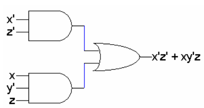
Karnaugh maps

- Last time we saw applications of Boolean logic to circuit design.
  - The basic Boolean operations are AND, OR and NOT.
  - These operations can be combined to form complex expressions, which can also be directly translated into a hardware circuit.
  - Boolean algebra helps us simplify expressions and circuits.
- Today we'll look at a graphical technique for simplifying an expression into a minimal sum of products (MSP) form:
  - There are a minimal number of product terms in the expression.
  - Each term has a minimal number of literals.
- Circuit-wise, this leads to a *minimal* two-level implementation.

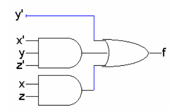


Review: Standard forms of expressions

- We can write expressions in many ways, but some ways are more useful than others
- A sum of products (SOP) expression contains:
  - Only OR (sum) operations at the "outermost" level
  - Each term that is summed must be a product of literals

f(x,y,z) = y' + x'yz' + xz

- The advantage is that any sum of products expression can be implemented using a two-level circuit
  - literals and their complements at the "0th" level
  - AND gates at the first level
  - a single OR gate at the second level
- This diagram uses some shorthands…
  - NOT gates are implicit
  - literals are reused
  - this is *not* okay in LogicWorks!



Terminology: Minterms

- A minterm is a special product of literals, in which each input variable appears exactly once.
- A function with n variables has 2^n minterms (since each variable can appear complemented or not)
- A three-variable function, such as f(x,y,z), has $2^3 = 8$ minterms:

x'y'z'	x'y'z	x'yz'	x'yz
xy'z'	xy'z	xyz'	xyz

- Each minterm is true for exactly one combination of inputs:

Minterm	Is true when…	Shorthand
x'y'z'	x=0, y=0, z=0	m_0
x'y'z	x=0, y=0, z=1	m_1
x'yz'	x=0, y=1, z=0	m_2
x'yz	x=0, y=1, z=1	m_3
xy'z'	x=1, y=0, z=0	m_4
xy'z	x=1, y=0, z=1	m_5
xyz'	x=1, y=1, z=0	m_6
xyz	x=1, y=1, z=1	m_7

Terminology: Sum of minterms form

- Every function can be written as a sum of minterms, which is a special kind of sum of products form
- The sum of minterms form for any function is *unique*
- If you have a truth table for a function, you can write a sum of minterms expression just by picking out the rows of the table where the function output is 1.

x	y	z	f(x,y,z)	f'(x,y,z)
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

f = x'y'z' + x'y'z + x'yz' + x'yz + xyz'
  = $m_0 + m_1 + m_2 + m_3 + m_6$
  = $\Sigma m(0,1,2,3,6)$

f' = xy'z' + xy'z + xyz
  = $m_4 + m_5 + m_7$
  = $\Sigma m(4,5,7)$

f' contains all the minterms not in f

Re-arranging the truth table

- A two-variable function has four possible minterms. We can re-arrange these minterms into a Karnaugh map.

x	y	minterm
0	0	x'y'
0	1	x'y
1	0	xy'
1	1	xy

X \ Y	0	1
0	x'y'	x'y
1	xy'	xy

- Now we can easily see which minterms contain common literals.
  - Minterms on the left and right sides contain y' and y respectively.
  - Minterms in the top and bottom rows contain x' and x respectively.

X \ Y	0	1
0	x'y'	x'y
1	xy'	xy

	y'	y
x'	x'y'	x'y
x	xy'	xy

Karnaugh map simplifications

- Imagine a two-variable sum of minterms:

x'y' + x'y

- Both of these minterms appear in the top row of a Karnaugh map, which means that they both contain the literal x'.

		Y
	x'y'	x'y
X	xy'	xy

- What happens if you simplify this expression using Boolean algebra?

x'y' + x'y	= x'(y' + y)	[Distributive]
	= x' • 1	[y + y' = 1]
	= x'	[x • 1 = x]

More two-variable examples

- Another example expression is x'y + xy.
  - Both minterms appear in the right side, where y is uncomplemented.
  - Thus, we can reduce x'y + xy to just y.

		y
	x'y'	x'y
X	xy'	xy

- How about x'y' + x'y + xy?
  - We have x'y' + x'y in the top row, corresponding to x'.
  - There's also x'y + xy in the right side, corresponding to y.
  - This whole expression can be reduced to x' + y.

		y
	x'y'	x'y
X	xy'	xy

A three-variable Karnaugh map

- For a three-variable expression with inputs x, y, z, the arrangement of minterms is more tricky:

		YZ			
		00	01	11	10
X	0	x'y'z'	x'y'z	x'yz	x'yz'
	1	xy'z'	xy'z	xyz	xyz'

		YZ			
		00	01	11	10
X	0	m_0	m_1	m_3	m_2
	1	m_4	m_5	m_7	m_6

- Another way to label the K-map (use whichever you like):

			y	
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		

			y	
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z		

Why the funny ordering?

- With this ordering, any group of 2, 4 or 8 adjacent squares on the map contains common literals that can be factored out.

			y	
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		

x'y'z + x'yz
= x'z(y' + y)
= x'z • 1
= x'z

- "Adjacency" includes wrapping around the left and right sides:

			y	
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		

x'y'z' + xy'z' + x'yz' + xyz'
= z'(x'y' + xy' + x'y + xy)
= z'(y'(x' + x) + y(x' + x))
= z'(y'+y)
= z'

- We'll use this property of adjacent squares to do our simplifications.

