitance = *cap2* + 14C = *cap2p*
The C subtree has the capacitance = *cap2* + 14C + 8C = *cap2p* + 8C
The D subtree has the capacitance = *cap2*+ 18C + 4C = *cap2p* + 4C

**Branch delays**

The resistance to the subtree AB/CD is 1/16 R, and to the leaf subtrees (1+16)/16=17/16 R:

**A:** Branch delay for leaf A = delay due to subtree CD + delay due to subtree B =
R/16 (*cap12p* + 8C) + 17/16 R (*cap2p*) = *commonbd* + RC/16 * 8
**B:** Branch delay for leaf B = delay due to subtree CD + delay due to subtree A =
R/16 * (*cap12p* + 8C) + 17/16 R * (*cap2p* + 4C) = *commonbd* + RC/16 (8 + 4*17)

**C:** Branch delay for leaf C = delay due to subtree AB + delay due to subtree D =
R/16 * (*cap12p*) + 17/16 R * (*cap2p* + 4C) = *commonbd* + RC/16 * 4*17

**D:** Branch delay for leaf D = delay due to subtree AB + delay due to subtree C =
R/16 * (*cap12p*) + 17/16 R * (*cap2p* + 8C) = *commonbd* + RC/16 * 8*17

**Total delays**

Now we can write the entire delays for each of the four leaves. To simplify the equation we call the common part of the delay *cd*; *cd* is *common12p* + *commonbd*:

delayA = *cd* + RC/16 ( 4*49 + 8) = *cd* + RC/4*(49+2) = *cd* + RC/4 * 51
delayB = *cd* + RC/16 (8 + 4*17) = *cd* + RC/4*(2+17) = *cd* + RC/4 * 19
delayC = *cd* + RC/16 (8*49 + 4*17) = *cd* + RC/4*(2*49+17) = *cd* + RC/4 * 115
delayD = *cd* + RC/16 (4*49 + 8*17) = *cd* + RC/4 (49+2*17) = *cd* + RC/4 * 83

As expected leaf B has the shortest delay and leaf C the longest one. The interesting result here is that the delays for leaves A and D are not the same even though their leaf capacitances are the same. This is due to the branch delays. This result shows that one has to be careful in balancing the tree to make all the delays the same.

Now all we have to do is find the delay differences for all pairs of signals. We use the signal in the column as the reference.

| Clock skew *RC/4 Reference signal | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 51-19 = **32** | 51-115 = **-64** | 51-83 = **-32** |
| B | **-32** | 0 | 19-115 = **-96** | 19-83 = **-64** |
| C | **64** | **96** | 0 | 115-83 = **32** |
| D | **32** | **64** | **-32** | 0 |

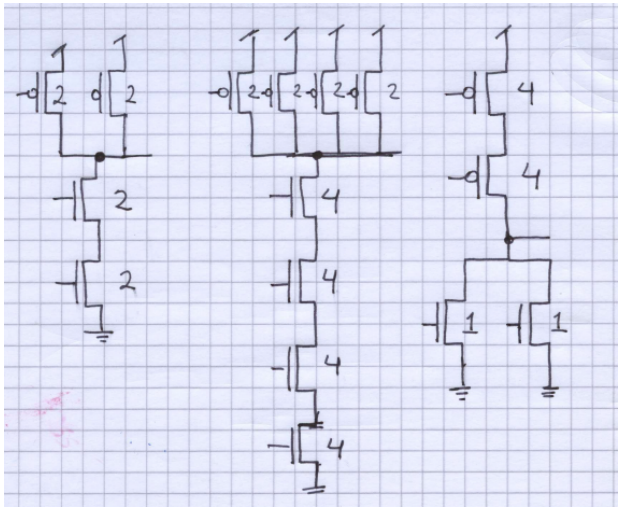
Or we can we can multiply it out with the factor 1/4:

| Clock skew in *RC Reference signal | A | B | C | D |
|---|---|---|---|---|
| A | 0 | **8** | **-16** | **-8** |
| B | **-8** | 0 | **-24** | **-16** |
| C | **16** | **24** | 0 | **8** |
| D | **8** | **16** | **-8** | 0 |

It is OK to give the clock skews without signs, however it is clearer with the signs when one has multiple clock skews.

