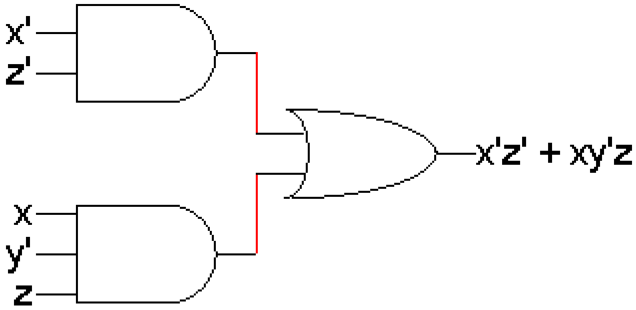
Karnaugh maps

- 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.



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

		y: 0	y: 1
X	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.

		y: 0	y: 1
X	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?

$$\begin{aligned} x'y' + x'y &= x'(y' + y) && [\text{ Distributive }] \\ &= x' \bullet 1 && [\ y + y' = 1\ ] \\ &= x' && [\ x \bullet 1 = x\ ] \end{aligned}$$

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

$$
\begin{aligned}
& x'y'z + x'yz \\
= \; & x'z(y' + y) \\
= \; & x'z \bullet 1 \\
= \; & x'z
\end{aligned}
$$

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

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

$$
\begin{aligned}
& x'y'z' + xy'z' + x'yz' + xyz' \\
= \; & z'(x'y' + xy' + x'y + xy) \\
= \; & z'(y'(x' + x) + y(x' + x)) \\
= \; & z'(y'+y) \\
= \; & z'
\end{aligned}
$$

- 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

$$\begin{aligned} f(x,y,z) &= x'y'z + xy'z + xyz' + xyz \\ &= m_1 + m_5 + m_6 + m_7 \end{aligned}$$

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.

$$\begin{aligned} xy + y'z + xz &= (xy \bullet 1) + (y'z \bullet 1) + (xz \bullet 1) \\ &= (xy \bullet (z' + z)) + (y'z \bullet (x' + x)) + (xz \bullet (y' + y)) \\ &= (xyz' + xyz) + (x'y'z + xy'z) + (xy'z + xyz) \\ &= xyz' + xyz + x'y'z + xy'z \end{aligned}$$

- 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."

			Y	
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z		

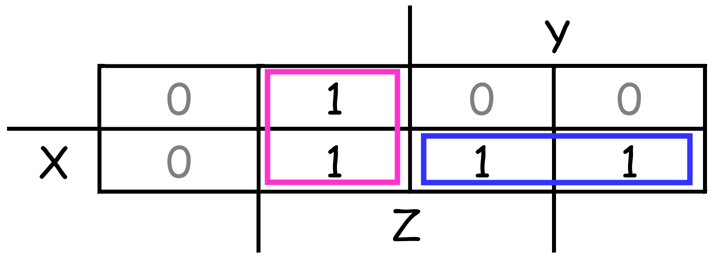
x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

			Y	
	0	1	0	0
X	0	1	1	1
		Z		

	~Y~Z	~YZ	YZ	Y~z
~X				
X				

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.



- 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.

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

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

- 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 $m_1 + m_3 + m_5 + m_6$.

			y	y
x				
		z	z	

			y	y
	m_0	m_1	m_3	m_2
x	m_4	m_5	m_7	m_6
		z	z	

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.

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

- 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.

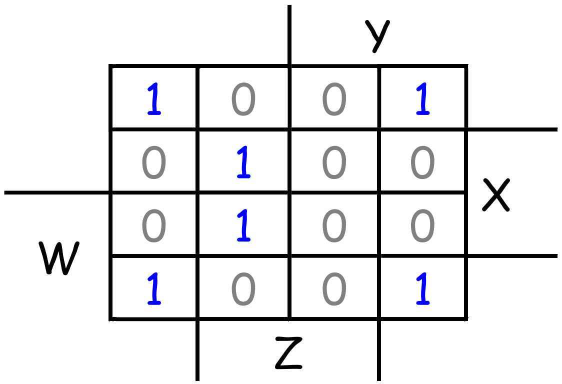
			y	y	
	w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'	
	w'xy'z'	w'xy'z	w'xyz	w'xyz'	x
w	wxy'z'	wxy'z	wxyz	wxyz'	x
w	wx'y'z'	wx'y'z	wx'yz	wx'yz'	
		z	z		

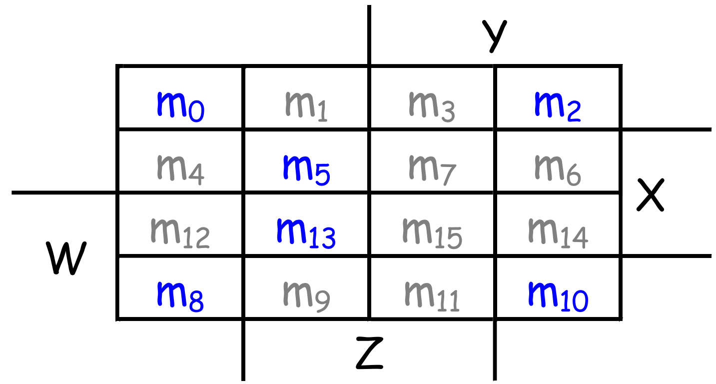
			y	y	
	m_0	m_1	m_3	m_2	
	m_4	m_5	m_7	m_6	x
w	m_{12}	m_{13}	m_{15}	m_{14}	x
w	m_8	m_9	m_{11}	m_{10}	
		z	z		

- 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.

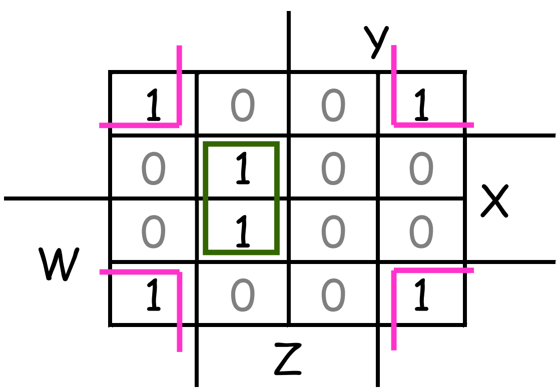
Example: Simplify $m_0+m_2+m_5+m_8+m_{10}+m_{13}$

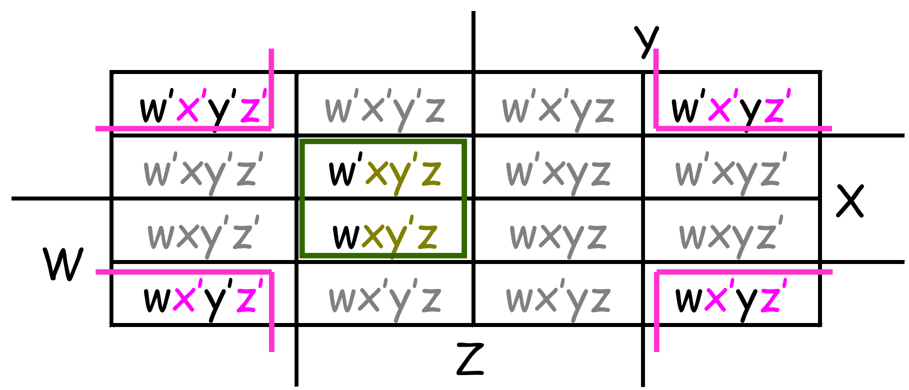
- The expression is already a sum of minterms, so here's the K-map:





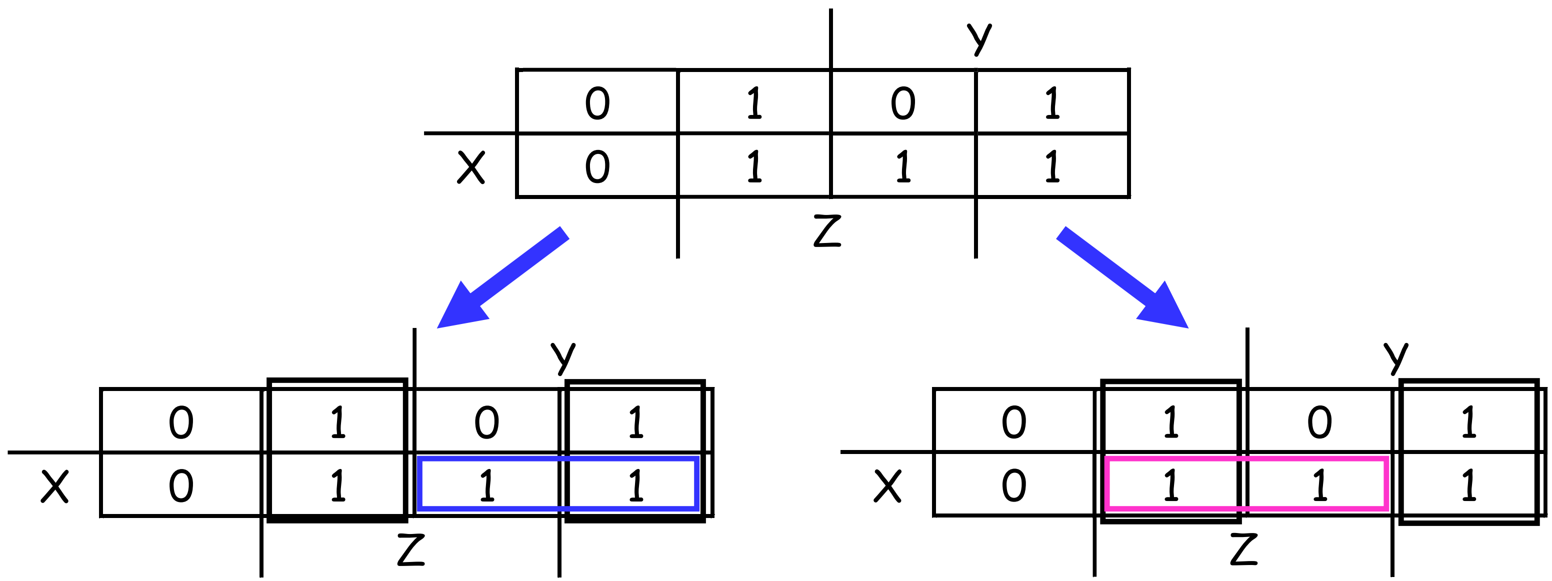
- We can make the following groups, resulting in the MSP $x'z' + xy'z$.





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.



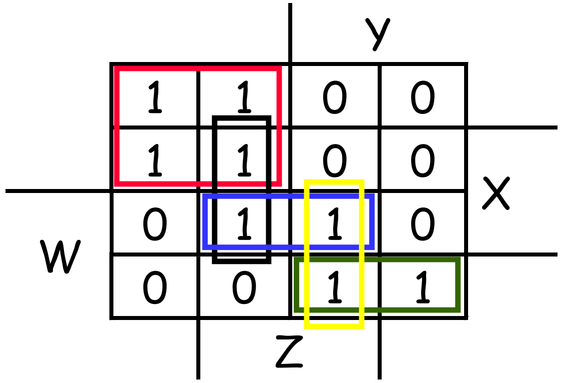
$y'z + yz' + xy$

$y'z + yz' + xz$

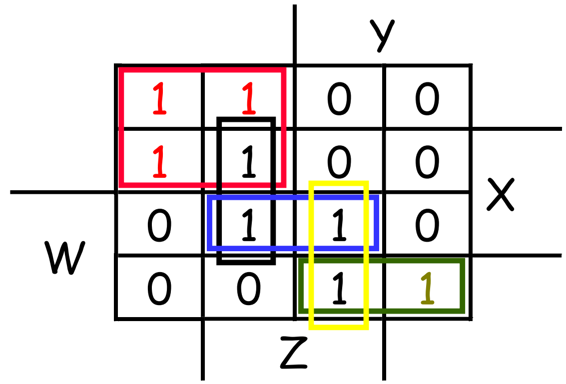
- Remember that overlapping groups is possible, as shown above.

