Karnaugh Maps

- Karnaugh Maps (or K-maps) are a powerful visual tool for carrying out simplification and manipulation of logical expressions having up to 5 variables
- The K-map is a rectangular array of cells
  - Each possible state of the input variables corresponds uniquely to one of the cells
  - The corresponding output state is written in each cell

K-maps example

- From truth table to K-map

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

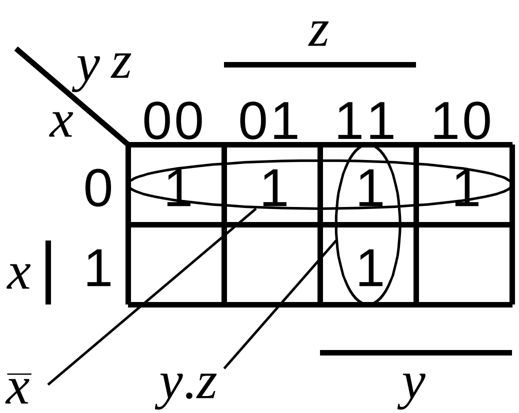
x \ yz	00	01	11	10
0	1	1	1	1
1			1	

Note that the logical state of the variables follows a Gray code, i.e., only one of them changes at a time

The exact assignment of variables in terms of their position on the map is not important

K-maps example

- Having plotted the minterms, how do we use the map to give a simplified expression?



- Group terms
  - Having size equal to a power of 2, e.g., 2, 4, 8, etc.
  - Large groups best since they contain fewer variables
  - Groups can wrap around edges and corners

So, the simplified func. is,

$$f = \bar{x} + y.z$$ as before

K-maps – 4 variables

- K maps from Boolean expressions
  - Plot $f = \overline{a}.b + b.\overline{c}.\overline{d}$

$ab \backslash cd$	00	01	11	10
00				
01	1	1	1	1
11	1			
10				

- See in a 4 variable map:
  - 1 variable term occupies 8 cells
  - 2 variable terms occupy 4 cells
  - 3 variable terms occupy 2 cells, etc.

K-maps – 4 variables

- For example, plot

$$f = \overline{b}$$

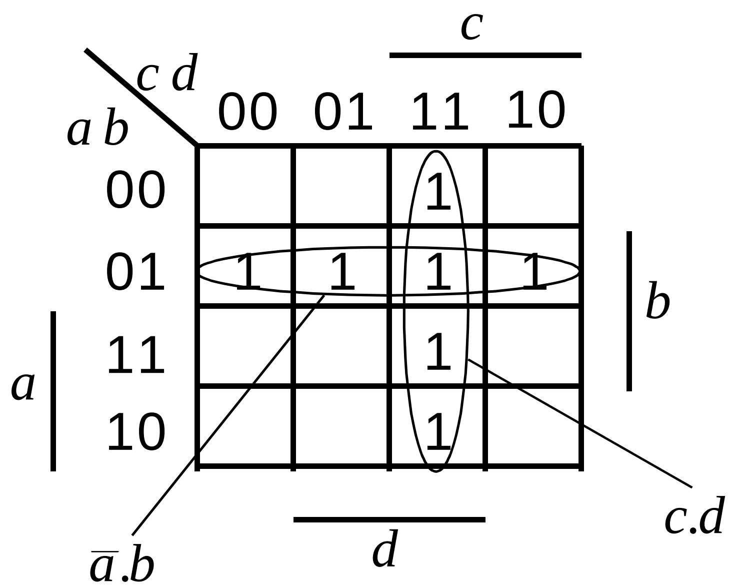
$ab \backslash cd$	00	01	11	10
00	1	1	1	1
01				
11				
10	1	1	1	1

$$f = \overline{b}.\overline{d}$$

$ab \backslash cd$	00	01	11	10
00	1			1
01				
11				
10	1			1

K-maps – 4 variables

- Simplify, $f = \overline{a}.b.\overline{d} + b.c.d + \overline{a}.b.\overline{c}.d + c.d$