## 5. Logical effort

a) We assume that pMOS transistors are twice as wide as the nMOS transistors as usual. The schematics for three gates (2- and 4-input NAND and 2-input NOR gate) and their transistor sizes for determining the logical efforts are shown in the figure below:




An inverter with the pMOS transistor twice as wide as the nMOS transistor is assumed to have the parasitics $p_{inv}$.

| Gate | g, logical effort<br>input capacitance for particular input / reference inverter input capacitance | p, parasitics<br>total width of transistors connected to output node / width connected to inverter output * $p_{inv}$ |
|---|---|---|
| 2-input NAND | (2+2)/3 = 4/3 | (2+2+2)/3*$p_{inv}$ = 2 $p_{inv}$ |
| 2-input NOR | (1+4)/3 = 5/3 | (4+1+1)/3 = 2 $p_{inv}$ |
| 4-input NAND | (2+4)/3 = 2 | (2+2+2+2+4)/3*$p_{inv}$= 4 $p_{inv}$ |
| inverter | 1 | $p_{inv}$ |

(1) $d_1 = (gh+p)_{NAND4} + (gh + p)_{inv} = 2*3/6+4\ p_{inv} + 1*1 + p_{inv} = 2 + 5\ p_{inv}$
(2) $d_2 = (gh+p)_{NAND2} + (gh + p)_{NOR2} = 4/3*5/4 + 2\ p_{inv} + 5/3*3/5 + 2\ p_{inv} = 5/3+1 + 4\ p_{inv} = 8/3 + 4\ p_{inv}$

So it is not obvious which solution is better. If $p_{inv}$ is large solution (2) is better, if $p_{inv}$ is small solution (1) is better. The cross-over point is when $2 + 5\ p_{inv} = 8/3 + 4\ p_{inv}$ that is when $p_{inv} = 2/3$.

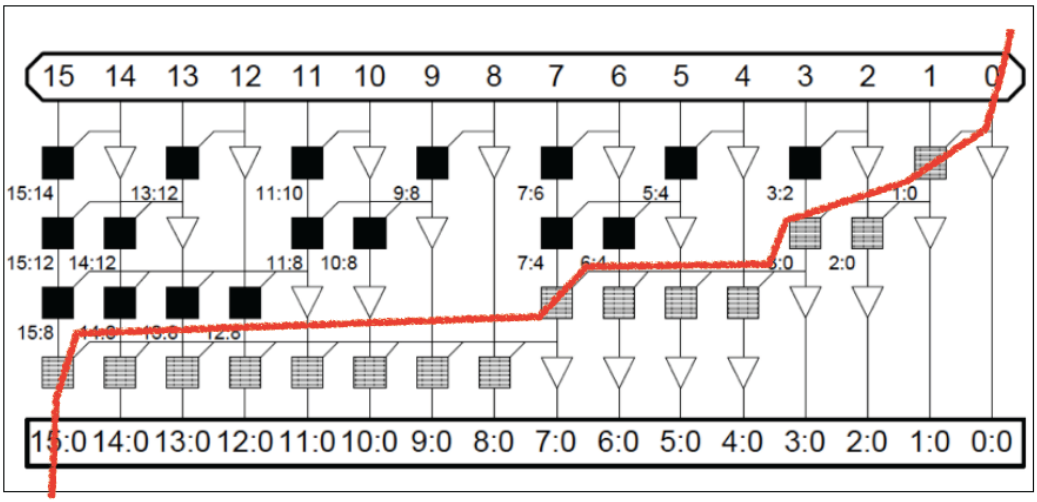
b) When we make a model of the entire cell of the form gh + p we need to have **p** as the total part of the delay that does not depend on the load capacitance. If cin is the input capacitance of the gate then the fanout **h** is cload/cin. Then the logical effort, **g,** is to be the factor with which we should multiply the fanout to get the right delay.

| Gate | Total delay (with h defined for the output gate) | Cin entire stage | Cin output gate | **g** logical effort for entire gate is g output stage * Cin entire / Cin output stage when **h** is defined for the entire gate | **p,** the part of the delay that does not depend on the fanout |
|---|---|---|---|---|---|
| (1) | 1 + 5$p_{inv}$ +1*h | 6 | 3 | 1 * 6 /3 = 2 | 1 + 5$p_{inv}$ |
| (2) | 1 + 4$p_{inv}$ + 5/3*h | 4 | 5 | 5/3 * 4 / 5 = 4/3 | 1 + 4 $p_{inv}$ |