Prime implicants

- The challenge in using K-maps is selecting the right groups. If you don't minimize the number of groups and maximize the size of each group:
  - Your resulting expression will still be equivalent to the original one.
  - But it won't be a *minimal* sum of products.
- What's a good approach to finding an actual MSP?
- First find all of the largest possible groupings of 1s.
  - These are called the prime implicants.
  - The final MSP will contain a subset of these prime implicants.
- Here is an example Karnaugh map with prime implicants marked:

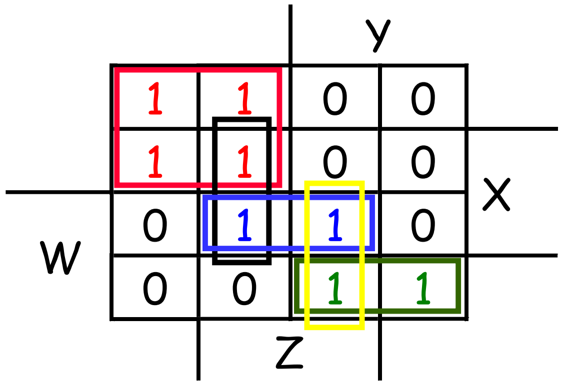


Essential prime implicants



- If any group contains a minterm that is not also covered by another overlapping group, then that is an essential prime implicant.
- Essential prime implicants *must* appear in the MSP, since they contain minterms that no other terms include.
- Our example has just two essential prime implicants:
  - The red group (w'y') is essential, because of m_0, m_1 and m_4.
  - The green group (wx'y) is essential, because of m_{10}.

Covering the other minterms



- Finally pick as few other prime implicants as necessary to ensure that all the minterms are covered.
- After choosing the red and green rectangles in our example, there are just two minterms left to be covered, m_{13} and m_{15}.
  - These are both included in the blue prime implicant, wxz.
  - The resulting MSP is w'y' + wxz + wx'y.
- The black and yellow groups are not needed, since all the minterms are covered by the other three groups.

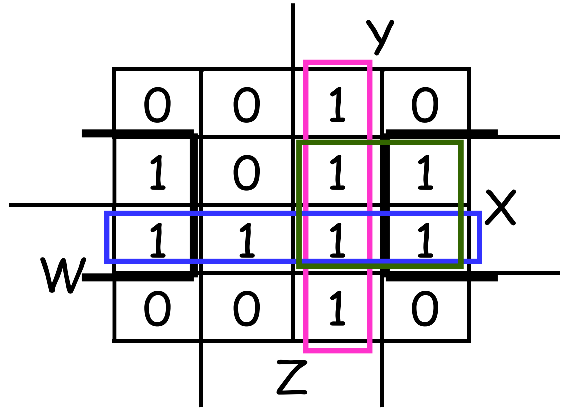
Practice K-map 2

- Simplify for the following K-map:

			Y	Y	
	0	0	1	0	
	1	0	1	1	X
W	1	1	1	1	X
W	0	0	1	0	
		Z	Z		

Solutions for practice K-map 2

- Simplify for the following K-map:



All prime implicants are circled.

Essential prime implicants are **xz'**, wx and yz.

The MSP is **xz'** + wx + yz.
(Including the group xy would be redundant.)

I don't care!

- You don't always need all 2^n input combinations in an n-variable function.
  - If you can guarantee that certain input combinations never occur.
  - If some outputs aren't used in the rest of the circuit.
- We mark don't-care outputs in truth tables and K-maps with Xs.

