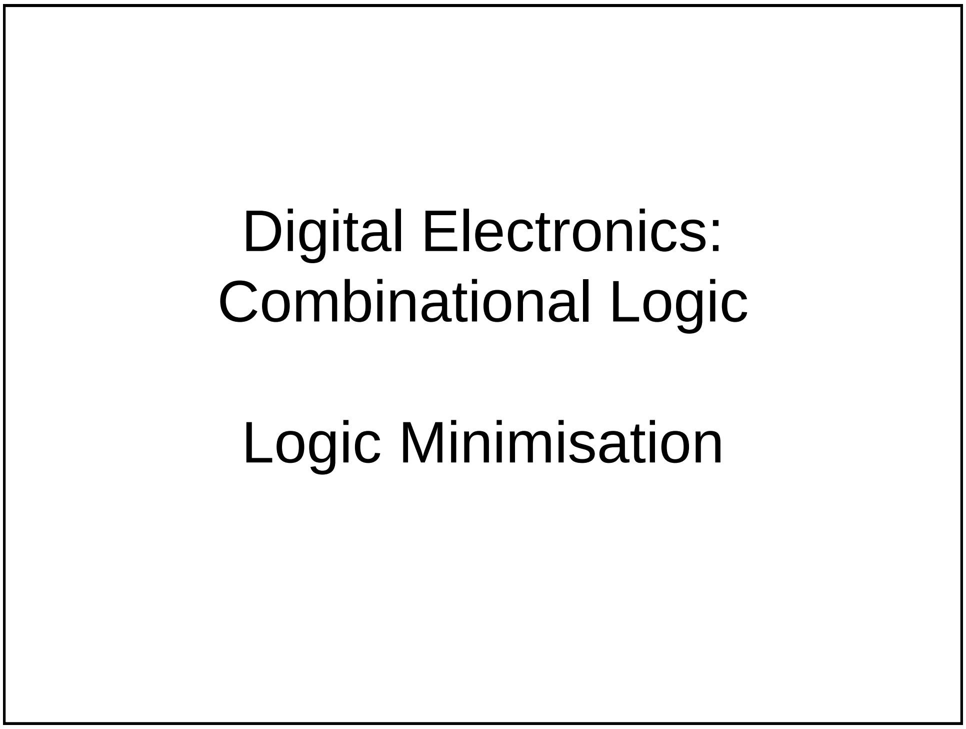
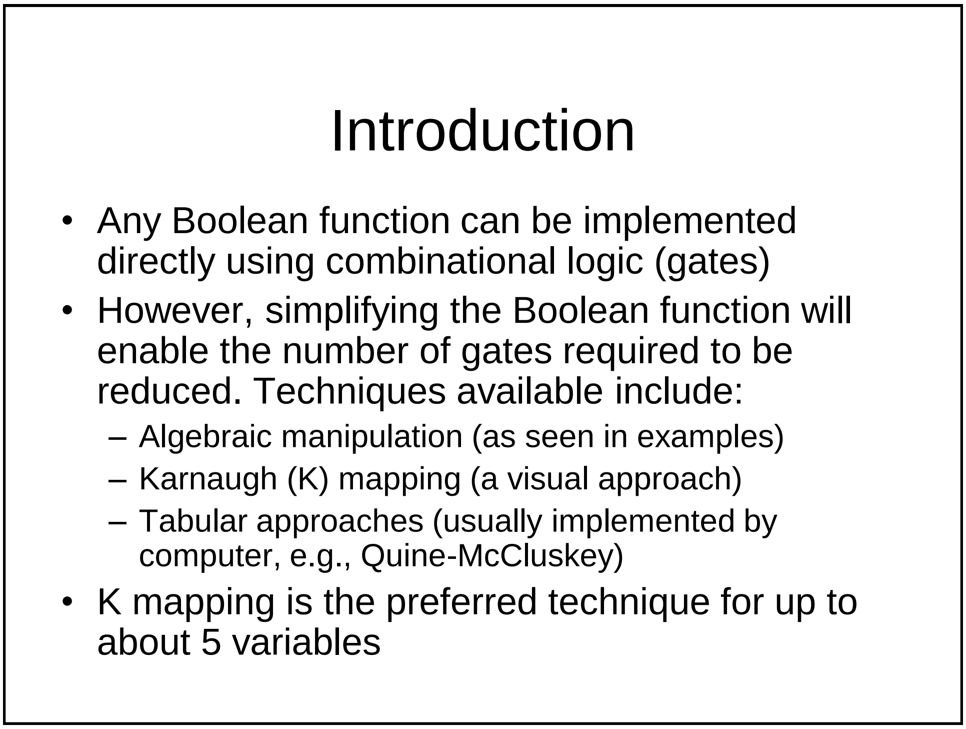


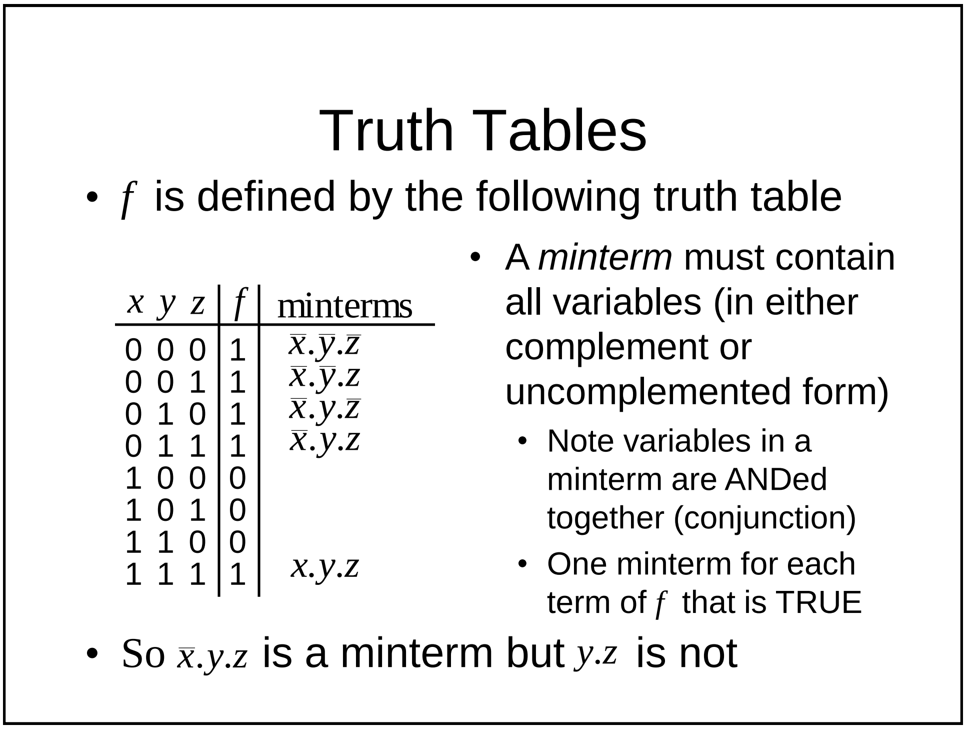
Digital Electronics: Combinational Logic

Logic Minimisation



Introduction

- Any Boolean function can be implemented directly using combinational logic (gates)
- However, simplifying the Boolean function will enable the number of gates required to be reduced. Techniques available include:
  - Algebraic manipulation (as seen in examples)
  - Karnaugh (K) mapping (a visual approach)
  - Tabular approaches (usually implemented by computer, e.g., Quine-McCluskey)
- K mapping is the preferred technique for up to about 5 variables

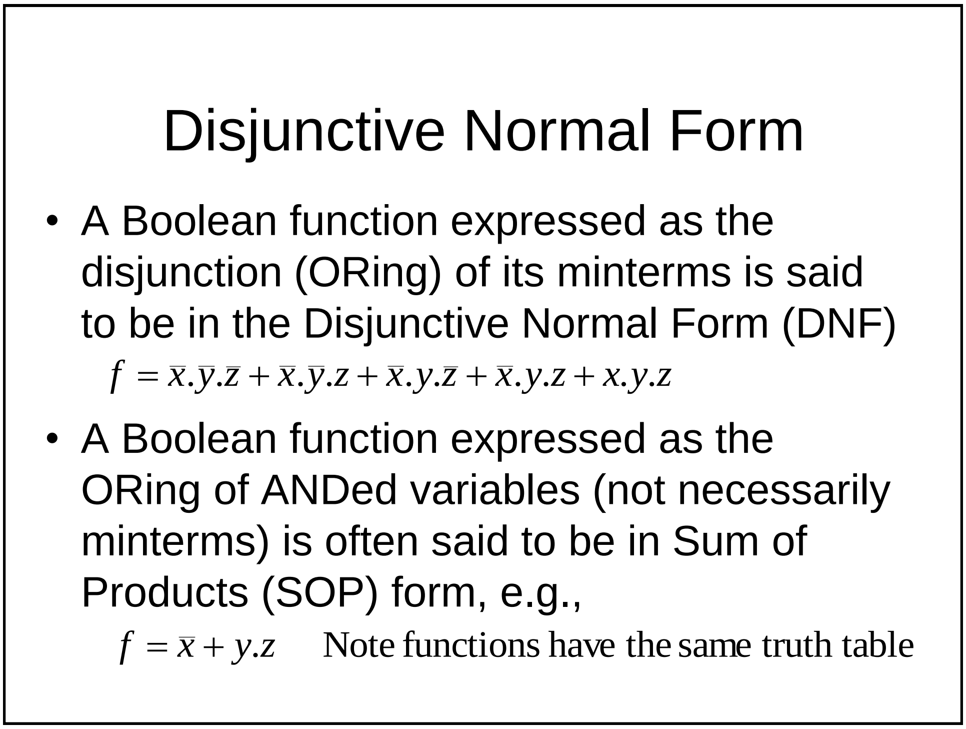


Truth Tables

- f is defined by the following truth table

x	y	z	f	minterms
0	0	0	1	$\bar{x}.\bar{y}.\bar{z}$
0	0	1	1	$\bar{x}.\bar{y}.z$
0	1	0	1	$\bar{x}.y.\bar{z}$
0	1	1	1	$\bar{x}.y.z$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$x.y.z$

- A *minterm* must contain all variables (in either complement or uncomplemented form)
  - Note variables in a minterm are ANDed together (conjunction)
  - One minterm for each term of f that is TRUE
- So $\bar{x}.y.z$ is a minterm but $y.z$ is not



Disjunctive Normal Form

- A Boolean function expressed as the disjunction (ORing) of its minterms is said to be in the Disjunctive Normal Form (DNF)

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

- A Boolean function expressed as the ORing of ANDed variables (not necessarily minterms) is often said to be in Sum of Products (SOP) form, e.g.,

$f = \bar{x} + y.z$ Note functions have the same truth table

Maxterms

- A maxterm of n Boolean variables is the disjunction (ORing) of all the variables either in complemented or uncomplemented form.
  - Referring back to the truth table for f, we can write,

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

    Applying De Morgan (and complementing) gives

    $$f = (\bar{x} + y + z).(\bar{x} + y + \bar{z}).(\bar{x} + \bar{y} + z)$$

    So it can be seen that the maxterms of f are effectively the minterms of $\bar{f}$ with each variable complemented

