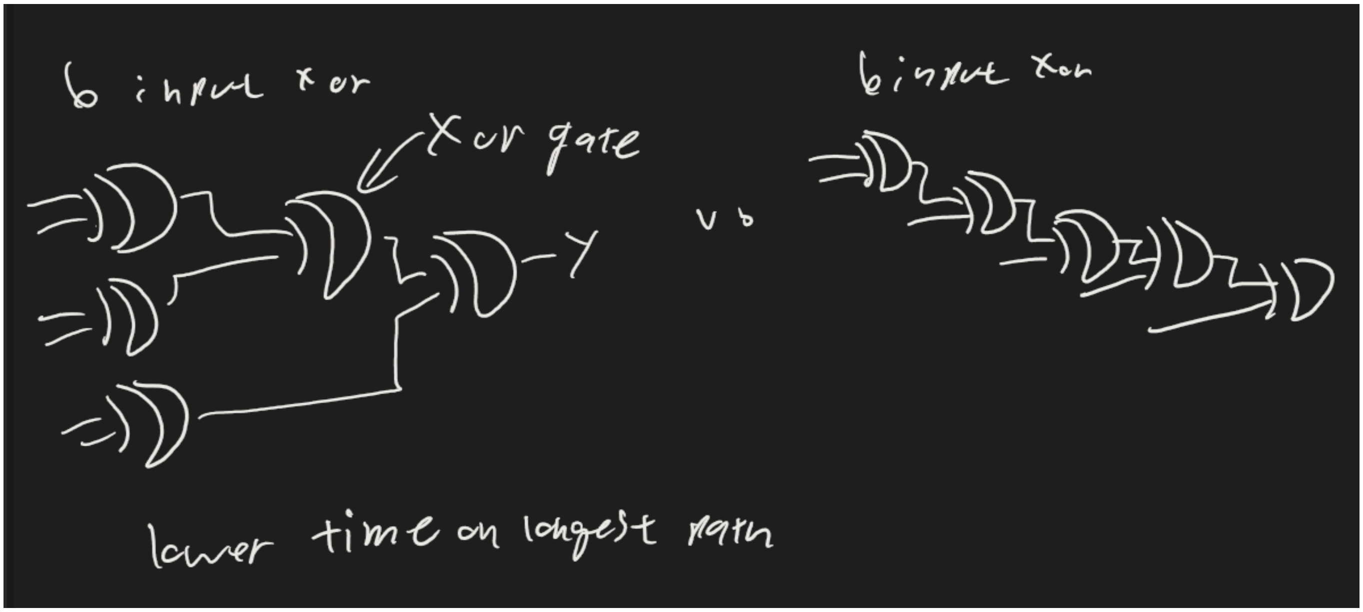
4 Logic and Notation

From Logic to Gates

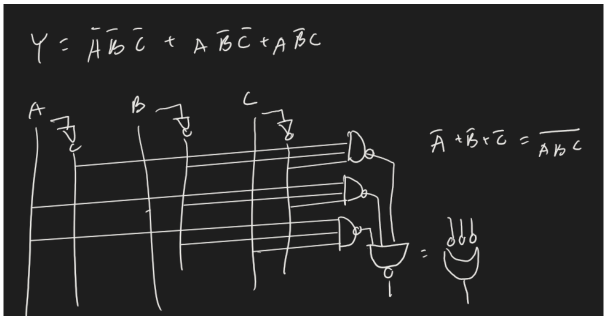


There can be multiple ways of constructing a circuit, and they can be useful for different purposes. The first is faster, but if we have everything queued and are just waiting on the last input to happen then the second would be faster.

Most of the time we build circuits in 2-level logic, which consists of $AND \to OR$.

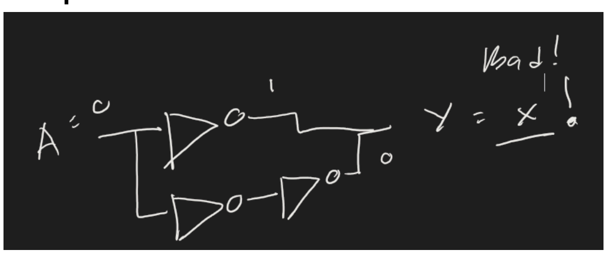
Take $Y = \bar{a}\bar{b}\bar{c} + A\bar{b}\bar{c} + a\bar{b}c$
We could easily add together each of the expressions with and gates, all going to an or gate. We could also swap the outputs to nand and invert before we go into the or gate. Or we can de Morgans, and bubble push

and make it an and gate!



X's and Z's, Oh My

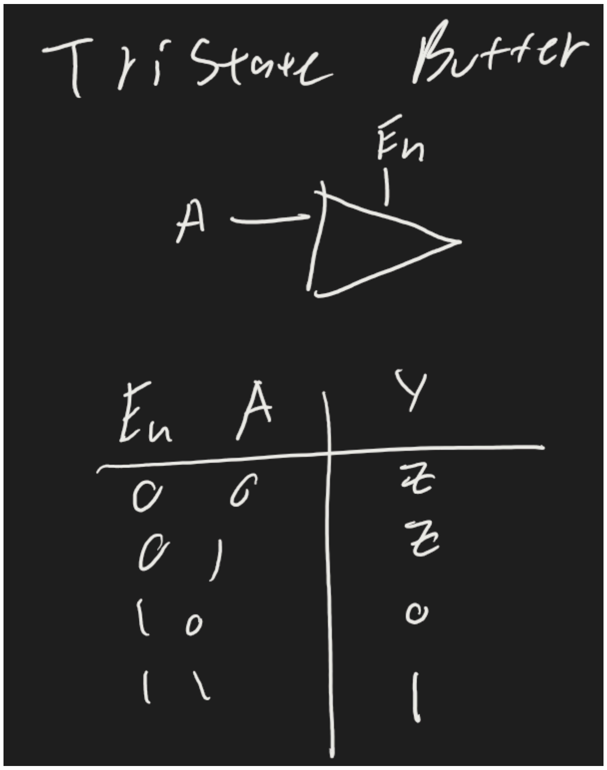
X means don't care so we can make a truth table where, if the output is independent of the input, we replace it with an x on the input side.
X also means invalid, so if we have an invalid circuit we can put it on the output.



We also have "undefined", when something is floating. This is called Z.

Sometimes we want things to float, if we have multiple outputs on a line. If we have three devices that want to talk to a computer, two of them can float and the third can have a signal, so the wire is good.

We use a Tristate buffer to regulate if a value is allowed to go out, or if it is floating.



Karnaugh Maps

We want to make a systematic way of reducing boolean expressions.

We make a table out of each variable, and mark down the ones. We then circle the most number of adjacent boxes that are powers of two (and let the boxes wrap around).

$$Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$$

C \ AB	00	01	11	10
0	1	0	0	1
1	0	0	0	1

$$Y = A\bar{B} + \bar{B}\bar{C}$$

CD \ AB	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	1	0
10	1	0	0	0

$$Y = AB + A\bar{B}\bar{C} + ABCD$$

or

$$Y = AB + \bar{B}\bar{C} + ABCD$$

For some practice: