CSEE 3827: Fundamentals of Computer Systems, Spring 2022

Lecture 3

Prof. Dan Rubenstein (danr@cs.columbia.edu)

Agenda (M&K 2.3-2.5)

- Standard Forms
  - Product-of-Sums (PoS)
  - Sum-of-Products (SoP)
    - converting between
  - Min-terms and Max-terms
- Simplification via Karnaugh Maps (K-maps)
  - 2, 3, and 4 variable
  - Implicants, Prime Implicants, Essential Prime Implicants
  - Using K-maps to reduce
  - PoS form
  - Don’t Care Conditions

Standard Forms

- There are many ways to express a boolean expression

$$
\begin{aligned}
F &= XYZ + XYZ + XZ \\
&= XY(Z + Z) + XZ \\
&= XZ
\end{aligned}
$$

- It is useful to have a standard or canonical way
- Derived from truth table
- Generally not the simplest (fewest literals) form

Two principle standard forms

- Sum-of-products (SoP)

- Product-of-sums (PoS)

- We will deal mostly in this course with SoP

Product and sum terms

- Product term: logical AND of literals (e.g., $X\overline{Y}Z$)
- Sum term: logical OR of literals (e.g., $A + \overline{B} + C$)

PoS & SoP

- Sum of products (SoP): OR of ANDs (OR of product terms)

$$\text{e.g., } F = \overline{Y} + \overline{X}Y\overline{Z} + XY$$

- Product of sums (PoS): AND of ORs (AND of sum terms)

$$\text{e.g., } G = X(\overline{Y} + Z)(X + Y + \overline{Z})$$

PoS and SoP not always simplest form

- e.g., F = ABD + ABE + C(D+E)
  - (AB+C) (D+E) is simplest (fewest literals) form (5 literals)
    - know it's simplest because each literal appears only once
  - simplest SoP form: ABD + ABE + CD + CE (10 literals)
  - simplest PoS form: (A+C)(B+C)(D+E) is (6 literals)

Converting any expression to SoP

- Just "multiply" through and simplify

- e.g., $G = X(Y + Z)(X + Y + Z)$
  - $= XYX + XYY + XYZ + XZX + XZY + XZZ$ and then simplify
  - $= XY + XY + XYZ + XZ + XZY + XZ$ (removed duplicate literals in prod. term)
  - $= XY + XZ$ (removed repeated prod. terms)

Converting from SoP to PoS

- Complement, multiply through, complement (swap + and · ops, flip literals)
- e.g., $F = \overline{Y}\,\overline{Z} + X\overline{Y}Z + XY\overline{Z}$

swap ops, flip literals

  - $\overline{F} = (Y+Z)(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + Z)$

simplify $\overline{F}$

  - $= YZ + \overline{X}Y + \overline{X}Z$ (after lots of simplifying)

swap ops, flip literals

  - $F = (\overline{Y}+\overline{Z})(X+\overline{Y})(X+\overline{Z})$
  - Why did this work?
    - Since $\overline{F} = YZ + \overline{X}Y + \overline{X}Z$ in SoP form
    - Complementing $\overline{F}$ in SoP form yields F in PoS form

Minterms and Maxterms

Minterms

e.g., Minterms for 3 variables A,B,C

A	B	C	minterm
0	0	0	m0 $\bar{A}\bar{B}\bar{C}$
0	0	1	m1 $\bar{A}\bar{B}C$
0	1	0	m2 $\bar{A}B\bar{C}$
0	1	1	m3 $\bar{A}BC$
1	0	0	m4 $A\bar{B}\bar{C}$
1	0	1	m5 $A\bar{B}C$
1	1	0	m6 $AB\bar{C}$
1	1	1	m7 ABC

2^3

- A product term in which all variables appear exactly once, either complemented or uncomplemented.
- Each minterm evaluates to 1 for exactly one assignment of values to all variables (one row in truth table), 0 for all others.
- e.g., for what values of A,B,C does $\bar{A}B\bar{C}$ evaluate to 1?

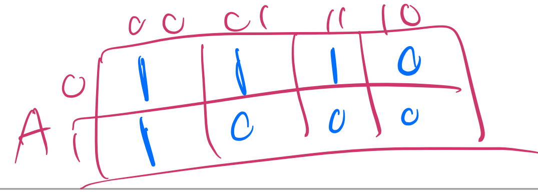
$\boxed{\bar{A}}\ \bar{B}\ \bar{C}$

0 0 0
0 0 1
0 1 1
1 0 0

$\Sigma(m0, m1, m3, m4) = 1$

BC

Minterms



e.g., Minterms for 3 variables A,B,C

A	B	C	minterm	
0	0	0	m0	$\bar{A}\bar{B}\bar{C}$
0	0	1	m1	$\bar{A}\bar{B}C$
0	1	0	m2	$\bar{A}B\bar{C}$
0	1	1	m3	$\bar{A}BC$
1	0	0	m4	$A\bar{B}\bar{C}$
1	0	1	m5	$A\bar{B}C$
1	1	0	m6	$AB\bar{C}$
1	1	1	m7	ABC

- A product term in which all variables appear exactly once, either complemented or uncomplemented.
- Each minterm evaluates to 1 for exactly one assignment of values to all variables (one row in truth table), 0 for all others.
- e.g., for what values of A,B,C does $\bar{A}\bar{B}C$ evaluate to 1?
- Ans: A=0, B=0, C=1 (that's the only one)

Minterms

e.g., Minterms for 3 variables A,B,C

A	B	C	minterm
0	0	0	m0 $\overline{A}\,\overline{B}\,\overline{C}$
0	0	1	m1 $\overline{A}\,\overline{B}C$
0	1	0	m2 $\overline{A}B\overline{C}$
0	1	1	m3 $\overline{A}BC$
1	0	0	m4 $A\overline{B}\,\overline{C}$
1	0	1	m5 $A\overline{B}C$
1	1	0	m6 $AB\overline{C}$
1	1	1	m7 ABC

- A product term in which all variables appear exactly once, either complemented or uncomplemented.
- Each minterm evaluates to 1 for exactly one assignment of values to all variables (one row in truth table), 0 for all others.
- Each product term denoted by mX where X corresponds to the row of the truth table where variable value assignments cause that midterm to equal 1.

Minterm examples (with 3 variables, A,B,C)

- ABC is a minterm, and is true when and only when A=1,B=1,C=1

- $\overline{A}B\overline{C}$ is a minterm and is true when and only when A=0,B=1, C=0

- A function can be described as a sum of its minterms
  - e.g., $F = B\,(A \oplus \overline{C}) = B(AC + \overline{A}\,\overline{C}) = ABC + \overline{A}B\overline{C}$ minterms
    - F=1 when ABC is true OR $\overline{A}B\overline{C}$ is true
      - i.e., F = 1 when A=1 & B=1 & C=1 OR when A=0 & B=1 & C=0
    - F = 0 otherwise

A few more Minterm examples (of 3 variables A,B,C)

- $G(A,B,C) = BC$ (G's value is independent of A)
  - $G = ABC + \bar{A}BC$ (G is defined over A,B,C, minterms contain all vars)
    - $G = 1$ when & only when $A=1,B=1,C=1$ OR $A=0,B=1,C=1$
- $H(A,B,C) = A$
  - $H = ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C}$ (all combos of B,C when A=1)
- $F = A + BC$
  - $= H + G = \underbrace{AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \rlap{$\overbrace{\phantom{ABC + \bar{A}BC}}^{BC}$}ABC}_{A} + \bar{A}BC$ (note minterm ABC repeated)

Minterms to describe a function

- *sometimes also called a **minterm expansion: OR (SUM) appropriate minterms together***

This “term” is TRUE when A=0,B=1,C=0

Hence F=1 when A=0,B=1,C=0

$$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$$

A	B	C	F	$\overline{F}$
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

$$\overline{F} = \overline{A}BC + AB\overline{C} + ABC$$

Function and its complement function use all midterms, share none in common

Sum of minterms form

- The logical OR of all minterms for which F = 1.

