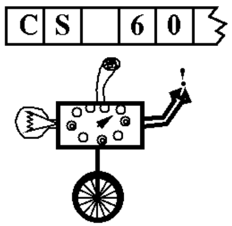


Circuit Simplification

November 12, 2001

Logic Expression Simplification

- Often it is possible to simplify function expressions from, say, their minterm form.
- This may be done by using various identities, as we know. But more systematic methods will be shown.

Simplification Example

"3-argument majority function"

Minterm form:

v w x	f
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

Notice Certain "Adjacencies"

3-argument majority function

v w x	f
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

These rows are "adjacent": their variable combinations differ in only 1 bit

$$\left.\begin{array}{l} v'wx \\ vwx \end{array}\right\} = wx$$

simpler definition:

majority(v,w,x) = $\boxed{wx}$ + vw'x + vwx'

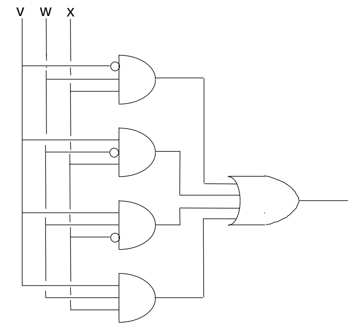
Any other Adjacencies?

(ok to use a row more than once)

v	w	x	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

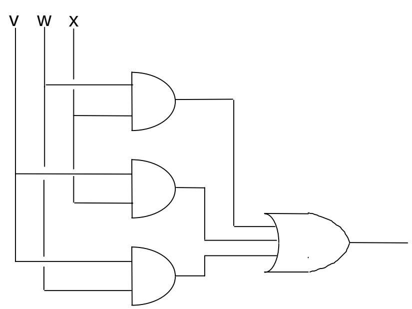
Unsimplified Majority

`majority(v,w,x) = v'wx + vw'x + vwx' + vwx`



Simplified Majority

`majority(v, w, x) = wx + vx + vw`



Comparison

- Unsimplified:
  4 and-gates, 3 inputs each,
  some with negation,
  1 4-input or-gate
- Simplified:
  3 and-gates, 2 inputs each,
  none with negation,
  1 3-input or-gate

Implications for Simplified Functions

- If communicated by human, easier to understand, transfer
- If implemented as hardware, fewer gates, wires
- If implemented as software, fewer tests, execution steps

Visualizing Adjacencies

- Hypercube representation of truth table
- Karnaugh map representation

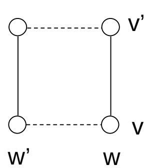
Hypercubes
(connected vertices = adjacent)