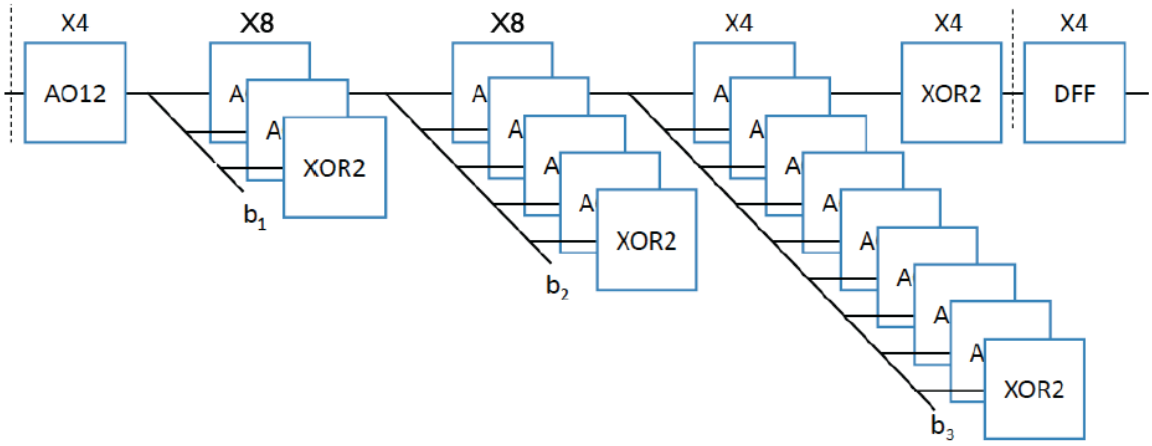
So with $pi_{nv} = 1$ we have in case (1): 6 +2*h and for (2): 5 + 4/3*h with h defined for the entire gate (1) or (2), respectively.

**6.**

a) The critical path is shown below.

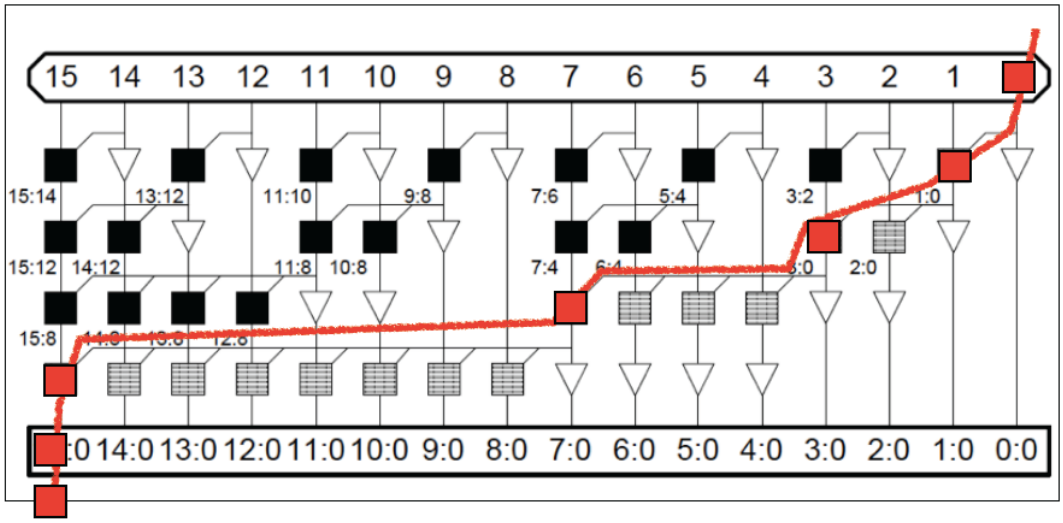



as

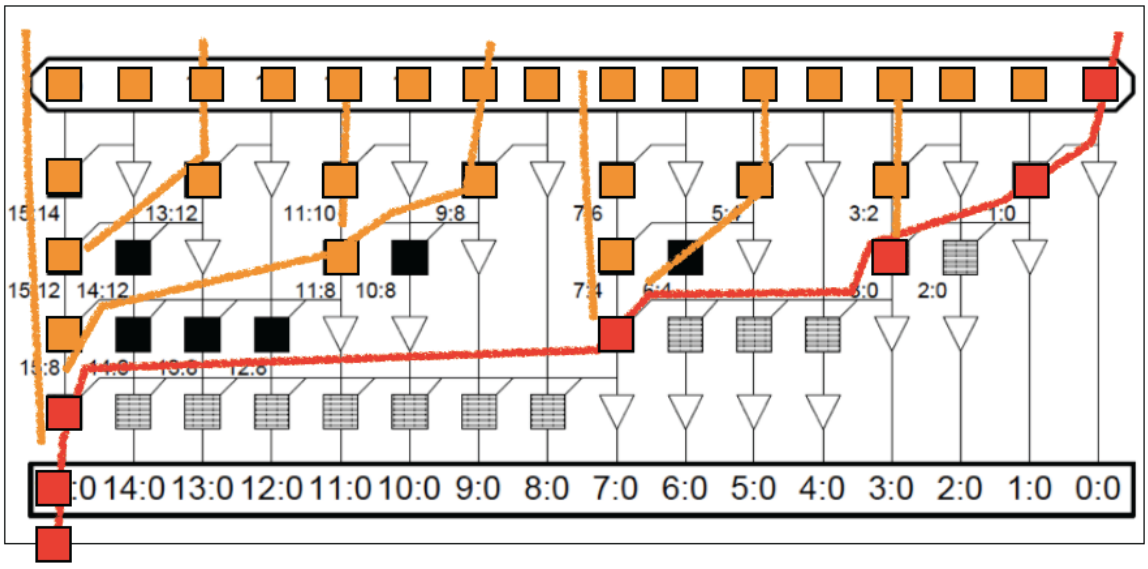



b) Above you see the figure of the critical path used by Kjell in his lecture. The delay of all the parasitics for all 6 stages above plus the setup EXOR gate that is not shown is 6+8+8+8+8+6+6= 50 τ. The Stage effort, f = gh, is 2 in each stage in the tree due to the branching. (See the derivation in lecture adder 7 for details). This is if we do not include the XOR gates that load the AO gates.
For simplicity we assume this stage effort also for the EXOR gates and DFF gates even though they probably have slightly lower stage efforts. The fanout-dependent part of the delay is then roughly 7*f = 14 τ. In total the delay is around 50 + 14 = 64 τ where the second part may be a bit smaller.
We find that the part that depends on the fanout is much smaller than the part that that does not depend on the fanout.

c) In b) we saw that the non-fanout dependent part of the delay dominates. So to analyze the adder structure roughly it would make sense to mainly look at the number of cells in the paths, and not so much on the fanout. From the figure above we see that the critical path has four AOI cells. Below the cells in the critical path are shown in red:

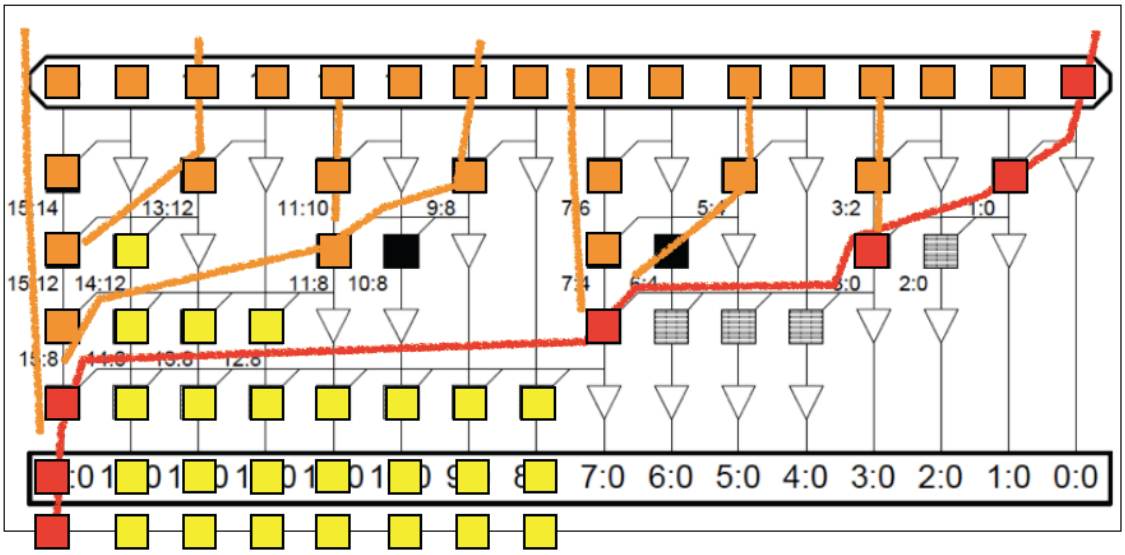



This is the path we want to speed up by using the faster library for the seven red cells. The critical path has 4 AOI cells in the tree. However, there are many additional paths with 4 AOI cells that lead to the bit15 output. Since the constant part is dominating all these paths have to also be implemented in the fast library. The figure below shows these paths and all their cells in orange:




If all the red and orange cells above are implemented with cells from the fast library, bit 15 should be generated in around ¾ of the time for the slower library.

However, there are several other paths in the Sklansky tree that also contain 4 AOI cells. If we only speed up the red and orange cells one of these other bits will have a longer delay that does bit15. The bits that are relevant are bits 8 through 14. In the figure below the cells of these paths are shown in yellow:




Now all the remaining paths have 3 or fewer AOI gates and their tree delays should be equal or lower than that of those with 4 AOI gates since their delays are ¾ as long and, except for bit 7, they all have lower fanout. In addition, all of these paths have some AOI gates that are in the fast library. However, we would have to make a more careful analysis of bits 0-7. There are now only 5 cells in the tree that are not in the fast library + the sum generation for bits 0-7, so maybe it would be better to implement the entire adder in the faster library.