A	B	C	minterm	F
0	0	0	m0 $\bar{A}\bar{B}\bar{C}$	0
0	0	1	m1 $\bar{A}\bar{B}C$	1
0	1	0	m2 $\bar{A}B\bar{C}$	1
0	1	1	m3 $\bar{A}BC$	1
1	0	0	m4 $A\bar{B}\bar{C}$	0
1	0	1	m5 $A\bar{B}C$	0
1	1	0	m6 $AB\bar{C}$	0
1	1	1	m7 ABC	0

$$
\begin{aligned}
F &= \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC \\
&= m1 + m2 + m3 \\
&= \sum m(1,2,3)
\end{aligned}
$$

Minterm form cont'd

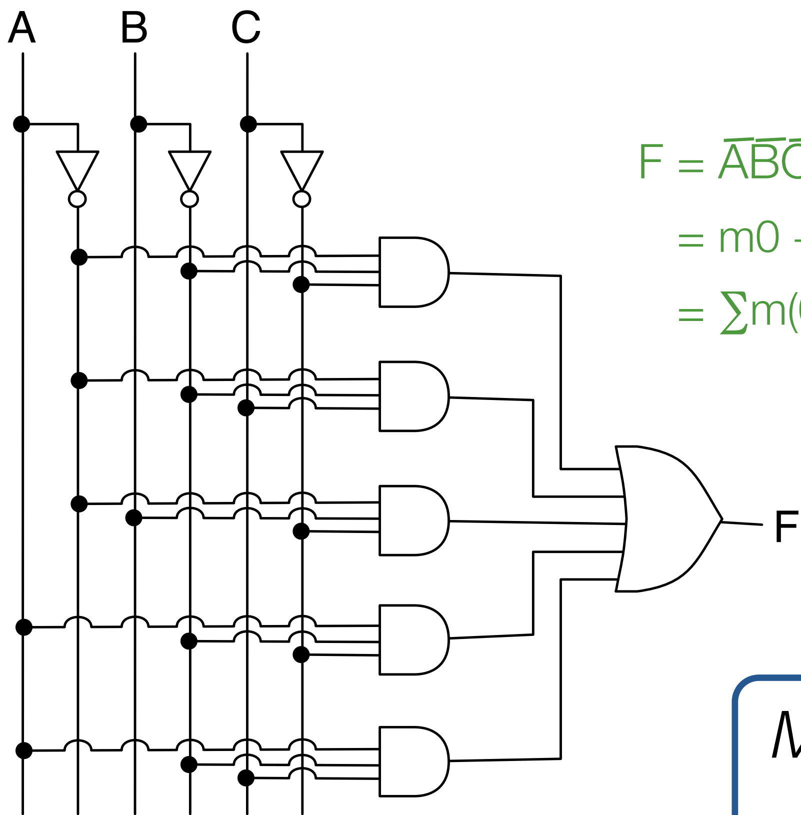
(variables appear once in each minterm)

A	B	C	F	$\overline{F}$	minterm
0	0	0	1	0	m0 $\overline{A}\,\overline{B}\,\overline{C}$
0	0	1	1	0	m1 $\overline{A}\,\overline{B}C$
0	1	0	1	0	m2 $\overline{A}B\overline{C}$
0	1	1	0	1	m3 $\overline{A}BC$
1	0	0	1	0	m4 $A\overline{B}\,\overline{C}$
1	0	1	1	0	m5 $A\overline{B}C$
1	1	0	0	1	m6 $AB\overline{C}$
1	1	1	0	1	m7 ABC

$$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$$
$$= m0 + m1 + m2 + m4 + m5$$
$$= \sum m(0,1,2,4,5)$$

$$\overline{F} = \overline{A}BC + AB\overline{C} + ABC$$
$$= m3 + m6 + m7$$
$$= \sum m(3,6,7)$$

Minterms as a circuit



$F = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$

$= m0 + m1 + m2 + m4 + m5$

$= \sum m(0,1,2,4,5)$

Minterm form is not simplified form!

Simplest Form v. SoP Form v. Minterm Form

- Can be the same, but not always

- e.g., $F = WX\,(Y+Z)$ (note 4 variables)
  - SoP form: $WXY + WXZ$
  - Minterm form: $WXYZ + WXY\overline{Z} + WX\overline{Y}Z$

- e.g., $F = WX\,(Y\overline{Z} + \overline{Y}Z)$
  - SoP form: $WX\overline{Y}Z + WXY\overline{Z}$
  - Minterm form: $WX\overline{Y}Z + WXY\overline{Z}$

Maxterms - "Dual" of minterms

A	B	C	maxterm	
0	0	0	M0	$A+B+C$
0	0	1	M1	$A+B+\overline{C}$
0	1	0	M2	$A+\overline{B}+C$
0	1	1	M3	$A+\overline{B}+\overline{C}$
1	0	0	M4	$\overline{A}+B+C$
1	0	1	M5	$\overline{A}+B+\overline{C}$
1	1	0	M6	$\overline{A}+\overline{B}+C$
1	1	1	M7	$\overline{A}+\overline{B}+\overline{C}$

- A sum term in which all variables appear once, either complemented or uncomplemented.
- Each maxterm evaluates to 0 for exactly one variable assignment, 1 for all others.
- Denoted by MX where X corresponds to the variable assignment for which MX = 0.

Maxterms: not as intuitive as minterms

- $F = \overline{A}\,\overline{C} + \overline{B}$
    - turns out F = 1 when (A=1 or B=0 or C=0) AND (A=0 or B=0 or C=1) AND (A=0 or B=0 or C=0),
    - i.e., $F = (A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)(\overline{A}+\overline{B}+\overline{C})$
- Think of it as forcing F to 0 when F doesn't equal a maxterm
    - e.g., F = 0 when A=1,B=1,C=1.
    - In other words, for F=1, it is necessary (but not sufficient) that either A=0 OR B=0 OR C=0, i.e., (A+B+C = 1) AND some other conditions
    - Thus, the maxterm $(\overline{A}+\overline{B}+\overline{C})$ is included in the sum for F

Maxterm description of a function

- *sometimes also called **conjunctive normal form** (CNF)*
- *sometimes also called a **maxterm expansion***

This "term" is FALSE when
A=1 & B=1 & C=0

$$F = (A+\overline{B}+\overline{C})\,(\overline{A}+\overline{B}+C)\,(\overline{A}+\overline{B}+\overline{C})$$

A	B	C	F	$\overline{F}$
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Force to 0

When inputs "satisfy" a Maxterm, the function equals 0

Product of maxterms form

- The logical AND of all maxterms for which F = 0.

A	B	C	maxterm	F
0	0	0	M0 $A+B+C$	0
0	0	1	M1 $A+B+\overline{C}$	1
0	1	0	M2 $A+\overline{B}+C$	1
0	1	1	M3 $A+\overline{B}+\overline{C}$	1
1	0	0	M4 $\overline{A}+B+C$	0
1	0	1	M5 $\overline{A}+B+\overline{C}$	0
1	1	0	M6 $\overline{A}+\overline{B}+C$	0
1	1	1	M7 $\overline{A}+\overline{B}+\overline{C}$	0

$$F = (A+B+C)\,(\overline{A}+B+C)\,(\overline{A}+B+\overline{C})\,(\overline{A}+\overline{B}+C)\,(\overline{A}+\overline{B}+\overline{C})$$

$$= (M0)\,(M4)\,(M5)\,(M6)\,(M7)$$

$$= \prod M(0,4,5,6,7)$$

Summary of Minterms and Maxterms

	F	$\overline{F}$
Minterms (SOP)	$\sum m(F = 1)$	$\sum m(F = 0)$
Maxterms (POS)	$\prod M(F = 0)$	$\prod M(F = 1)$

One final example

A	B	C	F	$\overline{F}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

	F	$\overline{F}$
Minterms (SOP)		
Maxterms (POS)		

Standard Form Example

A	B	C	F	$\overline{F}$	m/M
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	0	1	2
0	1	1	1	0	3
1	0	0	0	1	4
1	0	1	1	0	5
1	1	0	1	0	6
1	1	1	0	1	7

	F	$\overline{F}$
Sum of products (SOP)	$\sum m(1,3,5,6)$	$\sum m(0,2,4,7)$
Product of sums (POS)	$\prod M(0,2,4,7)$	$\prod M(1,3,5,6)$

Converting between canonical forms

	F	$\overline{F}$
Minterms (SOP)	$\sum m(F = 1)$	$\sum m(F = 0)$
Maxterms (POS)	$\prod M(F = 0)$	$\prod M(F = 1)$

DeMorgans: same terms

$$\overline{\sum m(F = 1)} = \prod M(F = 1)$$

Simplification Methods

Simplification Example

- Simplify $F = XY + X\overline{Y} + \overline{X}Y$

X	Y	F	m
0	0	0	$\overline{X}\overline{Y}$
0	1	1	$\overline{X}Y$
1	0	1	$X\overline{Y}$
1	1	1	XY

- $F = X\,(Y+\overline{Y}) + \overline{X}Y = \boxed{X + \overline{X}Y}$

- Can this be simplified further?