1-dimension =

1 logical variable

2-dimensions =

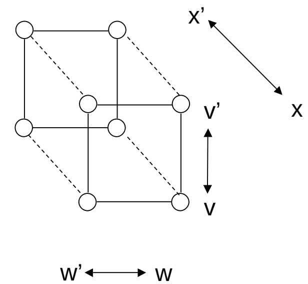
2 logical variables



Hypercubes

3-dimensions =

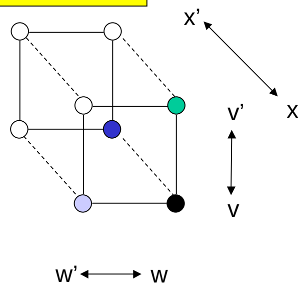
3 logical variables



Plotting Functions on Hypercubes

majority(v, w, x) = v'wx + vw'x + vwx' + vwx

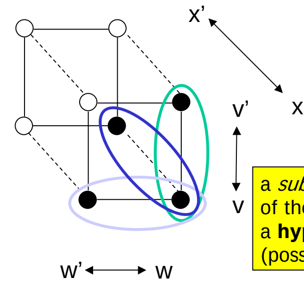
each minterm is a vertex



Simplified Functions on Hypercubes

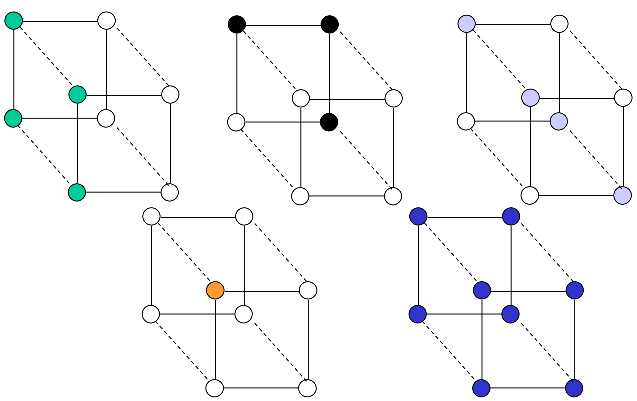
majority(v, w, x) = wx + vx + vw

each term is a ***sub-cube***



a *sub-cube* is a subset of the vertices which is a **hypercube** itself (possibly of smaller dimension)

Which sets are subcubes?



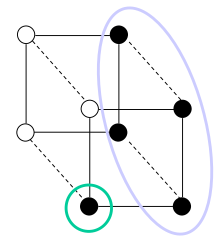
SOP Circuits

- An SOP will always correspond to a set of **sub-cubes**
  - Collectively the vertices in the sub-cubes *cover* the “on” vertices.
  - The vertices in the sub-cubes don’t cover any “off” vertices.
- Such a set is called a **cover** for the function.
- Each cover corresponds to a different gate implementation.

Simplified Covers

- In a fully-simplified SOP, each sub-cube will be ***maximal***
  - That is, not contained within another sub-cube.
  - If it is properly contained in another sub-cube, then it could be replaced with that sub-cube.
  - The replacement would have fewer variables, and hence be simpler.
- Making a sub-cube as large as possible, makes the corresponding term as small as possible.

Maximality

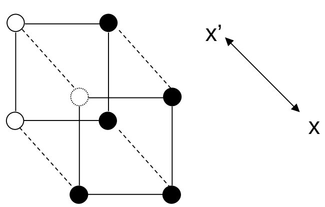


This set of 2 sub-cubes covers the function.

However, one of the sub-cubes is not maximal. Why?

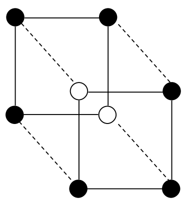
Einstein's Principle:

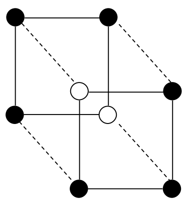
Explanations should be made as simple as possible, but no simpler.



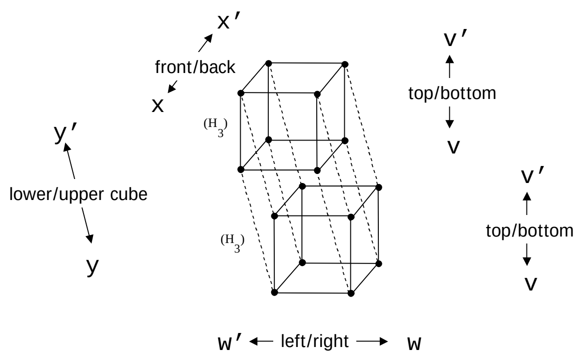
Including x by itself will not work!

Non-Uniqueness of Simplest Cover

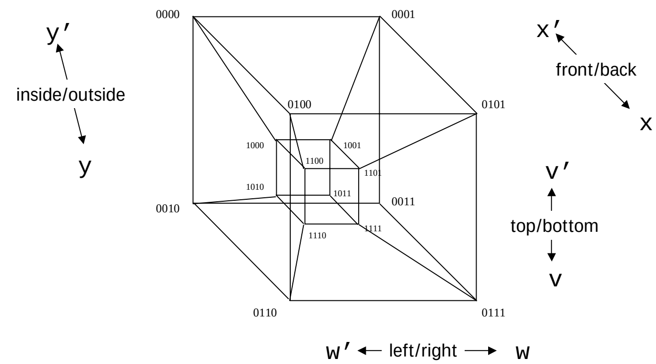




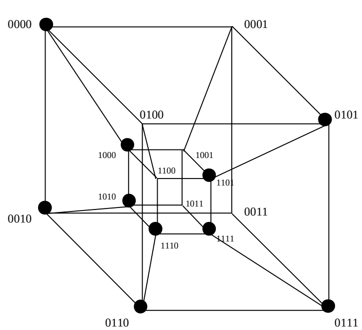
4-D Hypercube (Tesseract)



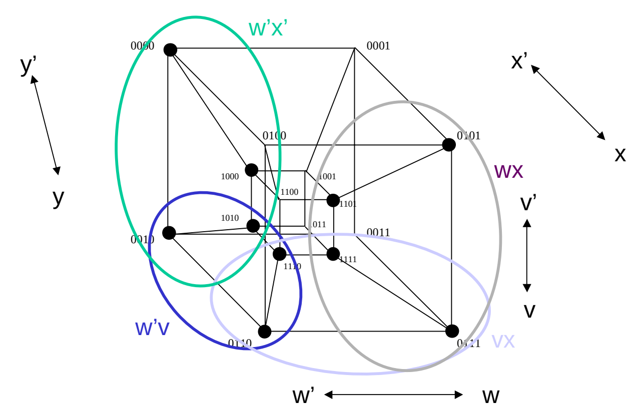
Alternate Representation



Plotting a Function on a 4-D Hypercube



Maximal Covering



Hypercube Function-Plotting Applet

An applet conceived by R. Keller and implemented by Ian Weiner, HMC ‘01 as a CS60 project.

http://www.cs.hmc.edu/~keller/javaExamples/hypercube

Sample inputs:

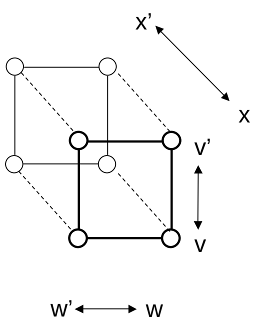
```
x*y+y*z
q*a+q'*b*c
```

Check out dimensions > 4.

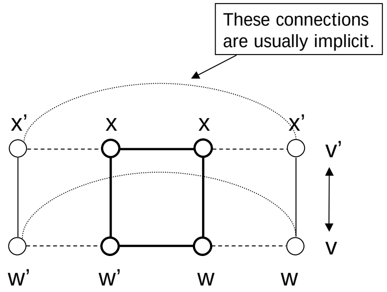
Karnaugh Maps

- Invented by Maurice Karnaugh, 1953, at Bell Labs
- A way to **visualize** hypercubes of up to 4 (and stretching to 5 or 6) dimensions
- Approach by “flattening” a hypercube
- A structured form of Venn Diagram

“3-D” Karnaugh Map

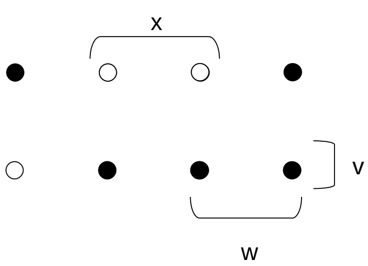


The Cube

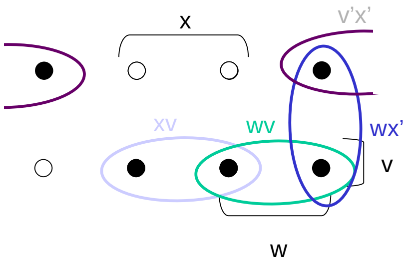


The Map

Covering on a Karnaugh Map

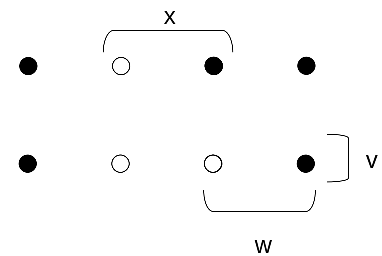


Covering on a Karnaugh Map

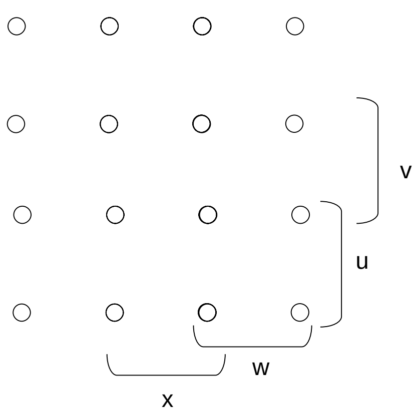


- "Subcubes" are now rectangles whose sides are powers of 2.
- Wraparound is permitted!

