! This class has been made inactive. No posts will be allowed until an instructor reactivates the class.

note @1481 278 views

[Extra content] Karnaugh maps

I got a question during lab today about how to simplify complex Boolean expressions (for example, when designing a circuit implementation of an FSM). While you won't need this in the context of 61C, an interesting tool to simplify these expressions is a Karnaugh map (or K-map). At a high level, a K-map is a different representation of a truth table that makes it easier to group outputs visually and assign a common Boolean algebra expression to them. This page explains it pretty well: https://www.eetimes.com/document.asp?doc_id=1278973#

If you're interested in learning more, feel free to swing by office hours 6-7 today and I'm happy to talk about this more!

Example: let's say we have the following truth table for a logic function with 4 inputs A-D and one output O.

A	B	C	D	O
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

The K-map for this will look like the following, putting AB on the vertical axis and CD on the horizontal:

	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	1	0	0
10	1	1	1	1

Note that we have two clearly defined, convenient groups of 1s: the horizontal group corresponding to AB=10 and the vertical group corresponding to CD=01. If a combination of inputs falls in either of these groups, we know that it produces a true output. This gives us:

$$O = A\overline{B} + \overline{C}D$$

Similarly, you can make groups of 0s and apply DeMorgan's law, knowing that a true output cannot fall into **any** group of 0s.

other

~ An instructor (Jerry Xu) thinks this is a good note ~

Updated 1 year ago by Dinesh Parimi

followup discussions *for lingering questions and comments*