Simplification example cont'd

- $F = X + \overline{X}Y$

- Note important identity: $X + XY = X$ or… $X = X + XY$

  - e.g., X=female, Y = red hair: Say "yes" if X (you are female) or XY (you are female and have red hair) - it's enough just ask if female

  - so $F = X + \overline{X}Y = (X + \overline{X}Y) + XY = X + X\overline{Y} + XY = X + (\overline{X} + X)Y = \boxed{X + Y}$

- so most simplified, $F = X+Y$

  (just ask if female or have red hair)

The point: simplification not always so easy / obvious Additional tools are needed!

X	Y	F	m
0	0	0	$\overline{X}\,\overline{Y}$
0	1	1	$\overline{X}Y$
1	0	1	$X\overline{Y}$
1	1	1	XY

Karnaugh Maps

Karnaugh Maps (K-Maps)

- K-maps are a nice structure to help simplify functions in sum-of-product form (or product-of-sums form)
  - Gets functions simplified to either SoP or PoS form which is not necessarily absolute simplest, but it's good enough (for this course)
- We will use it to simplify functions to SoP form with up to 4 variables

Karnaugh maps (a.k.a., K-maps)

- All functions can be expressed with a K-map

- There is one square in the map for each minterm in a function's truth table

X	Y	F
0	0	m0
0	1	m1
1	0	m2
1	1	m3

X \ Y	0	1
0	m0 $\overline{X}\,\overline{Y}$	m1 $\overline{X}Y$
1	m2 $X\overline{Y}$	m3 XY

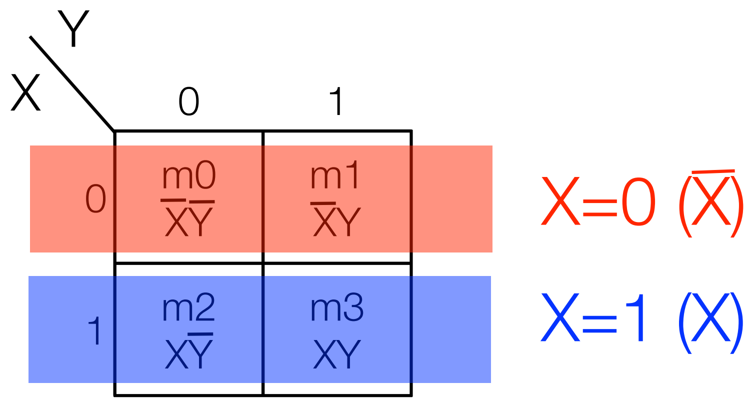
A K-map is just a 2-dimensional way of representing the function in a truth table

Karnaugh maps

- All functions can be expressed with a map

- There is one square in the map for each minterm in a function’s truth table

X	Y	F
0	0	m0
0	1	m1
1	0	m2
1	1	m3



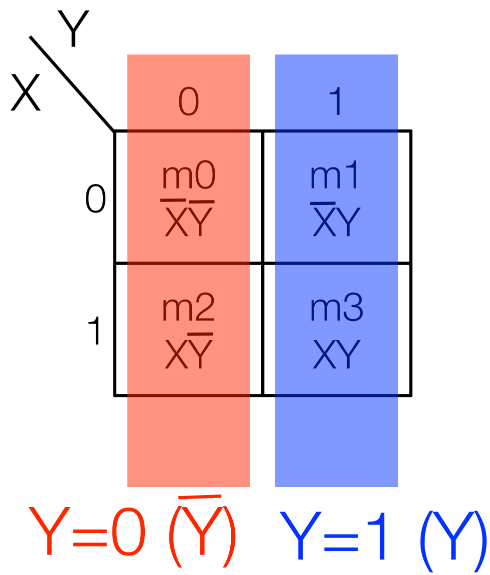
A K-map is just a 2-dimensional way of representing the function in a truth table

Karnaugh maps

- All functions can be expressed with a map

- There is one square in the map for each minterm in a function's truth table

X	Y	F
0	0	m0
0	1	m1
1	0	m2
1	1	m3



Karnaugh maps express functions

- Fill out table with value of a function

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

X \ Y	0	1
0	0	1
1	1	1

Karnaugh maps express functions

- Fill out table with value of a function

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

X \ Y	0	1
0	0	1
1	1	1

X	Y	m	F
0	0	m0	0
0	1	m1	1
1	0	m2	1
1	1	m3	1

$$F = \sum m(1,2,3) = \overline{X}Y + X\overline{Y} + XY$$

Karnaugh maps express functions

- Fill out table with value of a function

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

X \ Y	0	1
0	0	1
1	1	1

X	Y	m	F
0	0	m0	0
0	1	m1	1
1	0	m2	1
1	1	m3	1

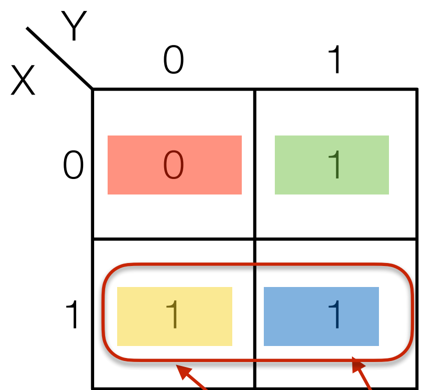
$$F=\sum m(1,2,3) = \bar{X}Y + \boxed{X\bar{Y} + XY}$$

Karnaugh maps express functions

- Fill out table with value of a function

$$X\overline{Y} + XY = X$$

To simplify, when both midterms XY and XY evaluate to 1, they can be "replaced" in the expression by X

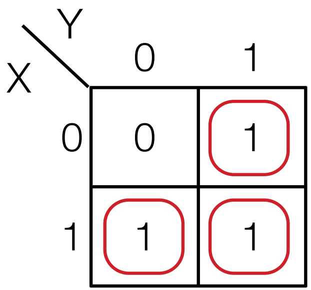


X	Y	m	F
0	0	m0	0
0	1	m1	1
1	0	m2	1
1	1	m3	1

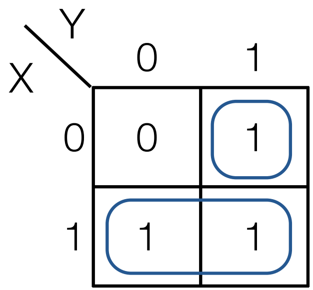
$$F=\sum m(1,2,3) = \overline{X}Y + \boxed{X\overline{Y} + XY}$$

Simplification using a k-map

- Whenever two squares share an edge and both are 1, those two terms can be combined to form a single term with one less variable

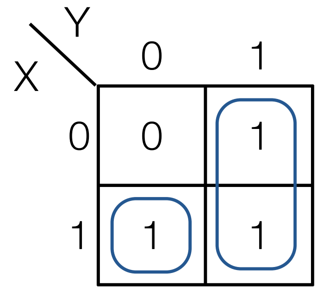


$F = \overline{X}Y + X\overline{Y} + XY$



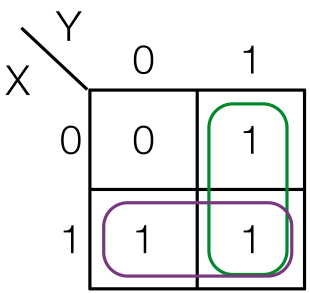
$F = X + \overline{X}Y$

(combined $X\overline{Y} + XY = X$)



$F = Y + X\overline{Y}$

(combined $\overline{X}Y + XY = Y$)

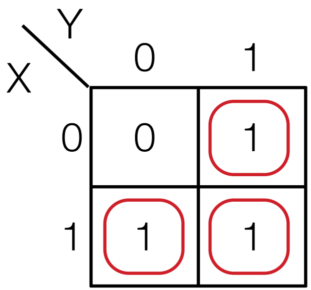


(combined $X\overline{Y} + XY = X$ and $\overline{X}Y + XY = Y$)

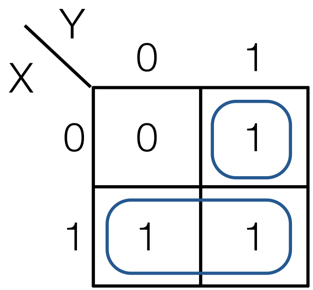
$F = X + Y$

Simplification using a k-map

- Whenever two squares share an edge and both are 1, those two terms can be combined to form a single term with one less variable

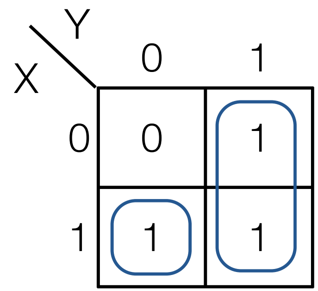


$F = \overline{X}Y + X\overline{Y} + XY$



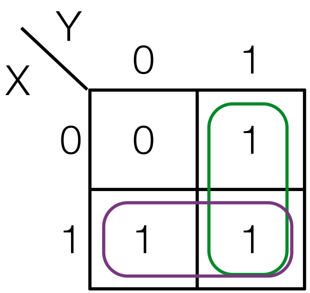
$F = X + \overline{X}Y$

(combined $X\overline{Y} + XY = X$)



$F = Y + X\overline{Y}$

(combined $\overline{X}Y + XY = Y$)



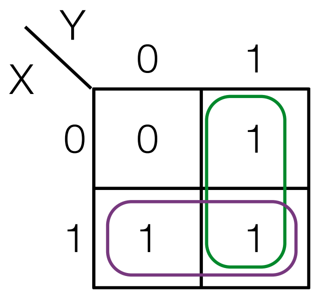
(combined $X\overline{Y} + XY = X$ and $\overline{X}Y + XY = Y$)

$F = X + Y$

(= $X\overline{Y} + XY + \overline{X}Y + XY$): XY term “repeated” to obtain simplified form

Simplification using a k-map (2)

- “Circle” contiguous squares of 1s (# of squares covered must be a power of 2)
- There is a correspondence between circles on a k-map and terms in a function expression
- The bigger the circle, the simpler the term
- Add circles (and terms) until all 1s on the k-map are circled



$$F = X + Y$$

3-variable Karnaugh maps

- Use gray ordering on edges with multiple variables
- Gray encoding: order of values such that only one bit changes at a time
- Two minterms are considered adjacent if they differ in only one variable (this means maps “wrap”)

X \ Y Z	0 0	0 1	1 1	1 0
0	m0 $\bar{X}\bar{Y}\bar{Z}$	m1 $\bar{X}\bar{Y}Z$	m3 $\bar{X}YZ$	m2 $\bar{X}Y\bar{Z}$
X=1 1	m4 $X\bar{Y}\bar{Z}$	m5 $X\bar{Y}Z$	m7 XYZ	m6 $XY\bar{Z}$

Or just

X

3-variable Karnaugh maps

- Use gray ordering on edges with multiple variables
- Gray encoding: order of values such that only one bit changes at a time
- Two minterms are considered adjacent if they differ in only one variable (this means maps “wrap”)

	X \ Y Z	0 0	0 1	1 1	1 0
X=0	0	m0 $\overline{X}\,\overline{Y}\,\overline{Z}$	m1 $\overline{X}\,\overline{Y}Z$	m3 $\overline{X}YZ$	m2 $\overline{X}Y\overline{Z}$
	1	m4 $X\overline{Y}\,\overline{Z}$	m5 $X\overline{Y}Z$	m7 XYZ	m6 $XY\overline{Z}$

Or just

$$\overline{X}$$

3-variable Karnaugh maps

- Use gray ordering on edges with multiple variables
- Gray encoding: order of values such that only one bit changes at a time
- Two minterms are considered adjacent if they differ in only one variable (this means maps “wrap”)

Y=1

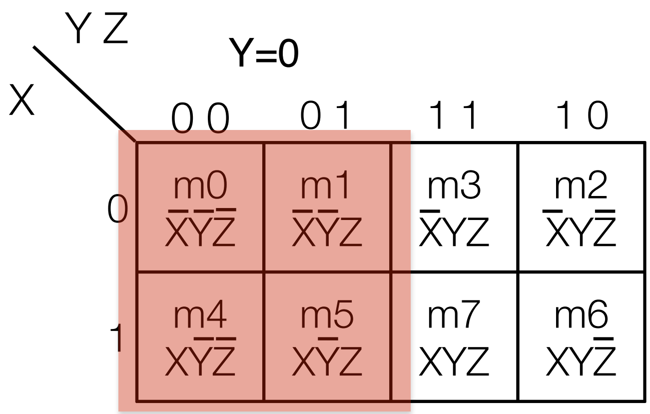
X \ Y Z	0 0	0 1	1 1	1 0
0	m0 $\overline{X}\,\overline{Y}\,\overline{Z}$	m1 $\overline{X}\,\overline{Y}Z$	m3 $\overline{X}YZ$	m2 $\overline{X}Y\overline{Z}$
1	m4 $X\overline{Y}\,\overline{Z}$	m5 $X\overline{Y}Z$	m7 XYZ	m6 $XY\overline{Z}$

Or just

Y

3-variable Karnaugh maps

- Use gray ordering on edges with multiple variables
- Gray encoding: order of values such that only one bit changes at a time
- Two minterms are considered adjacent if they differ in only one variable (this means maps “wrap”)



Or just

$$\overline{Y}$$

3-variable Karnaugh maps

- Use gray ordering on edges with multiple variables
- Gray encoding: order of values such that only one bit changes at a time
- Two minterms are considered adjacent if they differ in only one variable (this means maps “wrap”)

X \ YZ	0 0	0 1	1 1	1 0
0	m0 $\overline{X}\,\overline{Y}\,\overline{Z}$	m1 $\overline{X}\,\overline{Y}Z$	m3 $\overline{X}YZ$	m2 $\overline{X}Y\overline{Z}$
1	m4 $X\overline{Y}\,\overline{Z}$	m5 $X\overline{Y}Z$	m7 XYZ	m6 $XY\overline{Z}$

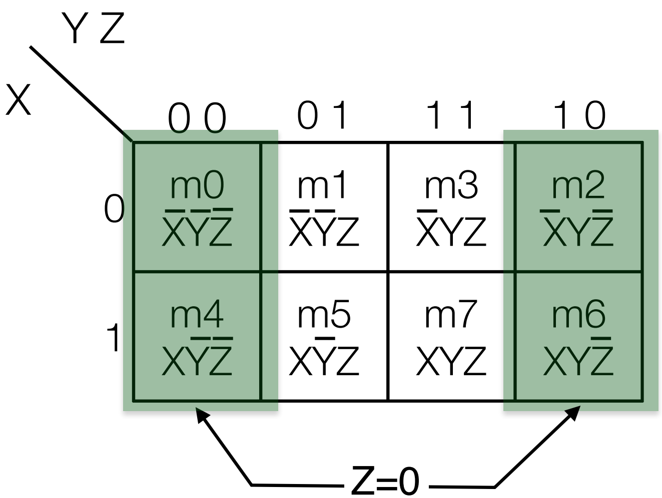
Z=1

Or just

Z

3-variable Karnaugh maps

- Use gray ordering on edges with multiple variables
- Gray encoding: order of values such that only one bit changes at a time
- Two minterms are considered adjacent if they differ in only one variable (this means maps “wrap”)

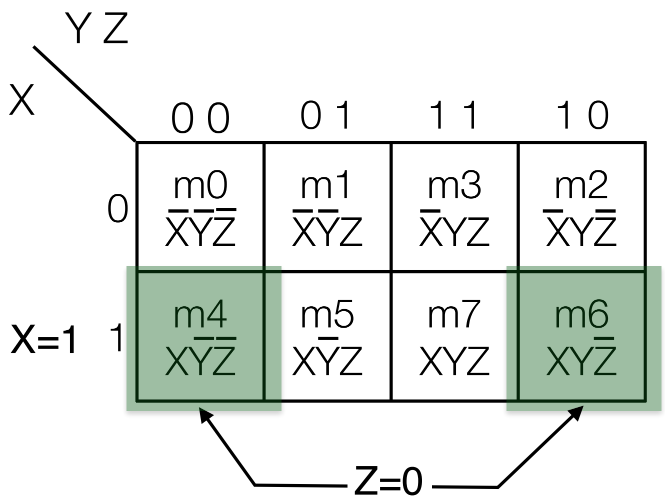


Or just

$$\overline{Z}$$

3-variable Karnaugh maps

- Use gray ordering on edges with multiple variables
- Gray encoding: order of values such that only one bit changes at a time
- Two minterms are considered adjacent if they differ in only one variable (this means maps “wrap”)



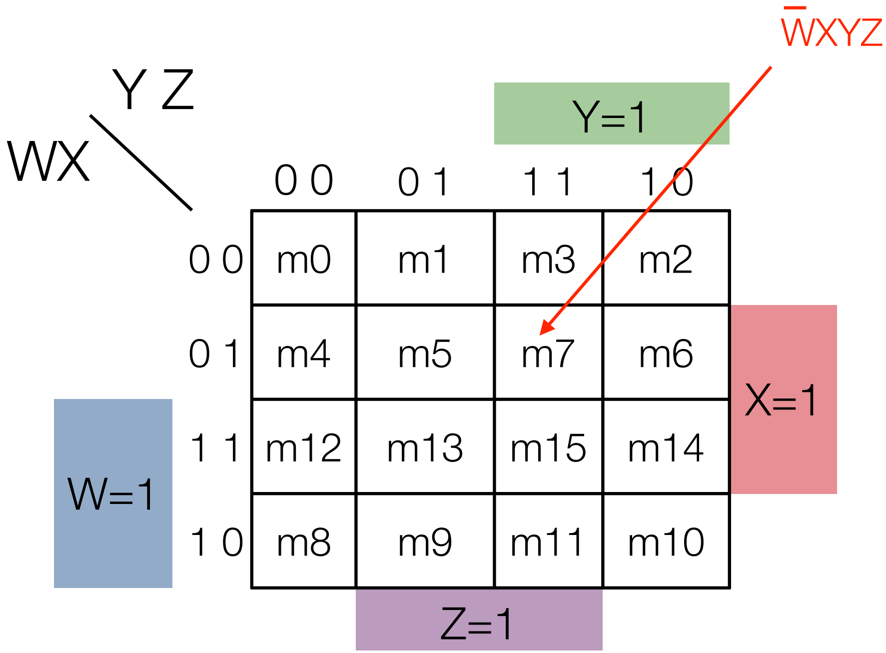
X=1 and Z=0

Or just

$$X\overline{Z}$$

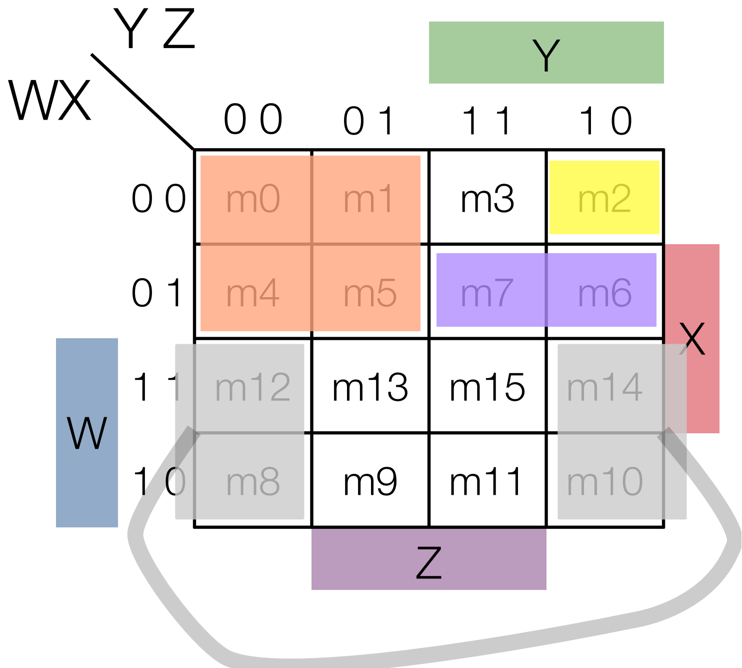
4-variable Karnaugh maps

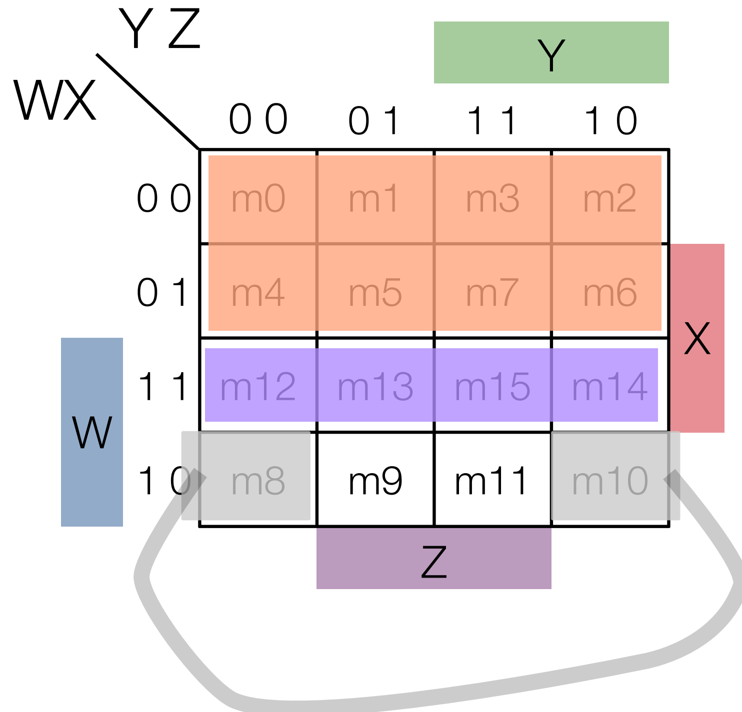
- Extension of 3-variable maps



Product terms are just 2^i x 2^j boxes (that might wrap around)

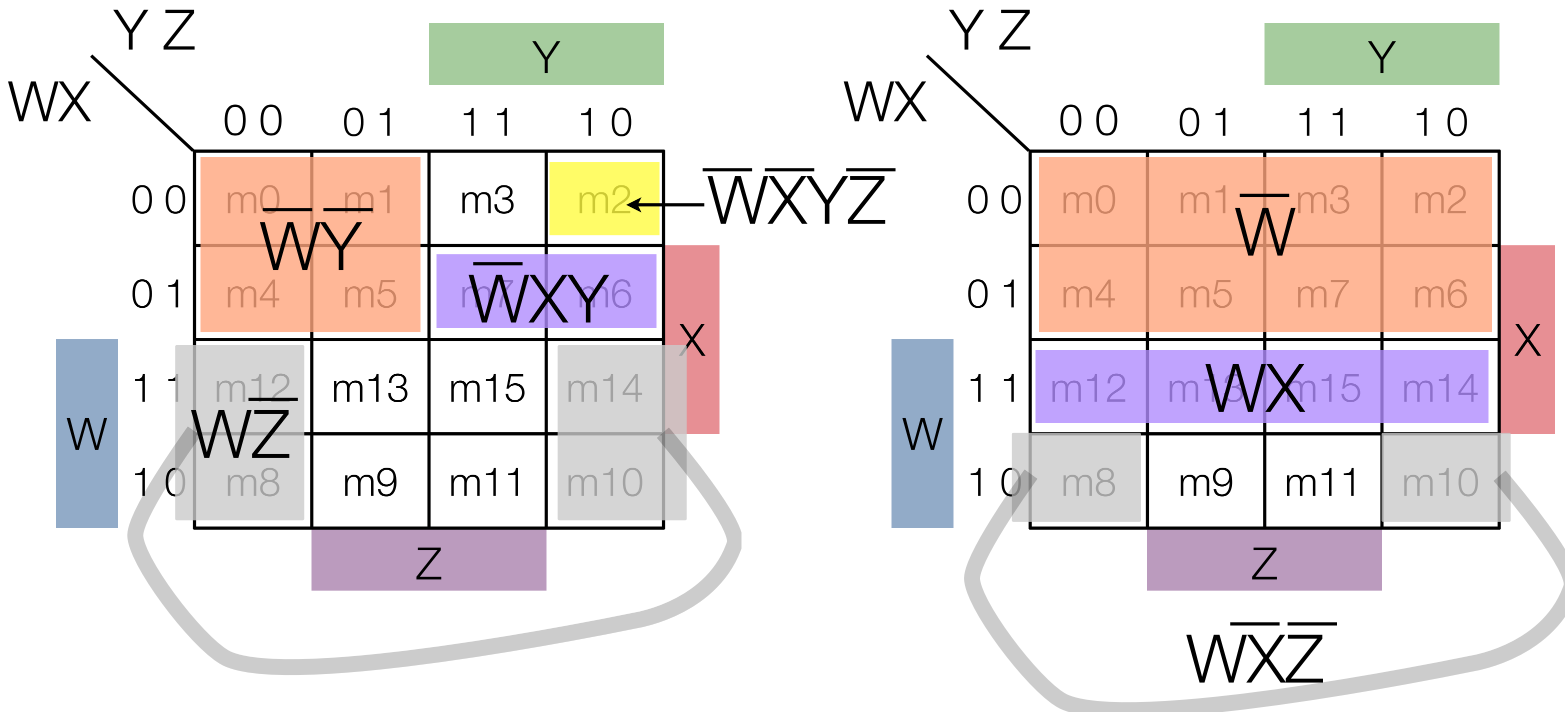
- A product term, which, viewed in a K-Map is a 2^i x 2^j size "rectangle" (possibly wrapping around) where i=0,1,2, j=0,1,2





Product terms are just 2^i x 2^j boxes (that might wrap around)

- A product term, which, viewed in a K-Map is a 2^i x 2^j size "rectangle" (possibly wrapping around) where i=0,1,2, j=0,1,2



Note: bigger rectangles = fewer literals

4-variable Karnaugh maps example

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

WX \ YZ	0 0	0 1	1 1	1 0
0 0	1	1	0	1
0 1	1	1	0	1
1 1	1	1	0	0
1 0	1	1	0	1

Y

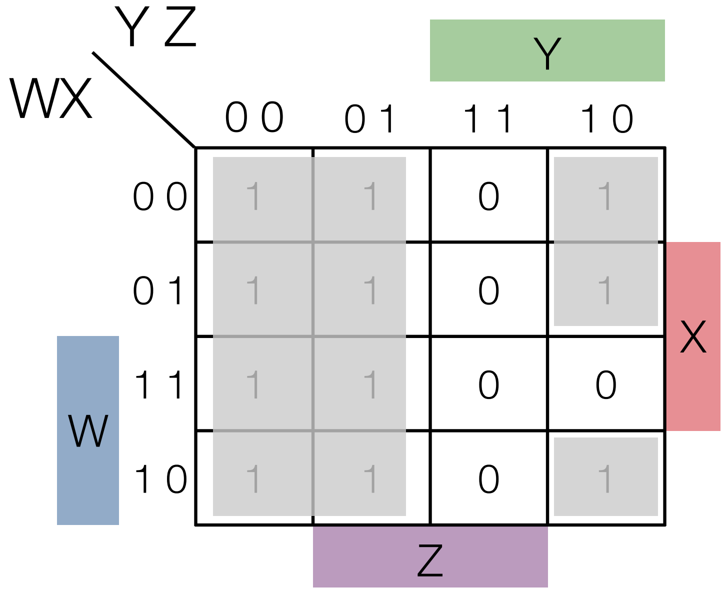
X

W

Z

4-variable Karnaugh maps example

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



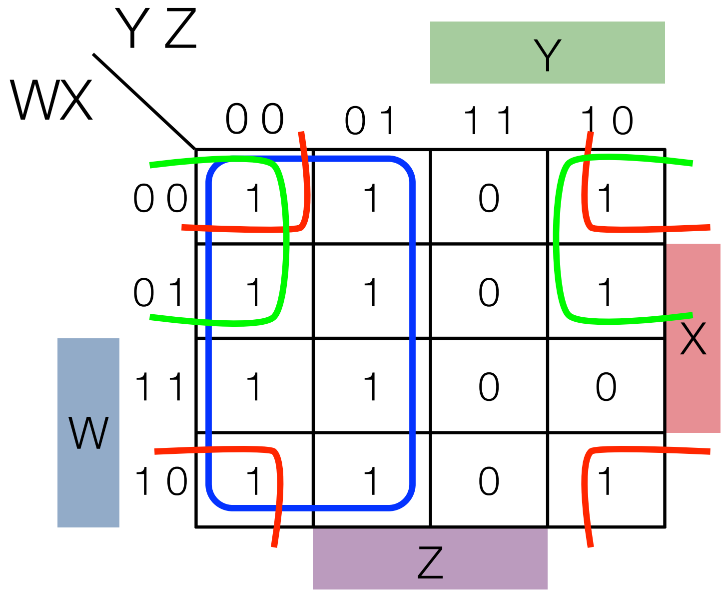
K-maps make F expressed as SoP easy to see, e.g.,

$$\bar{Y} + \bar{W}Y\bar{Z} + W\bar{X}Y\bar{Z}$$

Can the expression for F be simplified further?

4-variable Karnaugh maps example

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



$$\overline{Y} + \overline{W}\,\overline{Z} + \overline{X}\,\overline{Z}$$

Rule when picking product terms: Must cover only 1's, but OK to overlap. Bigger rectangles are better (fewer literals)

Implicant terminology for a function F

- Given a function F:
  - An **implicant** is a product term (i.e., a “box” whose dimensions are powers of 2) comprised of minterms for which the function F evaluates to 1
    - i.e., the “box” must only cover squares that are 1
- **prime implicant**: An implicant not contained within another implicant (remember that implicant dimensions must be powers of 2!)
- **essential prime implicant**: a **prime implicant** that is the **only prime implicant** to cover some minterm.

4-variable Karnaugh maps (3)

- List all all of the prime implicants for this function
- Is any of them an essential prime implicant?
- What is a simplified expression for this function?

	WX \ YZ	0 0	0 1	1 1	1 0	
				Y		
	0 0	0	0	1	0	
	0 1	1	1	1	0	X
W	1 1	0	1	1	1	
	1 0	0	1	0	0	
			Z			

4-variable Karnaugh maps (3)

- List all all of the prime implicants for this function
- Is any of them an essential prime implicant?
- What is a simplified expression for this function?

