COMS12200 lab. worksheet: week #4

Although some questions have a written solutions below, for others it makes more sense to experiment in a hands-on manner: the Logisim project

http://www.cs.bris.ac.uk/home/page/teaching/material/arch_new/sheet/lab-4.s.circ

supports such cases.

S4. a Recalling that NAND means "NOT-AND" and so the NAND of x, y and z is given by

$$\neg(x \wedge y \wedge z),$$

we can write the required truth table as follows:

x	y	z	$x \wedge y \wedge z$	$x \mathbin{\overline{\wedge}} y \mathbin{\overline{\wedge}} z$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

b If you think about AND to start with, we can see that

x	y	z	$x \wedge y \wedge z$	$t = x \wedge y$	$t \wedge z$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	0
1	1	1	1	1	1

and hence say

$$x \wedge y \wedge z \equiv (x \wedge y) \wedge z.$$

Put another way, for AND we *can* realise the 3-input version using two 2-input versions. Replacing the AND gates with NAND gates and given

x	y	$x \mathbin{\overline{\wedge}} y$
0	0	1
0	1	1
1	0	1
1	1	0

we can see that

x	y	z	$\neg(x \wedge y \wedge z)$	$t = x \mathbin{\overline{\wedge}} y$	$t \mathbin{\overline{\wedge}} z$
0	0	0	1	1	1
0	0	1	1	1	0
0	1	0	1	1	1
0	1	1	1	1	0
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	0	0	1

So here, the same fact does not hold, i.e.,

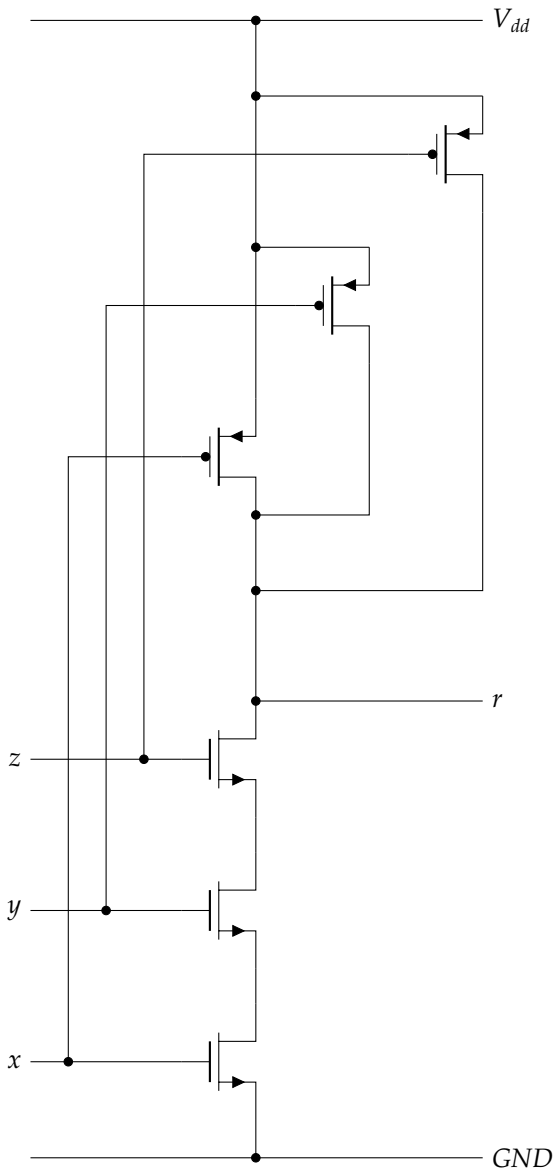
$$\neg(x \wedge y \wedge z) \not\equiv (x \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} z,$$

and so we cannot realise a 3-input NAND simply by using 2-input NANDs. As an aside, note that parenthesising the expression in the other direction means inspecting $x \mathbin{\overline{\wedge}} (y \mathbin{\overline{\wedge}} z)$, but gives a similar

result; the reason either fails to produce the result we require is basically because NAND is not associative (whereas AND is).

A fast(er) way to answer this question is simply to come up with a counter-example where the two differ (which implies they cannot be equivalent); this relates somewhat to the discussion of SAT within the lecture(s). So for example, stating that for $x = 0$, $y = 0$ and $z = 1$ the 3-input NAND produces 1 whereas the combination of 2-input NANDs produces 0 is more or less enough.

c This question is easier than it sounds; basically we just add extra transistors (one P-MOSFET and one N-MOSFET) to implement a similar approach (highlighted in the next question). Diagrammatically, the result is as follows:



d If you try to generalise the strategy used for the 2- and 3-input NAND gates, the basic idea is that we need

- a pull-up network of n P-MOSFET transistors that all operate in *parallel* (so if *any* is connected, the result is 1 due to the connection with V_{dd}), and
- a pull-down network of n N-MOSFET transistors that all operate in *series* (so if *all* are connected, the result is 0 due to the connection with *GND*).