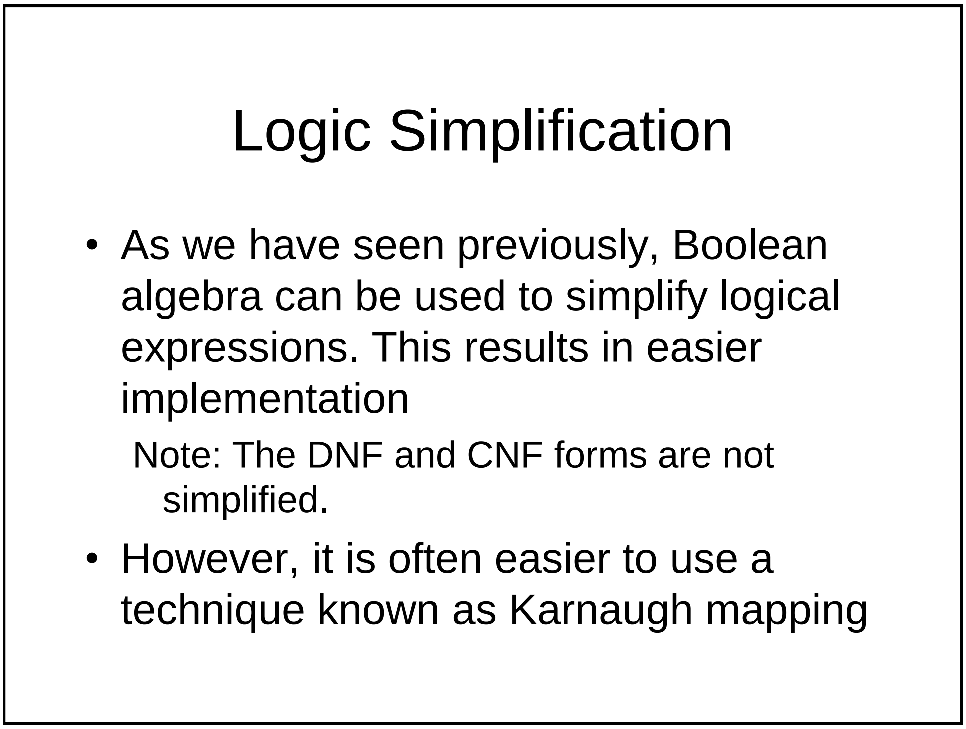
Conjunctive Normal Form

- A Boolean function expressed as the conjunction (ANDing) of its maxterms is said to be in the Conjunctive Normal Form (CNF)

  $$f = (\bar{x} + y + z).(\bar{x} + y + \bar{z}).(\bar{x} + \bar{y} + z)$$

- A Boolean function expressed as the ANDing of ORed variables (not necessarily maxterms) is often said to be in Product of Sums (POS) form, e.g.,

  $$f = (\bar{x} + y).(\bar{x} + z)$$

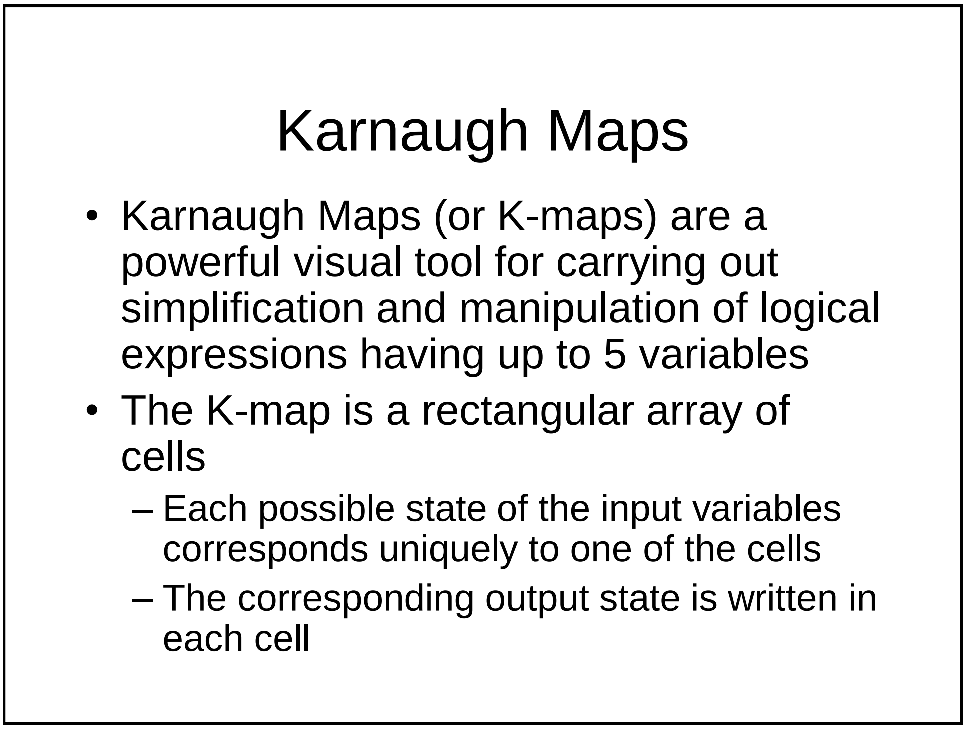


Logic Simplification

- As we have seen previously, Boolean algebra can be used to simplify logical expressions. This results in easier implementation

  Note: The DNF and CNF forms are not simplified.

- However, it is often easier to use a technique known as Karnaugh mapping



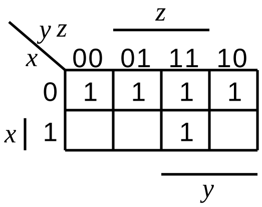
Karnaugh Maps

- 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



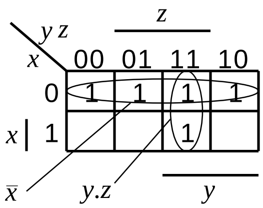
x \ yz	00	01	11	10
0	1	1	1	1
1			1	

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?



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

So, the simplified func. is,

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

- 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

K-maps – 4 variables

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

				c	c	
	ab \ cd	00	01	11	10	
	00					
	01	1	1	1	1	b
a	11	1				b
a	10					
			d	d		

- 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

$f = \overline{b}$

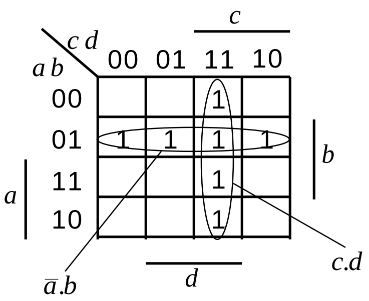
				c	c	
	ab \ cd	00	01	11	10	
	00	1	1	1	1	
	01					b
a	11					b
a	10	1	1	1	1	
			d	d		

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

				c	c	
	ab \ cd	00	01	11	10	
	00	1			1	
	01					b
a	11					b
a	10	1			1	
			d	d		

K-maps – 4 variables

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



c d a b	00	01	11	10
00			1	
01	1	1	1	1
11			1	
10			1	

So, the simplified func. is,

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

POS Simplification

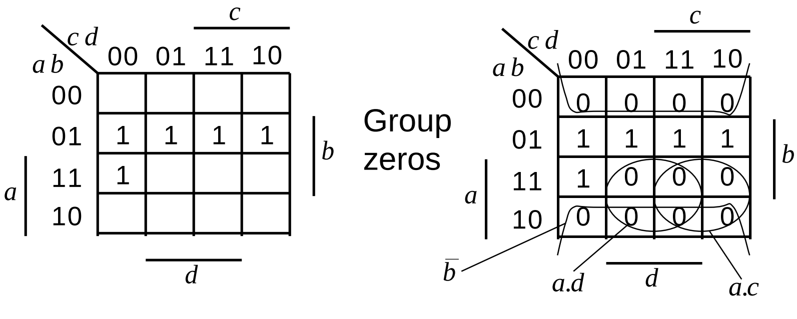
- Note that the previous examples have yielded simplified expressions in the SOP form
  - Suitable for implementations using AND followed by OR gates, or only NAND gates (using DeMorgans to transform the result – see previous Bubble logic slides)
- However, sometimes we may wish to get a simplified expression 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.



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

Group zeros

ab \ cd	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	0	0	0
10	0	0	0	0

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

POS Example

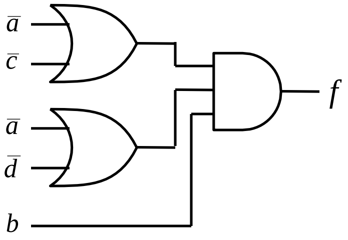
- Applying DeMorgans to

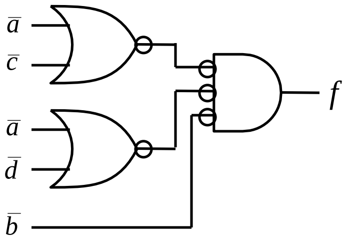
  $\bar{f} = \bar{b} + a.c + a.d$

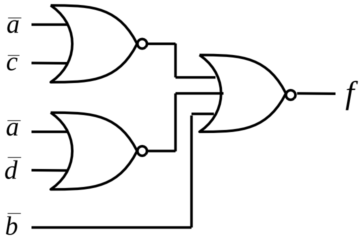
  gives,

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

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







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

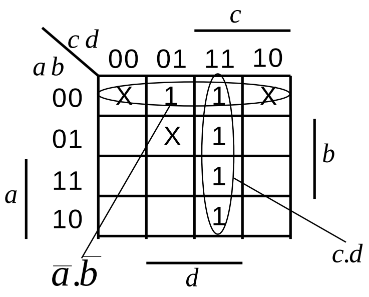
Don’t Care Conditions

- 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, $\overline{a}.\overline{b}.\overline{c}.\overline{d}\,,\ \overline{a}.\overline{b}.c.\overline{d}\,,\ \overline{a}.b.\overline{c}.d$



ab \ cd	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

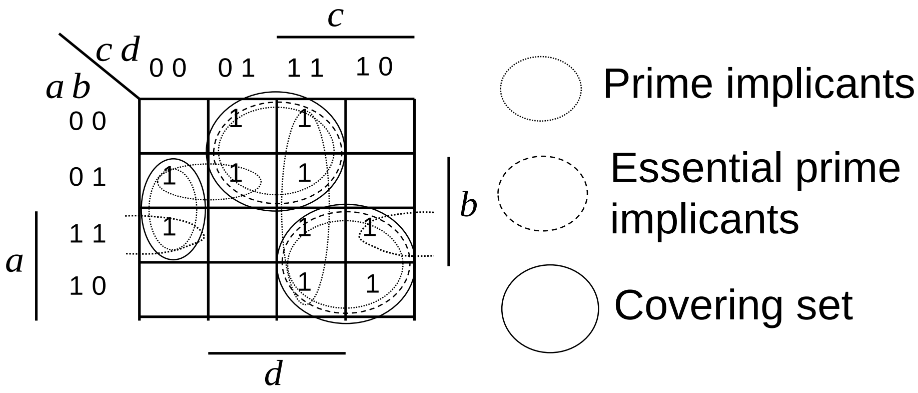
See only need to include Xs if they assist in making a bigger group, otherwise can ignore.

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

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 Prime Implicant – 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

Some Definitions - Example



ab \ cd	0 0	0 1	1 1	1 0
0 0		1	1	
0 1	1	1	1	
1 1	1		1	1
1 0			1	1

Prime implicants

Essential prime implicants

Covering set

Tabular Simplification

- Except in special cases or for sparse truth tables, the K-map method is not practical beyond 6 variables
- A systematic approach known as the *Quine-McCluskey (Q-M) Method* finds the minimised representation of any Boolean expression
- It is a tabular method that ensures all the prime implicants are found and can be automated for use on a computer

Q-M Method

- The Q-M Method has 2 steps:
  - First a table, known as the *QM implication table*, is used to find all the prime implicants;
  - Next the minimum cover set is found using the *prime implicant* chart.
- We will use a 4 variable example to show the method in operation:
  - Minterms are: 4,5,6,8,9,10,13
  - Don’t cares are: 0,7,15.

Q-M Method

- The first step is to list all the minterms and don’t cares in terms of their minterm indices represented as a binary number
  - Note the entries are grouped according to the number of 1s in the binary representation
  - The 1st column contains the minterms
  - After applying the method, the 2nd column will contain 3 variable terms. Similarly for subsequent columns.