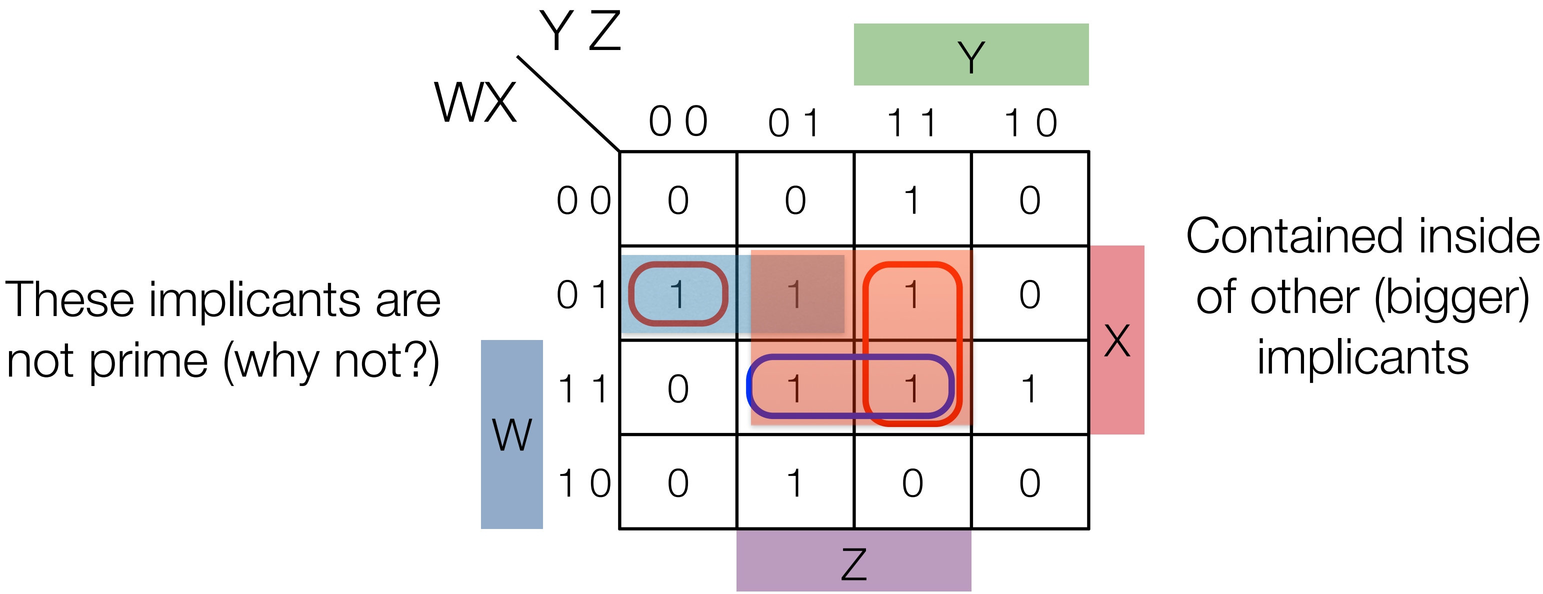
WX \ YZ	0 0	0 1	1 1	1 0
0 0	0	0	1	0
0 1	1	1	1	0
1 1	0	1	1	1
1 0	0	1	0	0

Y (columns 1 1, 1 0); X (rows 0 1, 1 1); W (rows 1 1, 1 0); Z (columns 0 1, 1 1)

These implicants are not prime (why not?)

4-variable Karnaugh maps (3)

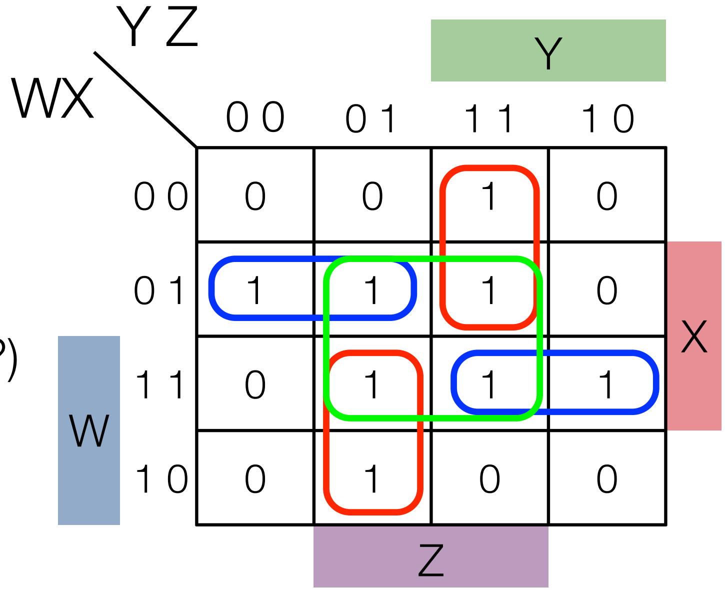
- List all all of the prime implicants for this function
- Is any of them an essential prime implicant?
- What is a simplified expression for this function?



4-variable Karnaugh maps (3)

- List all of the prime implicants for this function
- Is any of them an essential prime implicant?
- What is a simplified expression for this function?

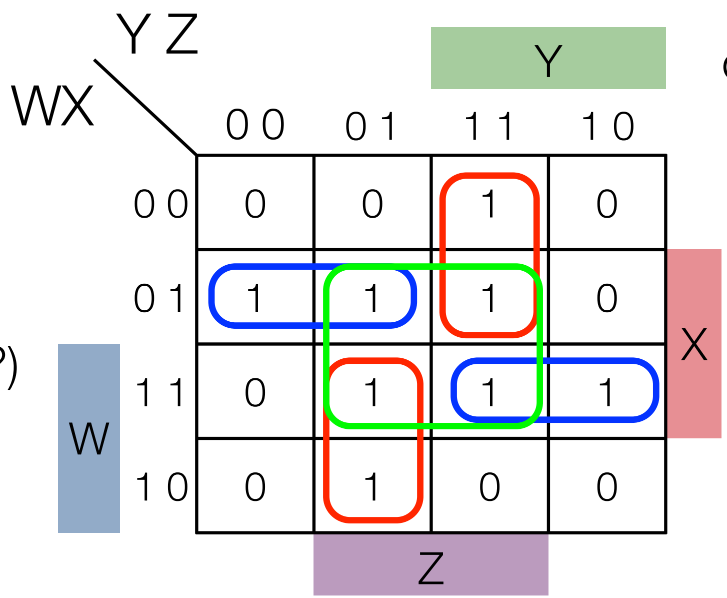
Prime implicants
(which are essential?)



4-variable Karnaugh maps (3)

- List all of the prime implicants for this function
- Is any of them an essential prime implicant?
- What is a simplified expression for this function?

Prime implicants (which are essential?)



The blue and red are essential: each has 1 minterm not covered by any other prime implicant

Green not essential: every covered minterm also covered by blue or red prime implicant

Using K-maps to build simplified circuits

- Step 1: Identify all PIs and essential PIs
- Step 2: Include all Essential PIs in the circuit (Why?)
- Step 3: If any 1-valued minterms are uncovered by EPIs, choose PIs that are "big" and do a good job covering
  - **Selection Rule:** a heuristic for usually choosing "good" PIs: choose the PIs that minimize overlap with one another and with EPIs

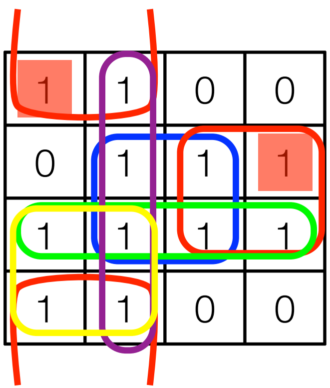
1	1	1	0
0	1	1	0
1	1	1	1
1	1	0	1

1	1	0	0
0	1	1	0
0	0	1	1
1	0	0	1

Using K-maps to build simplified circuits

- Step 1: Identify all PIs and essential PIs
- Step 2: Include all Essential PIs in the circuit (Why?)
- Step 3: If any 1-valued minterms are uncovered by EPIs, choose PIs that are “big” and do a good job covering
- **Selection Rule**: a heuristic for usually choosing “good” PIs: choose the PIs that minimize overlap with one another and with EPIs

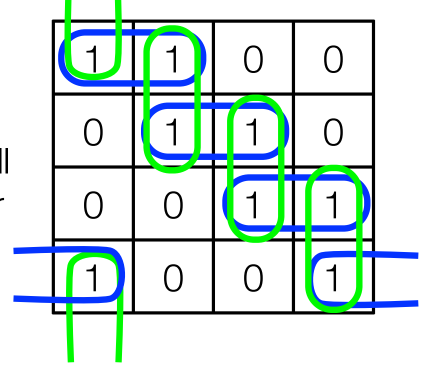
Red bounds are EPIs (solo-covered minterm shown in red)



Also need (purple or blue) and (yellow or green)

All blue PIs or all green PIs cover

No EPIs!



Design example : 2-bit multiplier

a_1	a_0	b_1	b_0	z_3	z_2	z_1	z_0
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

two 2-bit #'s multiplied together to give a 4-bit solution

e.g., $a_1a_0 = 10$, $b_1b_0 = 11$, $z_3z_2z_1z_0 = 0110$

Design example : 2-bit multiplier (SOLUTION)

a1	a0	b1	b0	z3	z2	z1	z0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Design example : 2-bit multiplier (SOLUTION)

a1	a0	b1	b0	z3	z2	z1	z0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

b_0 (top), a_0 (right), a_1 (left), b_1 (bottom)

0	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0

$z_3 = a_1 a_0 b_1 b_0$

b_0 (top), a_0 (right), a_1 (left), b_1 (bottom)

0	0	0	0
0	0	0	0
0	0	0	1
0	0	1	1

$z_2 = a_1 \overline{a_0} b_1 + a_1 b_1 \overline{b_0}$

$z_1 =$ (exercise)

b_0 (top), a_0 (right), a_1 (left), b_1 (bottom)

0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0

$z_0 = a_0 b_0$

b_0 (top), a_0 (right), a_1 (left), b_1 (bottom)

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

K-Maps: Complements, PoS, don’t care conditions

Finding $\overline{F}$

- Find prime implicants corresponding to the 0s on a k-map

WX \ YZ	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	1	0	0
10	1	1	0	1

$$F = \overline{Y} + \overline{X}\,\overline{Z} + \overline{W}\,\overline{Z}$$

WX \ YZ	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	0	1	1
10	0	0	1	0

$$\overline{F} = YZ + WXY$$

PoS expressions from a k-map

- Find $\overline{F}$ as SoP and then apply DeMorgan's
- or: Cover the boxes with all "0"s - the sum term "0's out" that box

WX \ YZ	0 0	0 1	1 1	1 0
0 0	1	1	0	1
0 1	1	0	0	0
1 1	1	0	0	0
1 0	1	1	0	1

$$\overline{F} = YZ + XZ + YX$$

DeMorgan's

$$F = (\overline{Y}+\overline{Z})(\overline{Z}+\overline{X})(\overline{Y}+\overline{X})$$

Don’t Care Conditions

Don't care conditions

- There are circumstances in which the value of an output doesn't matter
- For example, in that 2-bit multiplier, what we
- are told neither a nor b will be input as 0
  - "don't care" what the output looks like for the input cases that will not occur
- Don't care situations are denoted by an "X" in a truth table and in Karnaugh maps.
- Can also be expressed in minterm form:

$$z2 = \sum m(10,11,14)$$
$$d2 = \sum m(0,1,2,3,4,8,12)$$

a1	a0	b1	b0	z3	z2	z1	z0
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	X	X	X	X
0	1	0	0	X	X	X	X
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	X	X	X	X
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	X	X	X	X
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Simple Don’t Care Example

- Let F = AB + $\overline{A}\,\overline{B}$
- Suppose we know the input combo A=1, B=0 will never occur
- Can we replace F with a simpler function G whose output matches for all inputs we do care about?
- Let H be the function with Don’t-care conditions for obsolete inputs

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Simple Don't Care Example

- Let F = $AB + \overline{A}\,\overline{B}$
- Suppose we know the input combo A=1, B=0 will never occur
- Can we replace F with a simpler function G whose output matches for all inputs we do care about?

Inputs will not occur

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Since A=1,B=0 will not occur, don't care what circuit produces for A=1,B=0 inputs

Simple Don't Care Example

- Let $F = AB + \overline{A}\,\overline{B}$
- Suppose we know the input combo A=1, B=0 will never occur
- Can we replace F with a simpler function G whose output matches for all inputs we do care about?
- Let H be the function with Don't-care conditions for obsolete inputs

A	B	F	H
0	0	1	1
0	1	0	0
1	0	0	X
1	1	1	1

Inputs will not occur

Simple Don't Care Example

- Let $F = AB + \bar{A}\bar{B}$
- Suppose we know the input combo A=1, B=0 will never occur
- Can we replace F with a simpler function G whose output matches for all inputs we do care about?
- Let H be the function with Don't-care conditions for obsolete inputs

Inputs will not occur

A	B	F	H	G
0	0	1	1	1
0	1	0	0	0
1	0	0	X	1
1	1	1	1	1

Simple Don't Care Example

- Let $F = AB + \overline{A}\,\overline{B}$
- Suppose we know the input combo A=1, B=0 will never occur
- Can we replace F with a simpler function G whose output matches for all inputs we do care about?
- Let H be the function with Don't-care conditions for obsolete inputs

Inputs will not occur

A	B	F	H	G
0	0	1	1	1
0	1	0	0	0
1	0	0	X	1
1	1	1	1	1

- Both F & G are appropriate functions for H

Simple Don’t Care Example

- Let $F = AB + \overline{A}\,\overline{B}$
- Suppose we know the input combo A=1, B=0 will never occur
- Can we replace F with a simpler function G whose output matches for all inputs we do care about?
- Let H be the function with Don’t-care conditions for obsolete inputs

A	B	F	H	G
0	0	1	1	1
0	1	0	0	0
1	0	0	X	1
1	1	1	1	1

Inputs will not occur

- Both F & G are appropriate functions for H

$$G = AB + \overline{B}$$

- G is (slightly) simpler than F (3 instead of 4 literals), and gets the job done!

2-bit multiplier non-0 multiplier (SOLUTION)

a1	a0	b1	b0	z3	z2	z1	z0
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	X	X	X	X
0	1	0	0	X	X	X	X
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	X	X	X	X
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	X	X	X	X
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

b_0

X	X	X	X
X	0	0	0
X	0	1	0
X	0	0	0

a_0 a_1 b_1

$z_3 = a_1 a_0 b_1 b_0$

b_0

X	X	X	X
X	0	0	0
X	0	0	1
X	0	1	1

a_0 a_1 b_1

$z_2 = a_1\overline{b_0} + \overline{a_0}b_1$

(vs. $a_1\overline{a_0}b_1 + a_1 b_1\overline{b_0}$)

Revised rule for "implicant":

1's must be covered

0's must not be covered

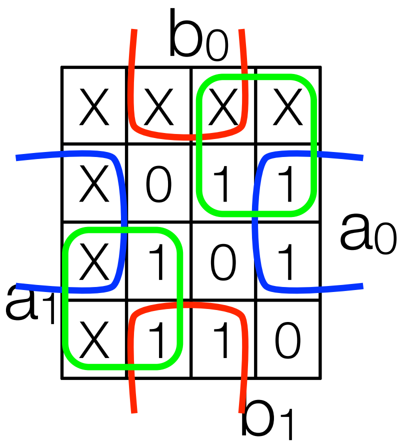
X's are optionally covered

2-bit multiplier non-0 multiplier (SOLUTION)

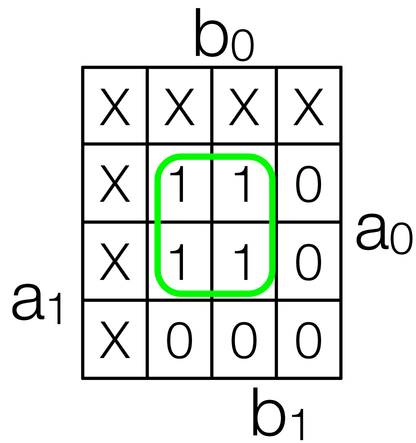
a1	a0	b1	b0	z3	z2	z1	z0
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	X	X	X	X
0	1	0	0	X	X	X	X
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	X	X	X	X
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	X	X	X	X
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Still have prime and essential prime implicants

z_1 = (exercise)



$z_0 = a_0 b_0$



All above prime implicants are essential

Final thoughts on Don’t care conditions

- Sometimes “don’t cares” greatly simplify circuitry

D

	1	X	X	X
	X	1	X	X	B
	0	0	1	X	
A	0	0	X	1	

C

$$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + ABCD + A\overline{B}C\overline{D} \quad \text{vs. } \overline{A} + C$$

High-level Summary

- minterm (of k variables): a product term formed from the literals of each variable (exist without function)

$$A B \bar{C}$$

- A function (e.g., for $F = A + \bar{A}BC$) can be constructed by ORing (summing) together the minterms for which the function = 1

$$ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC$$

- implicant (of a function): a product term (of literals) that is “contained” within the function (formed from minterms where the function = 1)

$$AB$$

- prime implicant: a product term which, if any literal is remove, is no longer “contained” within the function

- essential prime implicant: when a function is expressed as an OR (sum) of prime implicants, this one must be included.