So, the simplified func. is,

$$f = \overline{a}.b + c.d$$

POS Simplification

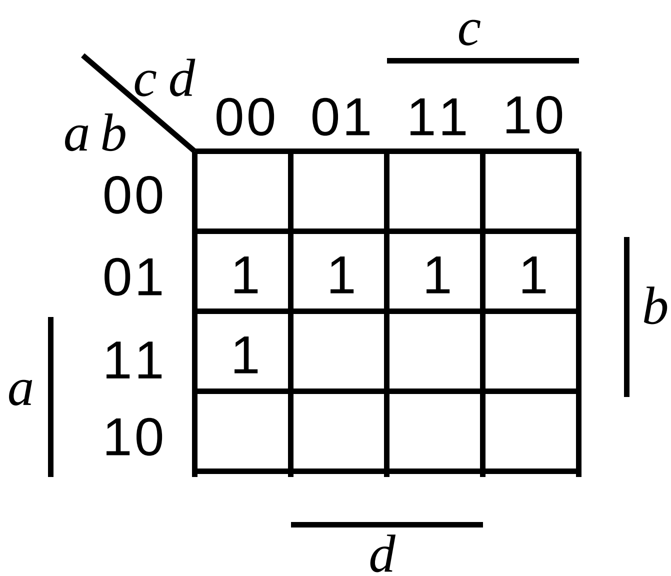
- Note that the previous examples have yielded simplified expressions in the SOP form
  - Suitable for implementations using AND followed by OR gates, or only NAND gates (using DeMorgans to transform the result – see previous Bubble logic slides)
- However, sometimes we may wish to get a simplified expression in POS form
  - Suitable for implementations using OR followed by AND gates, or only NOR gates

POS Simplification

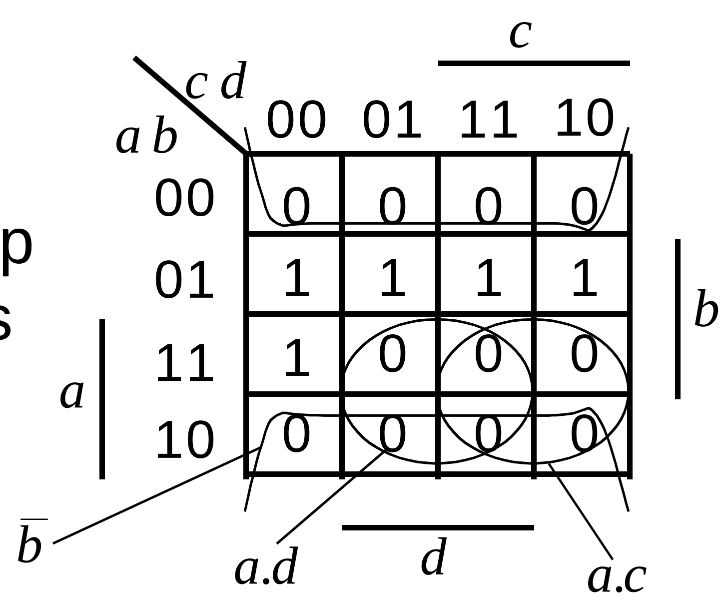
- To do this we group the zeros in the map
  - i.e., we simplify the complement of the function
- Then we apply DeMorgans and complement
- Use ‘bubble’ logic if NOR only implementation is required

POS Example

- Simplify $f = \overline{a}.b + b.\overline{c}.\overline{d}$ into POS form.



Group zeros



$\overline{f} = \overline{b} + a.c + a.d$

POS Example

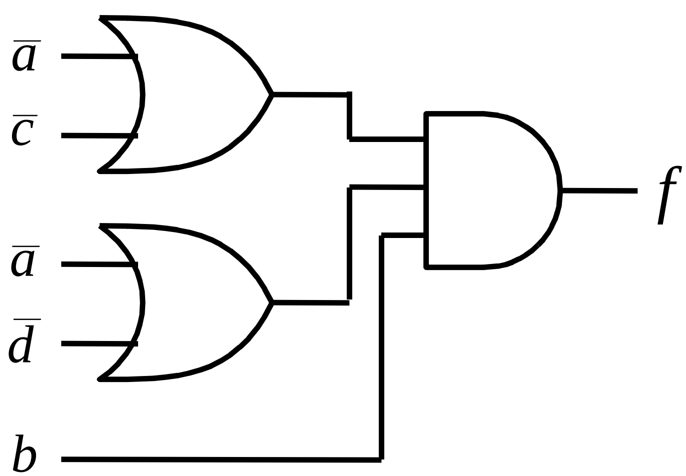
- Applying DeMorgans to

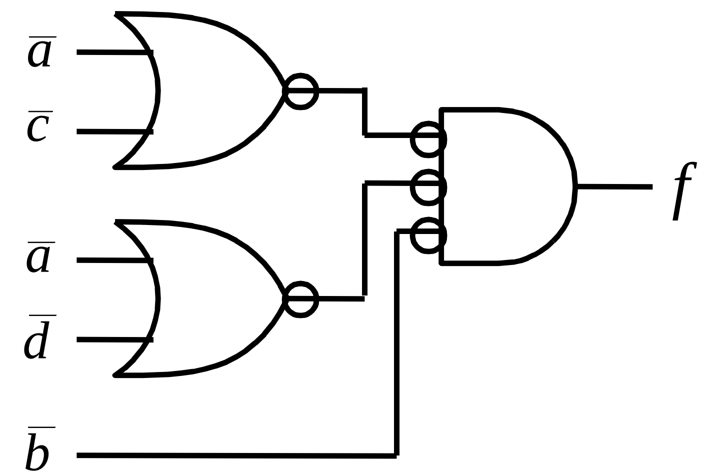
$$\bar{f} = \bar{b} + a.c + a.d$$

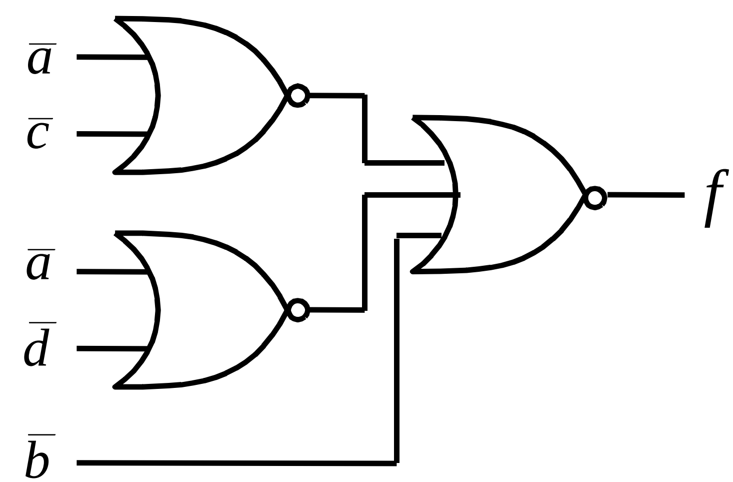
gives,

$$\bar{f} = \overline{b.(\bar{a}+\bar{c}).(\bar{a}+\bar{d})}$$

$$f = b.(\bar{a}+\bar{c}).(\bar{a}+\bar{d})$$







Expression in POS form

- Apply DeMorgans and take complement, i.e., $\bar{f}$ is now in SOP form
- Fill in zeros in table, i.e., plot $\bar{f}$
- Fill remaining cells with ones, i.e., plot f
- Simplify in usual way by grouping ones to simplify f

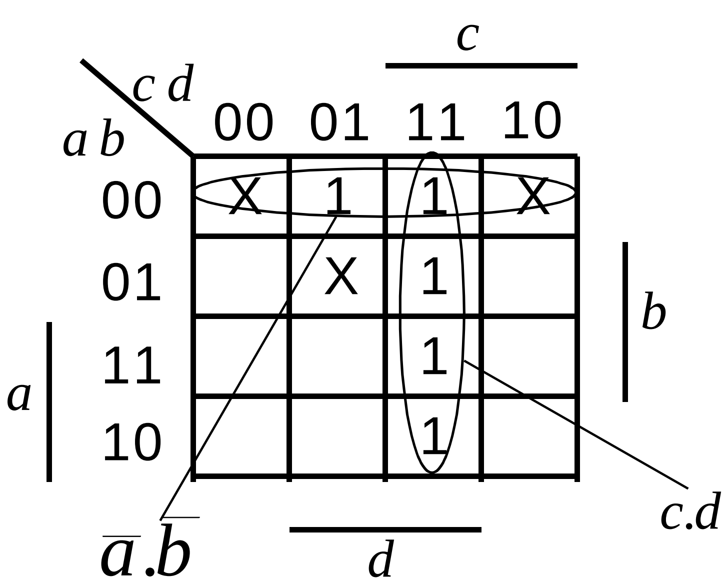
Don’t Care Conditions

- Sometimes we do not care about the output value of a combinational logic circuit, i.e., if certain input combinations can never occur, then these are known as *don’t care conditions*.
- In any simplification they may be treated as 0 or 1, depending upon which gives the simplest result.
  - For example, in a K-map they are entered as Xs

Don’t Care Conditions - Example

- Simplify the function $f = \overline{a}.\overline{b}.d + \overline{a}.c.d + a.c.d$

With don’t care conditions, $\overline{a}.\overline{b}.\overline{c}.\overline{d}\,,\ \overline{a}.\overline{b}.c.\overline{d}\,,\ \overline{a}.b.\overline{c}.d$



See only need to include Xs if they assist in making a bigger group, otherwise can ignore.

$$f = \overline{a}.\overline{b} + c.d \quad \text{or,} \quad f = \overline{a}.d + c.d$$

Some Definitions

- Cover – A term is said to cover a minterm if that minterm is part of that term
- Prime Implicant – a term that cannot be further combined
- Essential Term – a prime implicant that covers a minterm that no other prime implicant covers
- Covering Set – a minimum set of prime implicants which includes all essential terms plus any other prime implicants required to cover all minterms