Q-M Method

- The method begins by listing groups of minterms and don’t cares in groups containing ascending numbers of 1s with a blank line between the groups
  - Thus the first group has zero ones, the second group has a single 1 and the third has two 1s and so on
- We next apply the so called *uniting theorem* iteratively as follows

Q-M Method – Uniting Theorem

- Compare elements in the 1st group (no 1s) with all elements in the 2nd group. If they differ by a single bit, it means the terms are adjacent (think K-map)
- Adjacent terms are placed in the 2nd column with the single bit that differs replaced by a dash (-). Terms in the 1st column that contribute to a term in the second are *ticked*, i.e., they are *not* prime implicants.
- Now repeat for the groups in the 2nd column
- As before groups must differ only by a single bit but they must also have a – in the same position
- Groups in 2nd column that do not contribute to the 3rd column are marked with an asterix (*), i.e., they are prime implicants

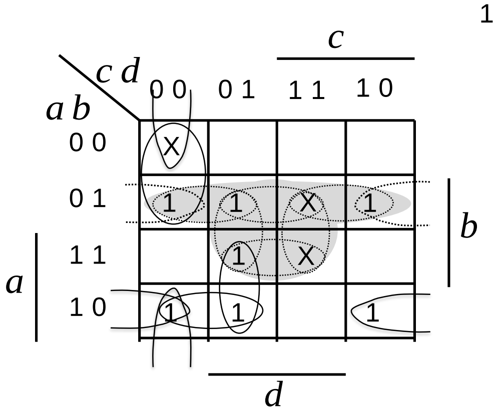
Q-M – Implication Table

- Minterms are: 4,5,6,8,9,10,13
- Don't cares are: 0,7,15.

Column 1	Column 2	Column 3
0 0 0 0✓	0 - 0 0*	0 1 - - *
	- 0 0 0*	
0 1 0 0✓		- 1 - 1*
1 0 0 0✓	0 1 0 - ✓	
	0 1 - 0✓	
0 1 0 1✓	1 0 0 - *	
0 1 1 0✓	1 0 - 0*	
1 0 0 1✓		
1 0 1 0✓	0 1 - 1✓	
	- 1 0 1✓	
0 1 1 1✓	0 1 1 - ✓	
1 1 0 1✓	1 - 0 1*	
1 1 1 1✓	- 1 1 1✓	
	1 1 - 1✓	

K-map view of Q-M example

1



ab \ cd	0 0	0 1	1 1	1 0
0 0	X			
0 1	1	1	X	1
1 1		1	X	
1 0	1	1		1

◌ Col. 2 adjacent minterms

○ Col. 2 * adjacent minterms, i.e., prime implicants

● Col. 3 prime implicants

Q-M – Finding Min Cover

- The second step is to find the lowest number of prime implicants that cover the function – this is achieved using the *prime implicant chart*
- This chart is organised as follows:
  - Label columns with the minterm indices (don't include don't cares)
  - Label rows with minterms covered by a given prime implicant. To do this dashes (-) in a prime implicant are replaced by all combinations of 0s and 1s
  - Place an X in the (row, column) location if the minterm represented by the column index is covered by the prime implicant associated with the row
  - The next slide shows the initial prime implicant chart

Q-M – Prime Implicant Chart

	4	5	6	8	9	10	13
0,4 (0-0 0)	X						
0,8 (- 0 0 0)				X			
8,9 (100 -)				X	X		
8,10 (10 - 0)				X		X	
9,13 (1- 0 1)					X		X
4,5,6,7 (01 - -)	X	X	X				
5,7,13,15 (- 1 - 1)		X					X

* Terms in Implication Table

Minterms (exc. don't cares)

- Now we look for the essential prime implicants – These are indicated when there is only a single X in any column, i.e., This means there is a minterm covered by one and only prime implicant

Q-M – Prime Implicant Chart

- The essential terms must be included in the final cover
  - Draw lines in the column and row that have a X associated with an essential prime implicant and draw a box around the prime
  - These minterms are already covered by the essential primes

	4	5	6	8	9	10	13
0,4 (0-0 0)	X						
0,8 (- 0 0 0)				X			
8,9 (100 -)				X	X		
8,10 (10 - 0)				X		X	
9,13 (1- 0 1)					X		X
4,5,6,7 (01 - -)	X	X	X				
5,7,13,15 (- 1 - 1)		X					X