Example K-map simplification

- Let's consider simplifying f(x,y,z) = xy + y'z + xz.
- First, you should convert the expression into a sum of minterms form, if it's not already.
  - The easiest way to do this is to make a truth table for the function, and then read off the minterms.
  - You can either write out the literals or use the minterm shorthand.
- Here is the truth table and sum of minterms for our example:

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

f(x,y,z) = x'y'z + xy'z + xyz' + xyz
= m_1 + m_5 + m_6 + m_7

Unsimplifying expressions

- You can also convert the expression to a sum of minterms with Boolean algebra.
  - Apply the distributive law in reverse to add in missing variables.
  - Very few people actually do this, but it's occasionally useful.

xy + y'z + xz = (xy • 1) + (y'z • 1) + (xz • 1)
= (xy • (z' + z)) + (y'z • (x' + x)) + (xz • (y' + y))
= (xyz' + xyz) + (x'y'z + xy'z) + (xy'z + xyz)
= xyz' + xyz + x'y'z + xy'z

- In both cases, we're actually "unsimplifying" our example expression.
  - The resulting expression is larger than the original one!
  - But having all the individual minterms makes it easy to combine them together with the K-map.

Making the example K-map

- Next up is drawing and filling in the K-map.
  - Put 1s in the map for each minterm, and 0s in the other squares.
  - You can use either the minterm products or the shorthand to show you where the 1s and 0s belong.
- In our example, we can write f(x,y,z) in two equivalent ways.

f(x,y,z) = x'y'z + xy'z + xyz' + xyz

			y	
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		

f(x,y,z) = m_1 + m_5 + m_6 + m_7

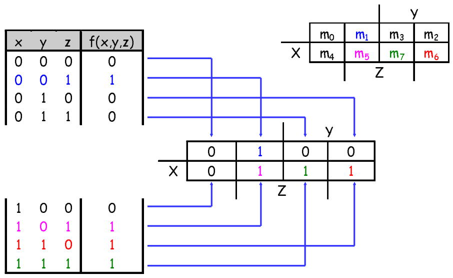
			y	
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z		

- In either case, the resulting K-map is shown below.

			y	
	0	1	0	0
X	0	1	1	1
		Z		

K-maps from truth tables

- You can also fill in the K-map directly from a truth table.
  - The output in row *i* of the table goes into square m_i of the K-map.
  - Remember that the rightmost columns of the K-map are "switched."



Grouping the minterms together

- The most difficult step is grouping together all the 1s in the K-map.
  - Make rectangles around groups of one, two, four or eight 1s.
  - All of the 1s in the map should be included in at least one rectangle.
  - Do *not* include any of the 0s.

	0	1	0	0
X	0	1	1	1

- Each group corresponds to one product term. For the simplest result:
  - Make as few rectangles as possible, to minimize the number of products in the final expression.
  - Make each rectangle as large as possible, to minimize the number of literals in each term.
  - It's all right for rectangles to overlap, if that makes them larger.

Reading the MSP from the K-map

- Finally, you can find the MSP.
  - Each rectangle corresponds to one product term.
  - The product is determined by finding the common literals in that rectangle.

	0	1	0	0
X	0	1	1	1

	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'

- For our example, we find that xy + y'z + xz = y'z + xy. (This is one of the additional algebraic laws from last time.)

Practice K-map 1

- Simplify the sum of minterms $m_1 + m_3 + m_5 + m_6$.

X				

	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6

Solutions for practice K-map 1

- Here is the filled in K-map, with all groups shown.
  - The magenta and green groups overlap, which makes each of them as large as possible.
  - Minterm m_6 is in a group all by its lonesome.

	0	1	1	0
X	0	1	0	1

- The final MSP here is x'z + y'z + xyz'.

Four-variable K-maps

- We can do four-variable expressions too!
  - The minterms in the third and fourth columns, *and* in the third and fourth rows, are switched around.
  - Again, this ensures that adjacent squares have common literals.

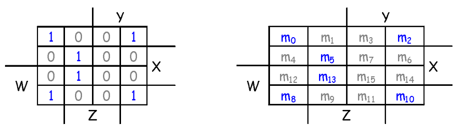
w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'	
w'xy'z'	w'xy'z	w'xyz	w'xyz'	X
wxy'z'	wxy'z	wxyz	wxyz'	X
wx'y'z'	wx'y'z	wx'yz	wx'yz'	

m_0	m_1	m_3	m_2	
m_4	m_5	m_7	m_6	X
m_{12}	m_{13}	m_{15}	m_{14}	X
m_8	m_9	m_{11}	m_{10}	

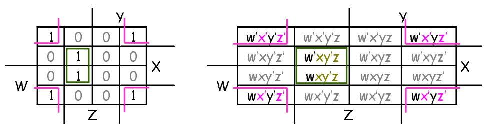
- Grouping minterms is similar to the three-variable case, but:
  - You can have rectangular groups of 1, 2, 4, 8 or 16 minterms.
  - You can wrap around *all four* sides.

Example: Simplify $m_0+m_2+m_5+m_8+m_{10}+m_{13}$

- The expression is already a sum of minterms, so here's the K-map:

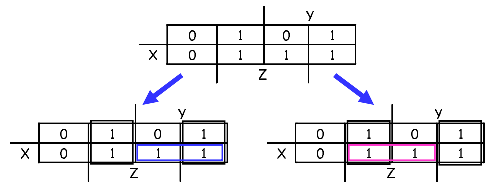


- We can make the following groups, resulting in the MSP x'z' + xy'z.



K-maps can be tricky!

- There may not necessarily be a *unique* MSP. The K-map below yields two valid and equivalent MSPs, because there are two possible ways to include minterm m_7.



y'z + yz' + xy

y'z + yz' + xz

- Remember that overlapping groups is possible, as shown above.