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

- Within a K-map, each X can be considered as either 0 or 1. You should pick the interpretation that allows for the most simplification.

Practice K-map 3

- Find a MSP for

  $f(w,x,y,z) = \Sigma m(0,2,4,5,8,14,15)$, $d(w,x,y,z) = \Sigma m(7,10,13)$

  This notation means that input combinations wxyz = 0111, 1010 and 1101 (corresponding to minterms m_7, m_{10} and m_{13}) are unused.

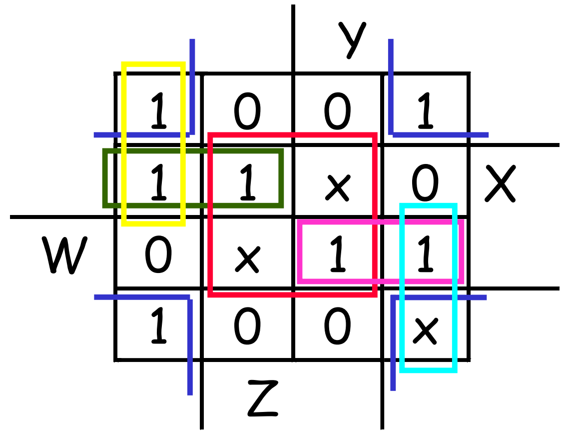
1	0	0	1
1	1	x	0
0	x	1	1
1	0	0	x

y x w z

Solutions for practice K-map 3

- Find a MSP for:

$f(w,x,y,z) = \Sigma m(0,2,4,5,8,14,15)$, $d(w,x,y,z) = \Sigma m(7,10,13)$



All prime implicants are circled. We can treat X's as 1s if we want, so the red group includes two X's, and the light blue group includes one X.

The *only* essential prime implicant is x'z'. The red group is not essential because the minterms in it also appear in other groups.

The MSP is x'z' + wxy + w'xy'. It turns out the red group is redundant; we can cover all of the minterms in the map without it.

Summary

- K-maps are an alternative to algebra for simplifying expressions.
  - The result is a *minimal sum of products*, which leads to a minimal two-level circuit.
  - It's easy to handle don't-care conditions.
  - K-maps are really only good for manual simplification of small expressions... but that's good enough for CS231!
- Things to keep in mind:
  - Remember the correct order of minterms on the K-map.
  - When grouping, you can wrap around all sides of the K-map, and your groups can overlap.
  - Make as few rectangles as possible, but make each of them as large as possible. This leads to fewer, but simpler, product terms.
  - There may be more than one valid solution.

Example: Seven Segment Display

Input: digit encoded as 4 bits: ABCD

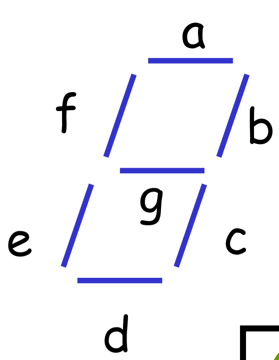


Table for e

Assumption: Input represents a legal digit (0-9)

CD / AB	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

CD' + B'D'

	A	B	C	D	e
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
X					X
X					X
X					X
X					X
X					X
X					X

Example: Seven Segment Display

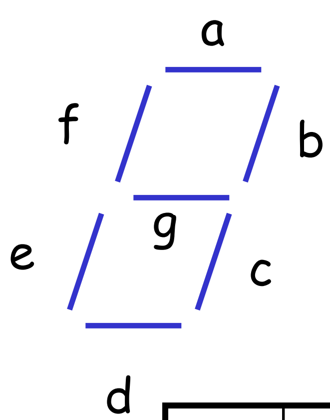


Table for a

Assumption: Input represents a legal digit (0-9)

CD / AB	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

A + C + BD + B'D'

	A	B	C	D	a
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
X					X
X					X
X					X
X					X
X					X
X					X