Q-M – Prime Implicant Chart

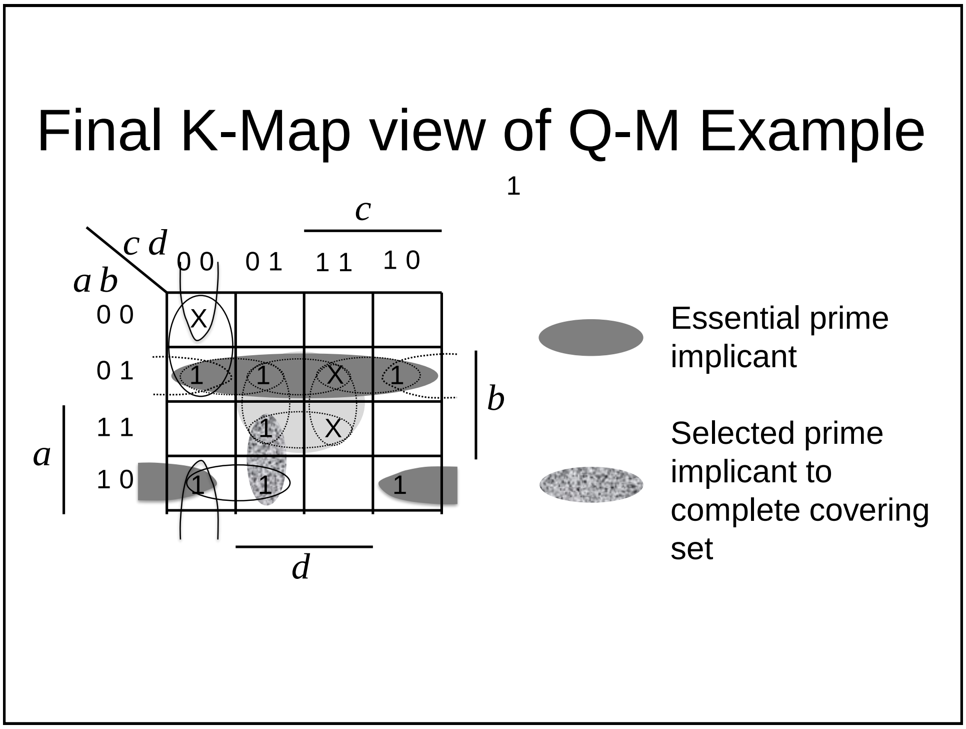
- The essential prime implicants usually cover additional minterms.
  - We must also cross out any columns that have an X in a row associated with an essential prime since these minterms are already covered by the essential primes

	4	5	6	8	9	10	13
0,4 (0-0 0)	X						
0,8 (- 0 0 0)				X			
8,9 (100 -)				X	X		
8,10 (10 - 0)				X		Ⓧ	
9,13 (1- 0 1)					X		X
4,5,6,7 (01 - -)	X	X	Ⓧ				
5,7,13,15 (- 1 - 1)		X					X

Q-M – Prime Implicant Chart

- We see 2 minterms are still uncovered (cols. 9 and 13)
  - The final step is to find as few primes as possible to cover the remaining minterms
  - We see the single prime implicant 1-01 covers both of them
  - The boxed terms show the final covering set

	4	5	6	8	9	10	13
0,4 (0-0 0)	X						
0,8 (- 0 0 0)				X			
8,9 (100 -)				X	X		
$\boxed{\text{8,10 (10 - 0)}}$				X		Ⓧ	
$\boxed{\text{9,13 (1- 0 1)}}$					X		X
$\boxed{\text{4,5,6,7 (01 - -)}}$	X	X	Ⓧ				
5,7,13,15 (- 1 - 1)		X					X



Final K-Map view of Q-M Example

1

a b \ c d	0 0	0 1	1 1	1 0
0 0	X			
0 1	1	1	X	1
1 1		1	X	
1 0	1	1		1

● Essential prime implicant

▨ Selected prime implicant to complete covering set