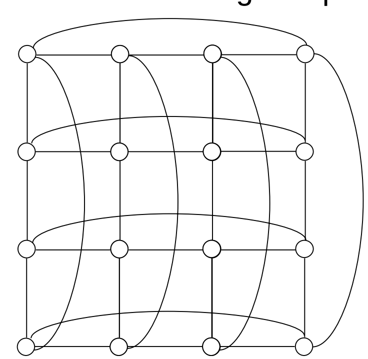
Try this one



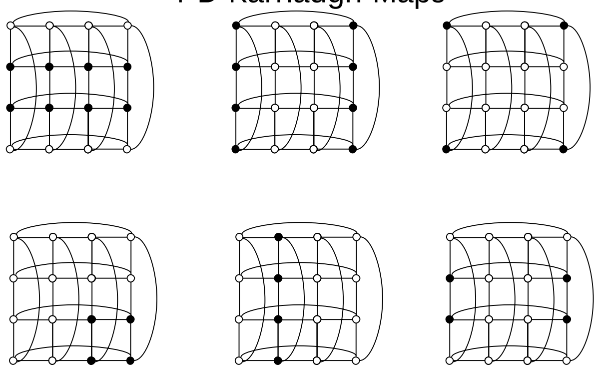
4-D Karnaugh Map



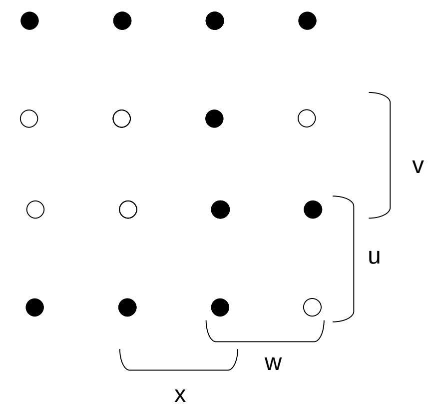
Implied connections on 4-D Karnaugh Map



Assorted Sub-Cubes on 4-D Karnaugh Maps



Try this one



Simplification with “Don’t Care” Combinations

- For certain problems, certain combinations (truth-table rows) never arise in actual operation.
- These are called “don’t care” combinations.
- Because their input combinations never occur, the function value can be either 0 or 1. This means we can ***elect*** which value to use at our discretion.
  - i.e., cover or not cover in the Karnaugh map, as convenient
- Usually we elect whichever value achieves the best simplification of the result.

Example: mod 3 adder

		uv 00	uv 01	uv 10
xy	00	0	0	1
xy	01	0	1	0
xy	10	1	0	0

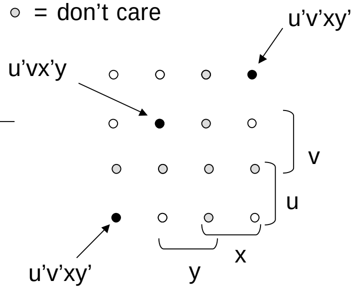
		uv 00	uv 01	uv 10
xy	00	0	1	0
xy	01	1	0	0
xy	10	0	0	1

Only 9 of 16 combinations actually occur, leaving 7 don’t care ones. The don’t cares are the same for both functions.

Plot Function on a K-Map

○ = don't care

xy \ uv	00	01	10
00	0	0	1
01	0	1	0
10	1	0	0

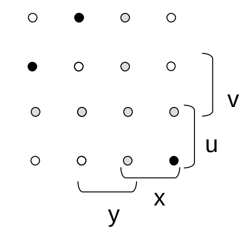


Simplified expression:

The other half.

○ = don't care

xy \ uv	00	01	10
00	0	1	0
01	1	0	0
10	0	0	1



Simplified expression: