CSEE 3827: Fundamentals of Computer Systems, Spring 2011

2. Boolean Logic & Algebra

Prof. Martha Kim (martha@cs.columbia.edu)
Web: http://www.cs.columbia.edu/~martha/courses/3827/sp11/

Contents (H&H 2.1-2.7, 2.9)

- Boolean Algebra
  - AND, OR, NOT
  - DeMorgan's
  - Duals
- Logic Gates
- NAND, NOR, XOR

- Standard Forms
  - Product-of-Sums (PoS)
  - Sum-of-Products (SoP)
    - conversion 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

Terminology

- Recall: Digital / Binary / Boolean: 0 = False, 1 = True
- Binary Variable: a symbolic representation of a value that might be 0 or 1, e.g., X, Y, A, B
- Complement (e.g., of a variable X): written $\overline{X}$: the opposite value of X

X	$\overline{X}$
0	1
1	0

- Literal: a boolean variable or its complement (e.g., X, $\overline{X}$, $\overline{Y}$)

Boolean Logic

- All logical functions can be implemented in terms of three logical operations:

NOT

x	$\overline{x}$
0	1
1	0

AND

can omit the "·"

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Logic 2

- Precedence rules just like decimal system
- Implied precedence: NOT > AND > OR
- Use parentheses as necessary

$AB + C$ same as $(AB) + C$

$(\overline{A} + B)C$ same as $((\overline{A}) + B)C$

Terminology cont'd

- Expression: a set of literals (possibly with repeats) combined with logic operations (and possibly ordered by parentheses)
  - e.g., 4 expressions: $AB + C$, $(AB) + C$, $(\overline{A} + B)C$, $((\overline{A}) + B)C$
  - Note: can compliment expressions, too, e.g., $\overline{((\overline{A}) + B)C}$
- Equation: expression1 = expression2
  - e.g., $(\overline{A} + B)C = ((\overline{A}) + B)C$
- Function of (possibly several) variables: an equation where the lefthand side is defined by the righthand side $F(A,B,C) = ((\overline{A}) + B)C$

Boolean Logic: Example

Truth Table: all combinations of input variables
k variables ➜ 2^k input combinations

D	X	A	$\overline{DX} + A$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Boolean Logic: Example

D	X	A	$\overline{X}$	$D\overline{X}$	$D\overline{X} + A$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	1

(M&K Table 2-2)

Boolean Logic: Example 2

X	Y	$XY + \overline{\overline{X}\overline{Y}}$
0	0	
0	1	
1	0	
1	1	

Boolean Algebra: Identities and Theorems

OR	AND	NOT	
$X+0 = X$	$X1 = X$		(identity)
$X+1 = 1$	$X0 = 0$		(null)
$X+X = X$	$XX = X$		(idempotent)
$X+\overline{X} = 1$	$X\overline{X} = 0$		(complementarity)
		$\overline{\overline{X}} = X$	(involution)
$X+Y = Y+X$	$XY = YX$		(commutativity)
$X+(Y+Z) = (X+Y)+Z$	$X(YZ) = (XY)Z$		(associativity)
$X(Y+Z) = XY + XZ$	$X+YZ = (X+Y)(X+Z)$		(distributive)
$\overline{X+Y} = \overline{X}\,\overline{Y}$	$\overline{XY} = \overline{X} + \overline{Y}$		(DeMorgan's theorem)

Boolean Algebra: Example

Simplify this equation using algebraic manipulation.

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

Boolean Algebra: Example

Simplify this equation using algebraic manipulation.

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

$\overline{X}Y(Z + \overline{Z}) + XZ$ (by reverse distribution)

$\overline{X}Y1 + XZ$ (by complementarity)

$\overline{X}Y + XZ$ (by identity)

Boolean Algebra: Example 2

Find the complement of F.

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

$$\overline{F} =$$

Boolean Algebra: Example 2

Find the complement of F.

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

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

$$(\overline{A\overline{B}})(\overline{\overline{A}B}) \qquad \text{(by DeMorgan's)}$$

$$(\overline{A} + \overline{\overline{B}})(\overline{\overline{A}} + \overline{B}) \qquad \text{(by DeMorgan's)}$$

$$(\overline{A} + B)(A + \overline{B}) \qquad \text{(by involution)}$$

DeMorgan's Theorem

- Procedure for complementing expressions
- Remove the "big bar" over AND or OR of 2 (or more) functions (e.g., F & G) and replace...
  - AND with OR, OR with AND
  - 1 with 0, 0 with 1
  - function F with $\overline{F}$, $\overline{F}$ with F

$$\overline{FG} = \overline{F} + \overline{G}$$

$$\overline{F + G} = \overline{F}\,\overline{G}$$

DeMorgan's Practice

$$\overline{\overline{A\overline{B}C} + \overline{ACD} + B\overline{C}}$$

DeMorgan's Practice

$$\overline{\overline{A\overline{B}C} + \overline{ACD} + B\overline{C}}$$

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

$= (A\overline{B}C)(ACD)(\overline{B\overline{C}})$

$(A\overline{B}C)(ACD) = A\overline{B}CD$, $F = B$, $G = \overline{C}$, $\overline{FG} = \overline{F}+\overline{G}$

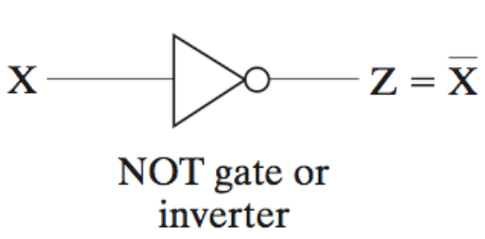
$= (A\overline{B}CD)(\overline{B}+C)$

$= A\overline{B}CD + A\overline{B}CD$

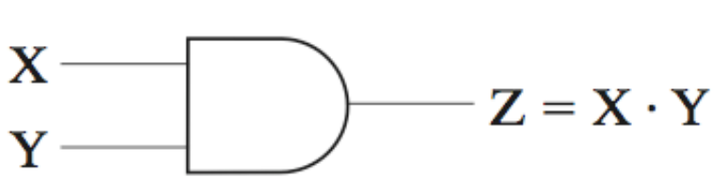
$= A\overline{B}CD$

Circuit Representation

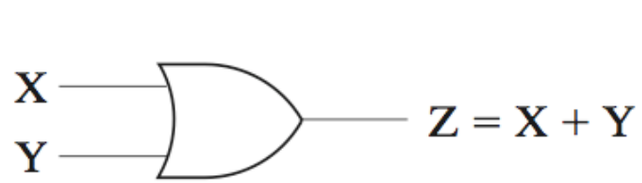
- Information flows from left to right
- Input(s) all the way on the left, output(s) on the right



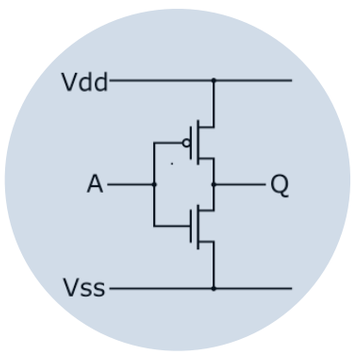
NOT gate or inverter

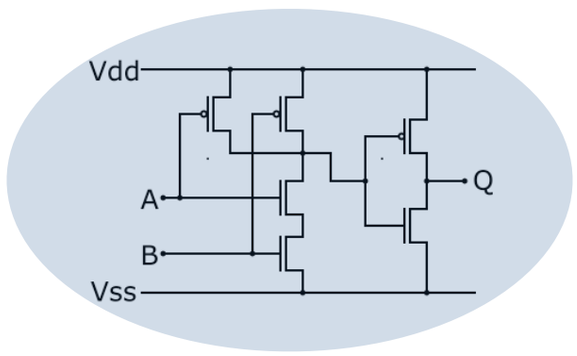


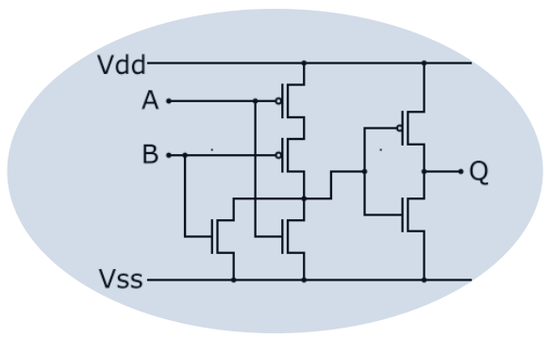
AND gate



OR gate







These circuits consume area, power, and time

Goal: minimize the amount of circuitry to compute the desired function

We simplify to reduce required circuitry...

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

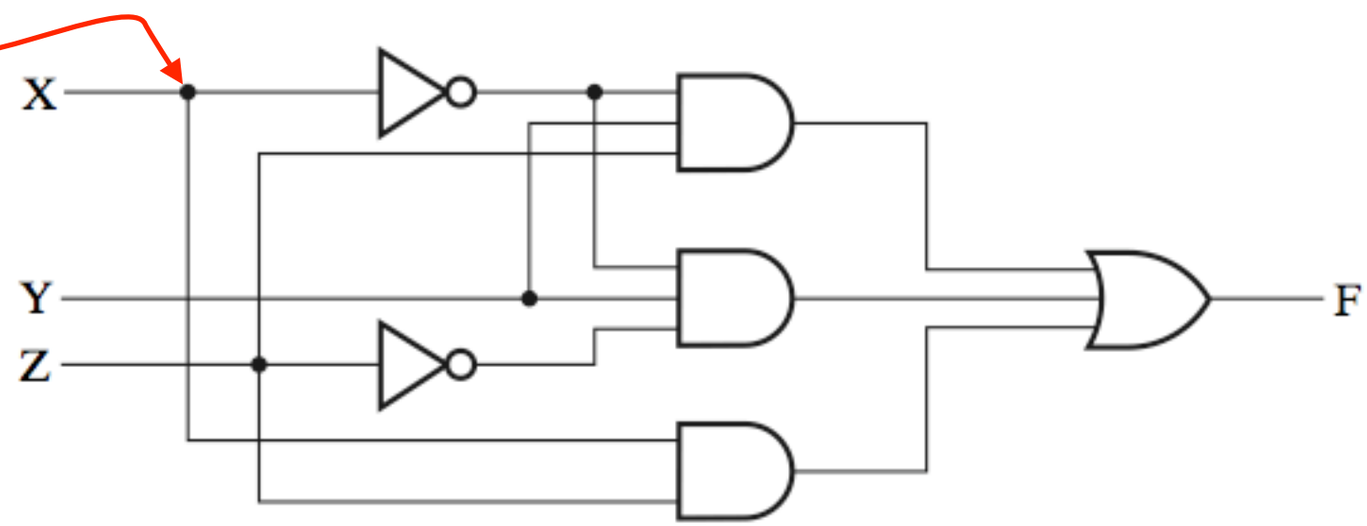
$\overline{X}Y(Z + \overline{Z}) + XZ$ (by reverse distribution)

$\overline{X}Y1 + XZ$ (by complementarity)

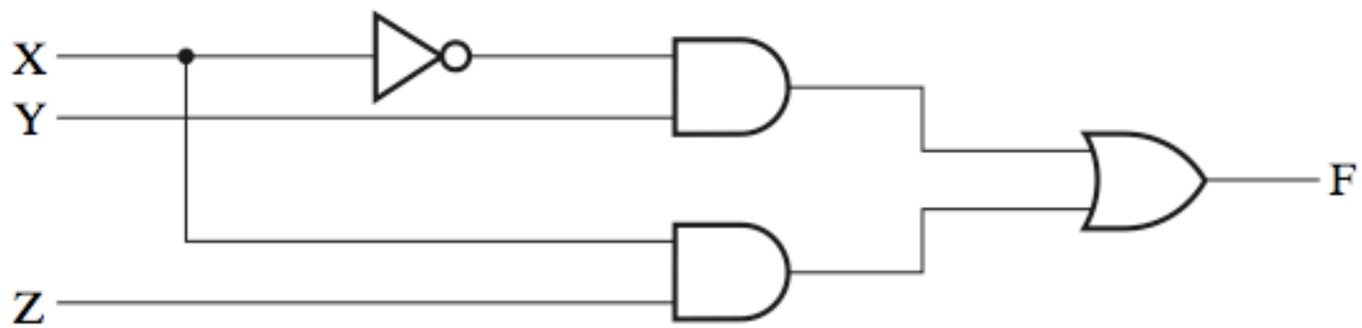
$\overline{X}Y + XZ$ (by identity)

Circuit view

wire connector: black dot signifies wires are connected



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



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

Universal gates: NAND, NOR

x	y	$z = \overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

Note: the "o" in a circuit represents a NOT (inverter)

x	y	$z = \overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

Different from "●" which represents wire connector

NAND and NOR universal because...

- NOT, AND, OR can each be implemented using only NAND gates
- NOT, AND, OR can each be implemented using only NOR gates

$\overline{A} = A \text{ NAND } A$	$\overline{A} = A \text{ NOR } A$
$AB = \overline{A \text{ NAND } B}$	$A+B = \overline{A \text{ NOR } B}$
$A+B = \overline{A} \text{ NAND } \overline{B}$	$AB = \overline{A} \text{ NOR } \overline{B}$

Duals

Duals

- All boolean expressions have duals
- Any theorem you can prove, you can also prove for its dual
- To form a dual...
  - replace AND with OR, OR with AND
  - replace 1 with 0, 0 with 1

What is the dual of this expression?

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

What is the dual of this expression?

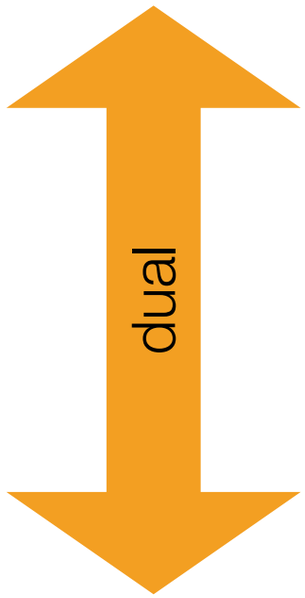
$$\overline{X} + \overline{Y} = \overline{XY}$$

dual

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

What are the complements of these expressions?

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



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

What are the complements of these expressions?

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

complement

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

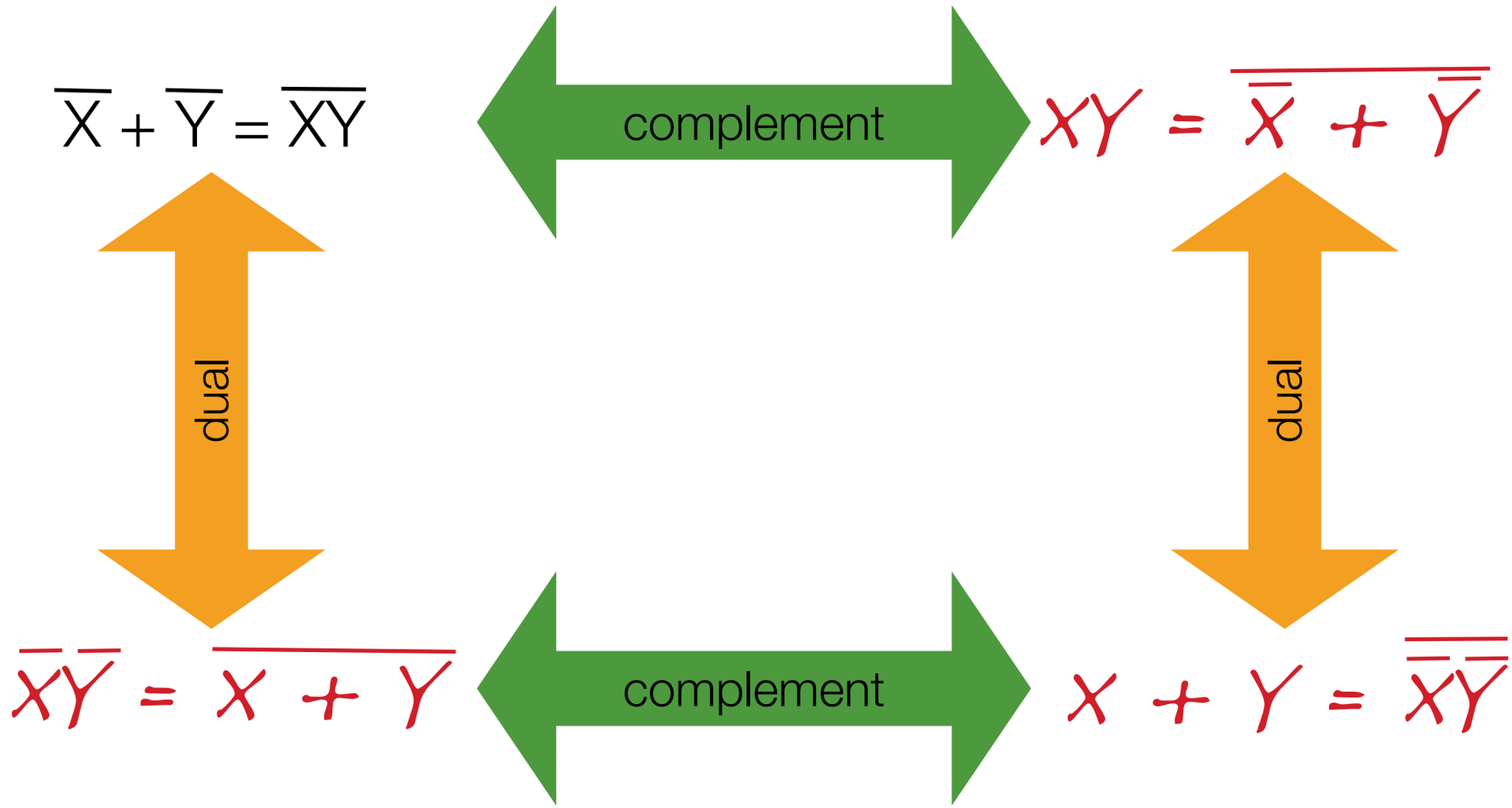
dual

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

complement

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

These are also the duals of one another.



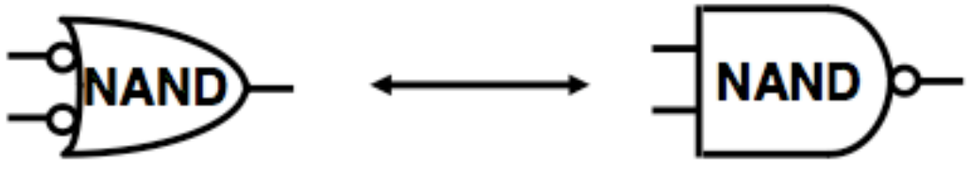
Note: to complement a function, compute its dual and complement literals

“Complement using Dual” example

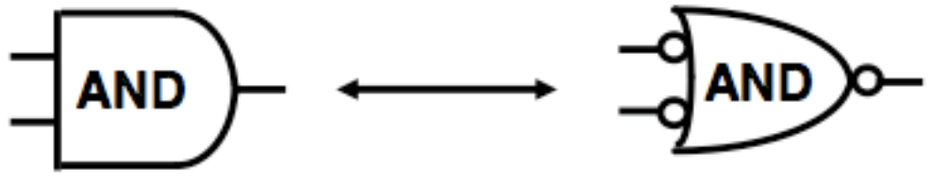
- $F = X + A\,(Z + \bar{X}\,(Y + W) + \bar{Y}\,(Z + W))$
- Dual: $F_{dual} = X\,(A + Z\,(\bar{X} + YW)(\bar{Y} + ZW))$
- $\bar{F} = \bar{X}\,(\bar{A} + \bar{Z}\,(X + \bar{Y}\bar{W})(Y + \bar{Z}\bar{W}))$

Can be used for gate manipulation.

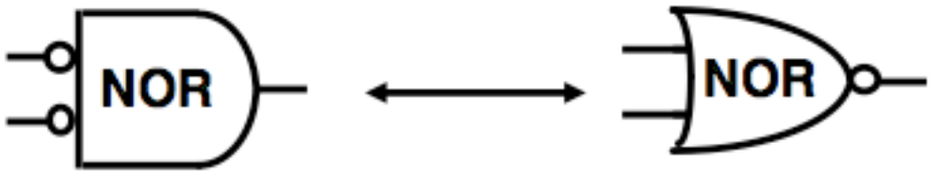
$$\overline{X} + \overline{Y} = \overline{XY}$$



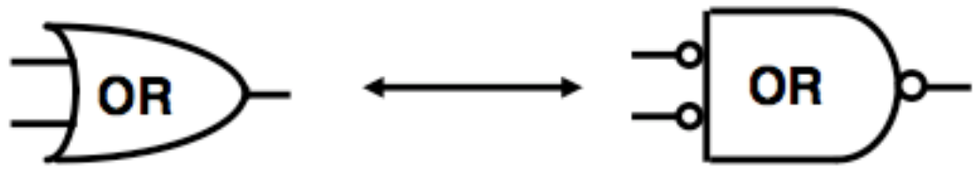
$$XY = \overline{\overline{X} + \overline{Y}}$$



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

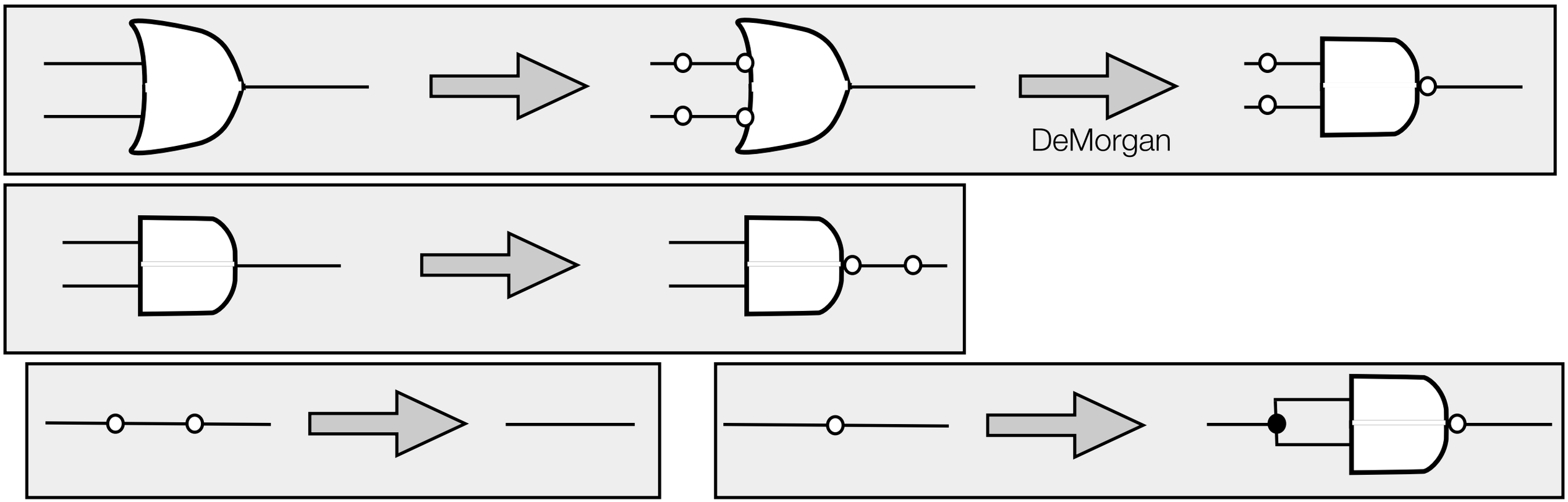


$$X + Y = \overline{\overline{X}\,\overline{Y}}$$

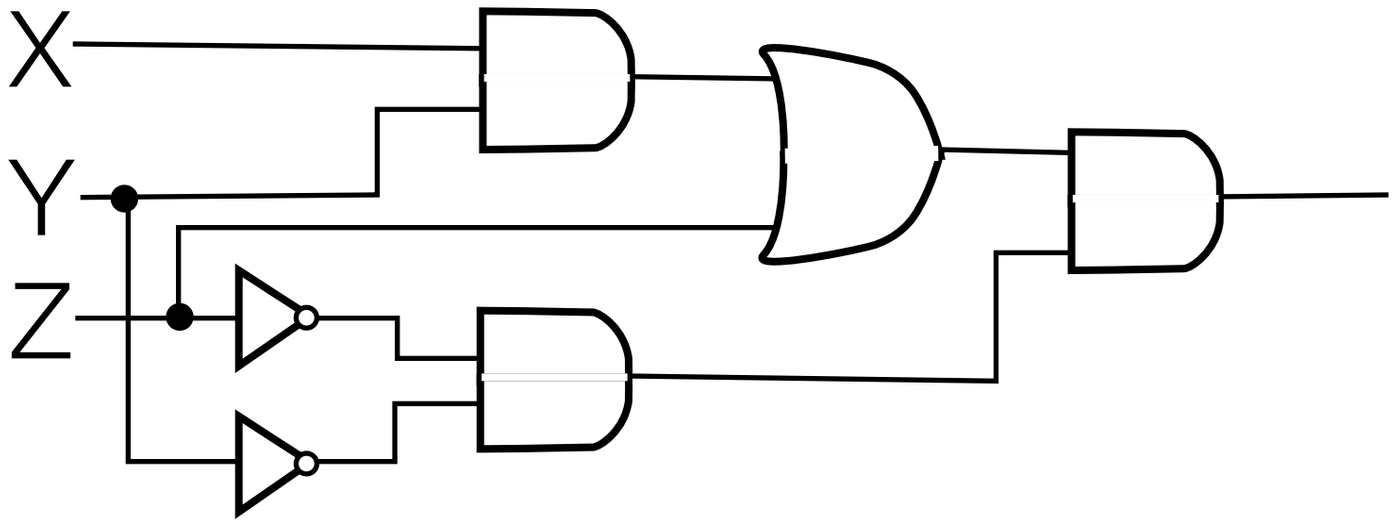


Converting circuits to all-NAND (or all-NOR)

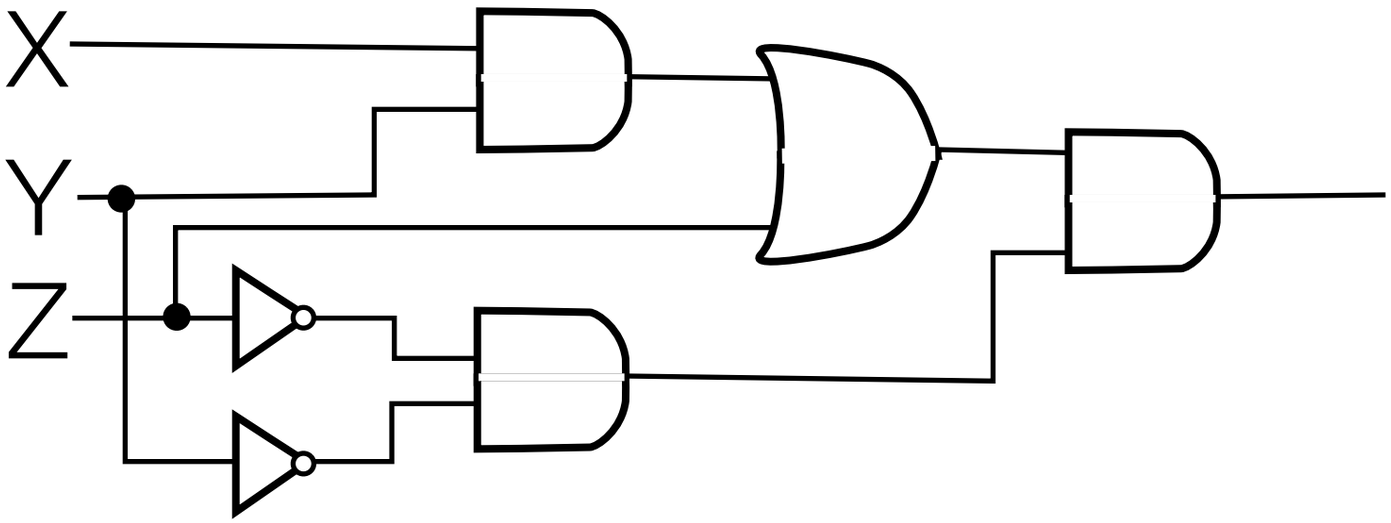
- Work from right to left
- When manipulating an (AND or OR) gate, stick in pairs of NOT gates to get it in “appropriate” form
- Isolated NOT gates are easily implemented as a NAND (NOR) gate
- example manipulations (for NAND gates)

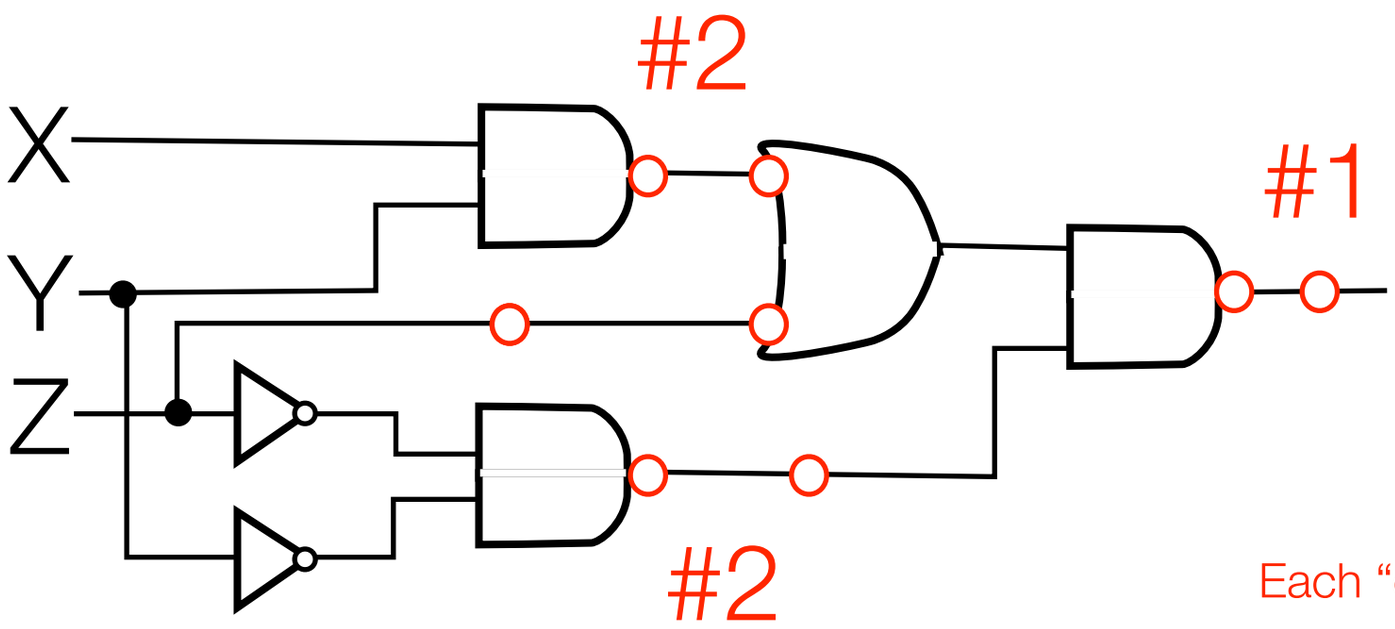


Convert-to-all-NAND example



Convert-to-all-NAND example





Each “o” by itself represents a NOT gate

XOR: the parity operation

- $X \oplus Y = X\bar{Y} + \bar{X}Y$

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

- In general, represents parity, i.e.,

- $X_1 \oplus X_2 \oplus X_3 \oplus ... \oplus X_k = 1$ when an odd number of $X_i = 1$

Standard Forms

Standard Forms

- There are many ways to express a boolean expression

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

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

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

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

Two principle standard forms

- Sum-of-products (SOP)
- Product-of-sums (POS)

Terminology

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

PoS & SoP

- Sum of products (SoP): OR of ANDs

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

- Product of sums (PoS): AND of ORs

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

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 from PoS (or any form) to SoP

Just multiply through and simplify, e.g.,

$$G = X(Y + Z)(X + Y + Z)$$

$$= XYX + XYY + XYZ + XZX + XZY + XZZ$$

$$= XY + XY + XYZ + XZ + XZY + XZ$$

$$= XY + XZ$$

Converting from SoP to PoS

Complement, multiply through, complement via DeMorgan, e.g.,

Note: X' = $\overline{X}$

$$F = Y'Z' + XY'Z + XYZ'$$

$$F' = (Y+Z)(X'+Y+Z')(X'+Y'+Z)$$

$$= YZ + X'Y + X'Z \quad \text{(after lots of simplification)}$$

$$F = (Y'+Z')(X+Y')(X+Z')$$

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 once, either complemented or uncomplemented (i.e., an entry in the truth table).
- Each minterm evaluates to 1 for exactly one variable assignment, 0 for all others.
- Denoted by mX where X corresponds to the variable assignment for which mX = 1.

Minterms to describe a function

*sometimes also called a **minterm expansion** or **disjunctive normal form (DNF)***

This "term" is TRUE 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

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

Minterm example, seen another way

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

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

m0		m1		m2	m3	m4		m5	m6	m7
1	+	0	+	0	0	0	+	0	0	0
0	+	1	+	0	0	0	+	0	0	0
0	+	0	+	1	0	0	+	0	0	0
0	+	0	+	0	1	0	+	0	0	0
0	+	0	+	0	0	1	+	0	0	0
0	+	0	+	0	0	0	+	1	0	0
0	+	0	+	0	0	0	+	0	1	0
0	+	0	+	0	0	0	+	0	0	1

Minterm example, conclusion

(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

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

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

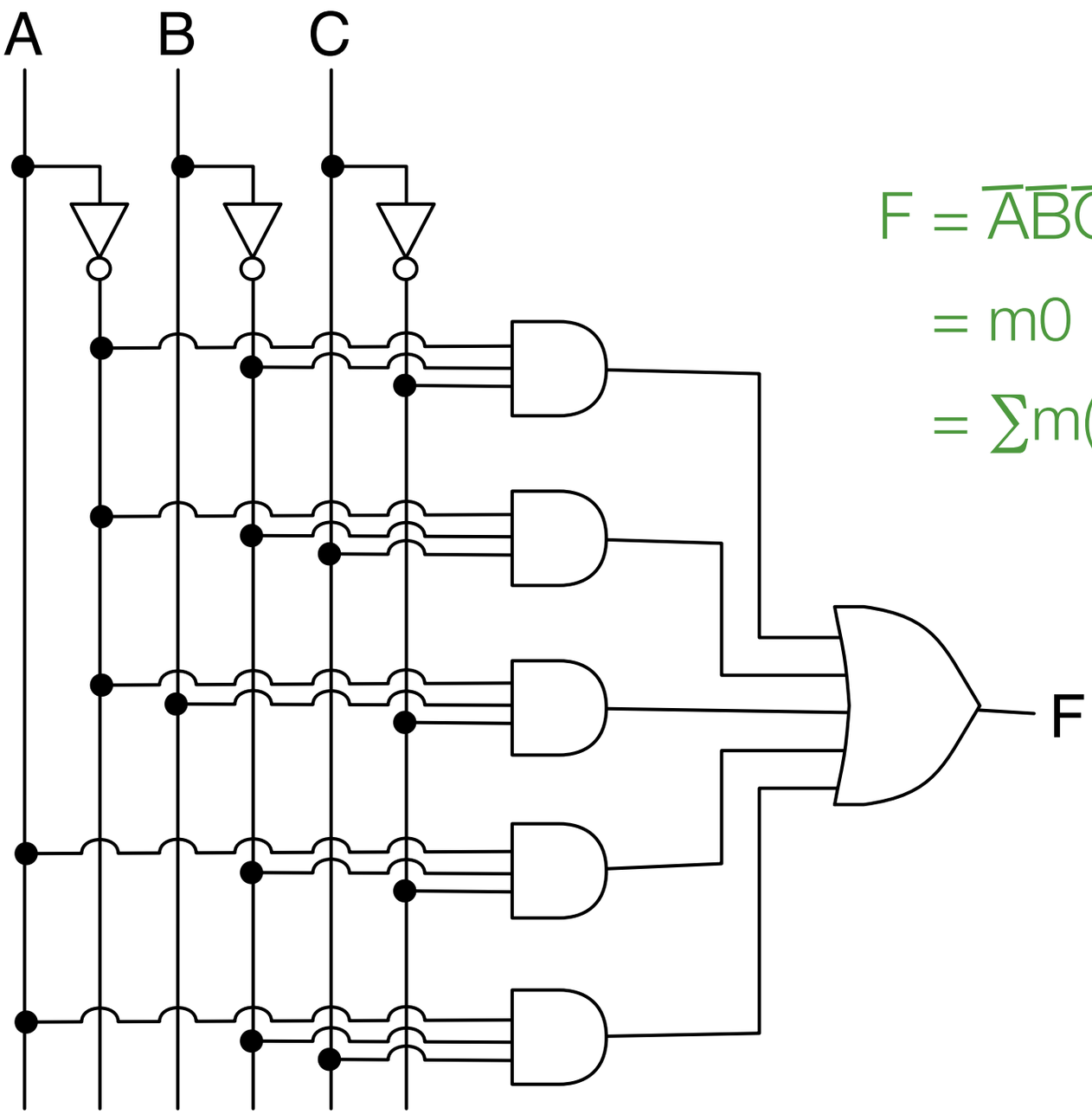
$$= \sum m(0,1,2,4,5)$$

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

$$= m3 + m6 + m7$$

$$= \sum m(3,6,7)$$

Minterms as a circuit



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

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

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

Standard form is not minimal form!

Simplest Standard Form v. Minimal Form

- Can be the same, but not always

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

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

Maxterms

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

- 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.

Maxterm description of a function

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

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

Maxterm example, seen another way

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

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

M0	M1	M2	M3	M4	M5	M6	M7
0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	0

Maxterm example, conclusion

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

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

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

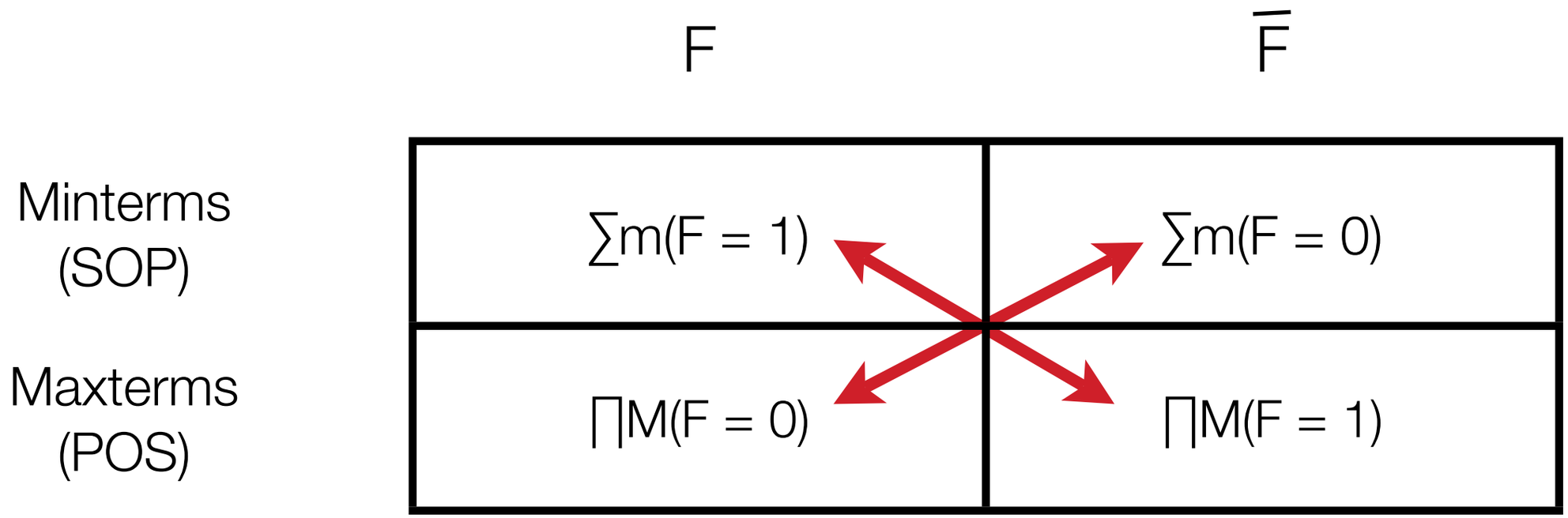
$$= (M3)\ (M6)\ (M7)$$

$$= \prod M(3,6,7)$$

Summary of Minterms and Maxterms

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

Converting between canonical forms



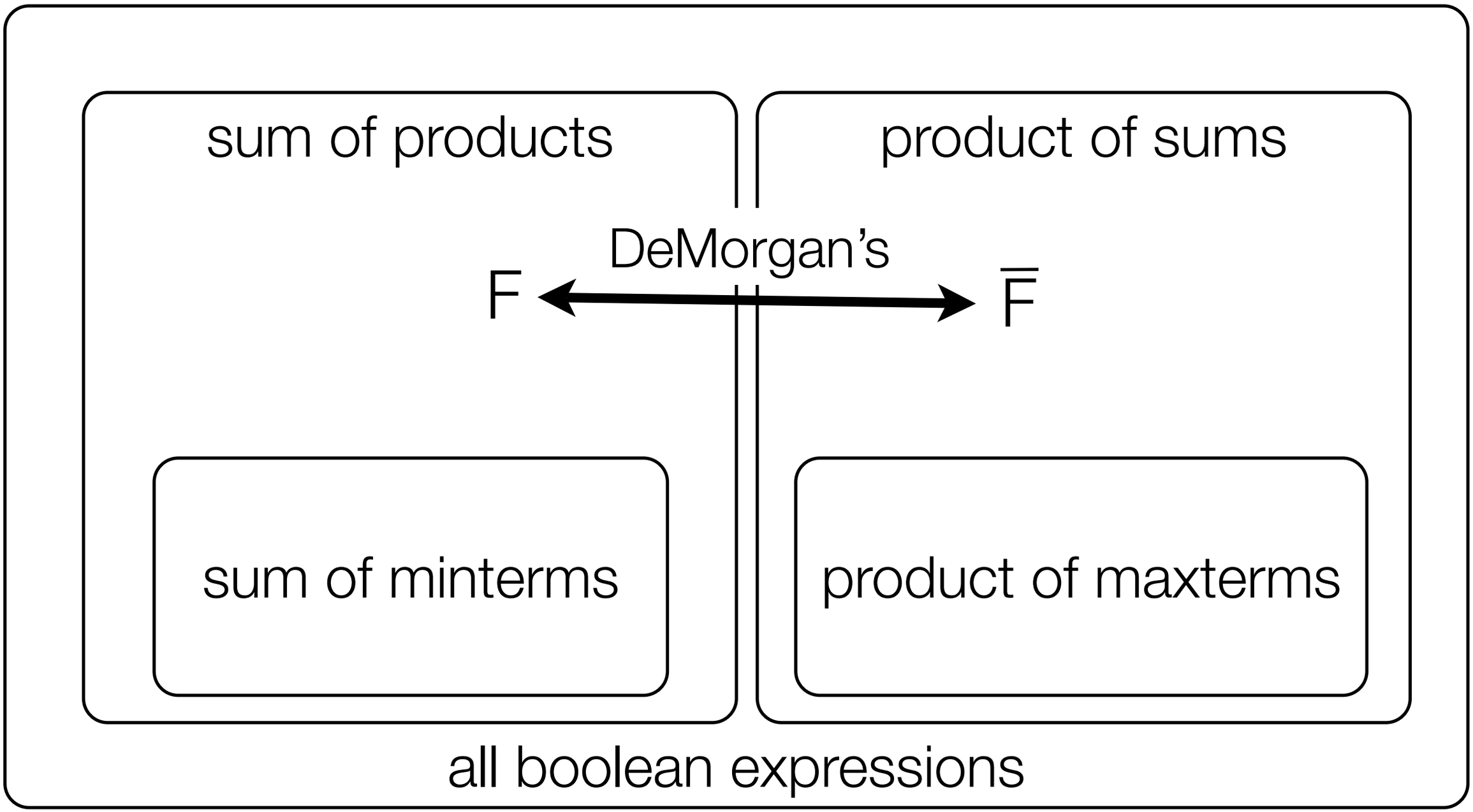
DeMorgans: same terms

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)		

Relations between standard forms



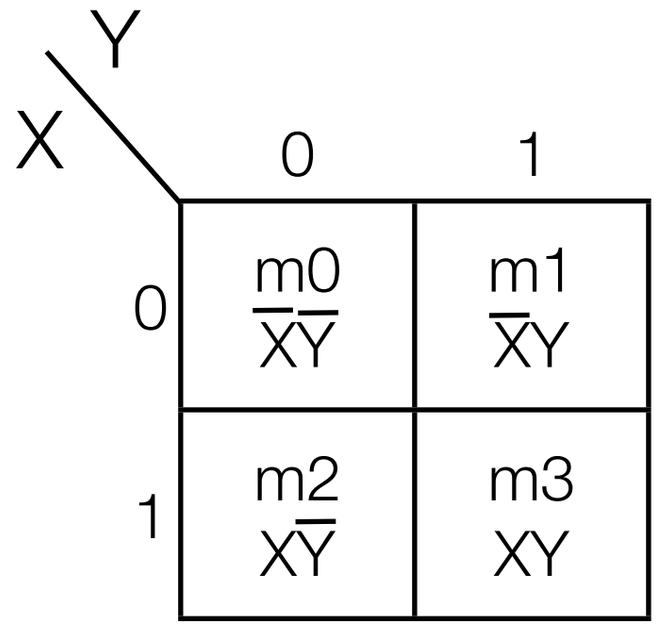
Circuit Simplification with Karnaugh Maps

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

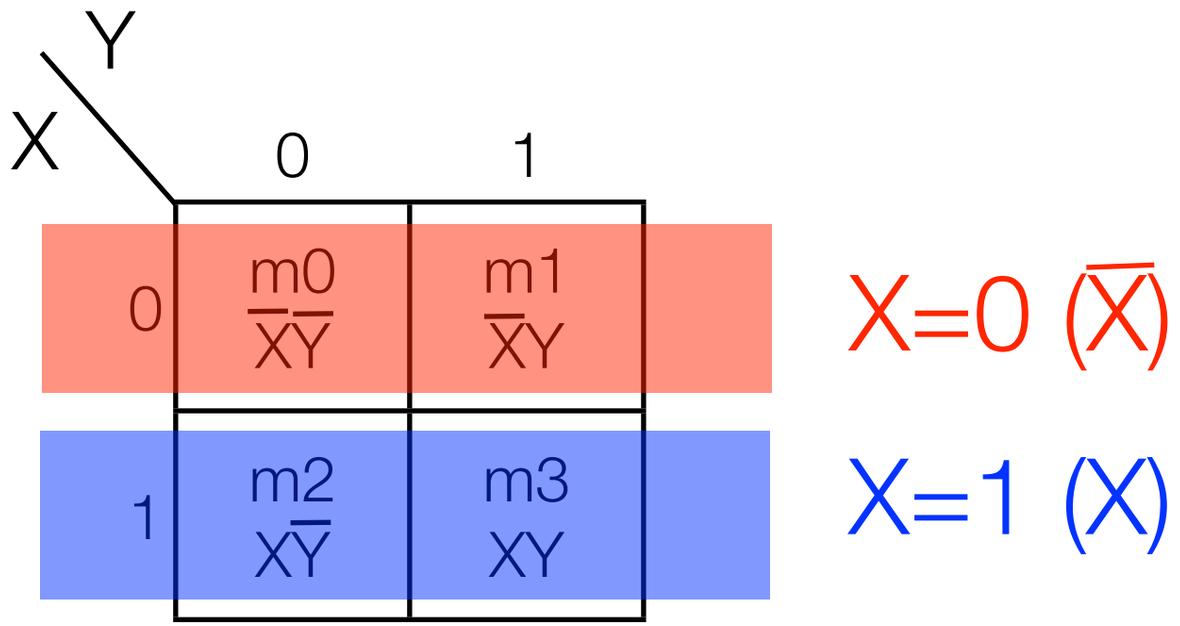


Karnaugh maps

- 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

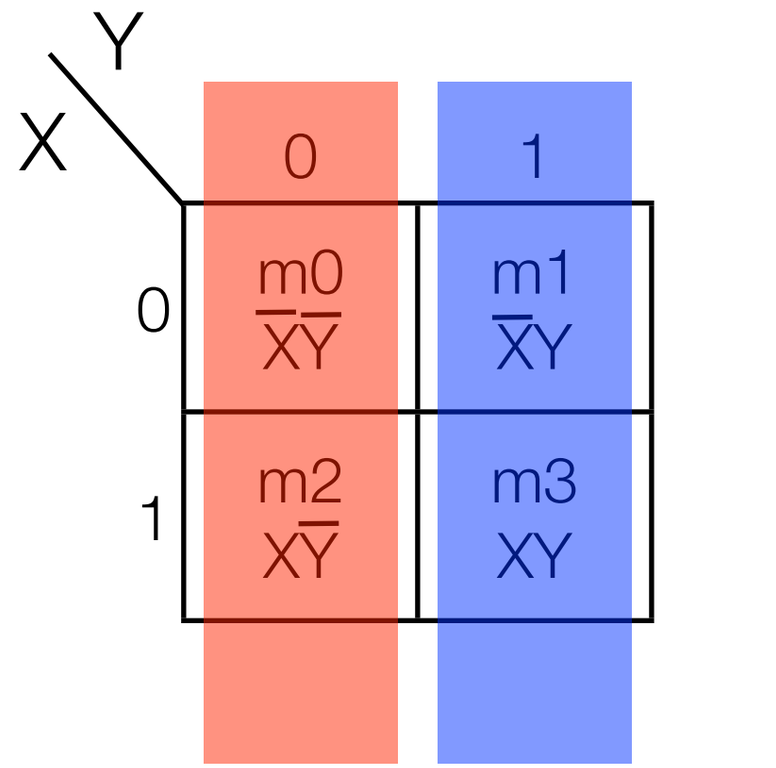


Karnaugh maps

- All functions can be expressed with a map

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

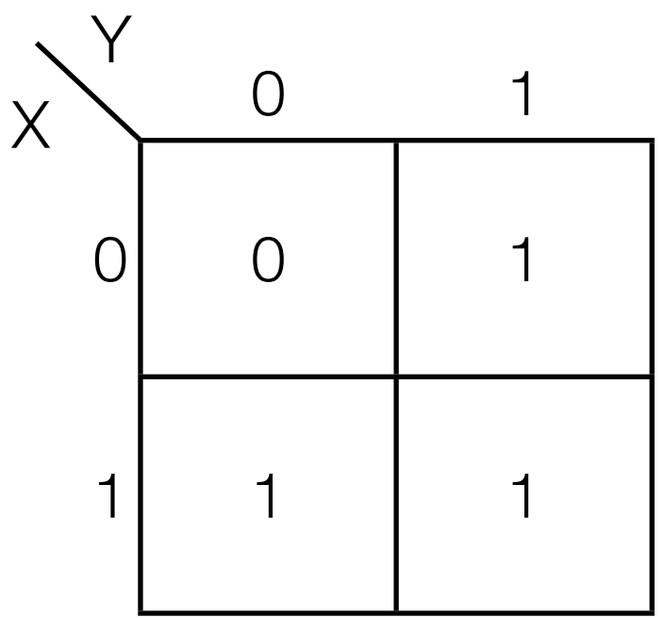
X	Y	F
0	0	m0
0	1	m1
1	0	m2
1	1	m3



Karnaugh maps express functions

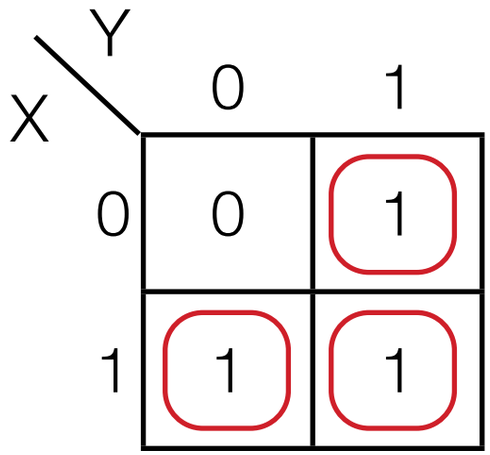
- Fill out table with value of a function

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

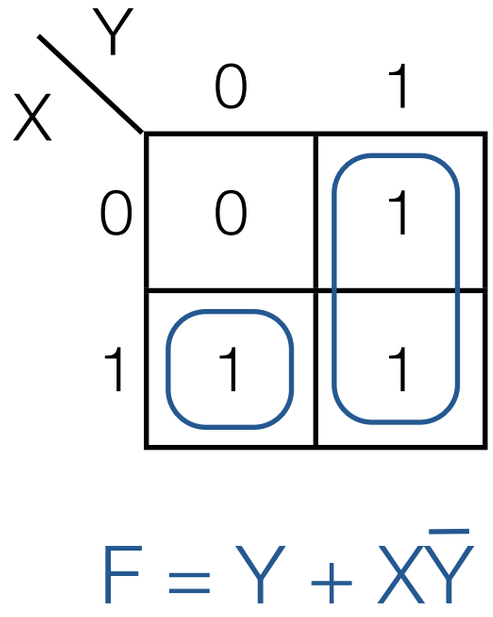


Simplification using a k-map

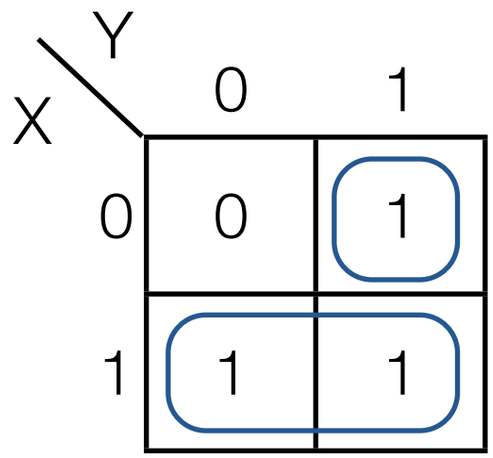
- Whenever two squares share an edge and both are 1, those two terms can be combined to form a single term with one less variable



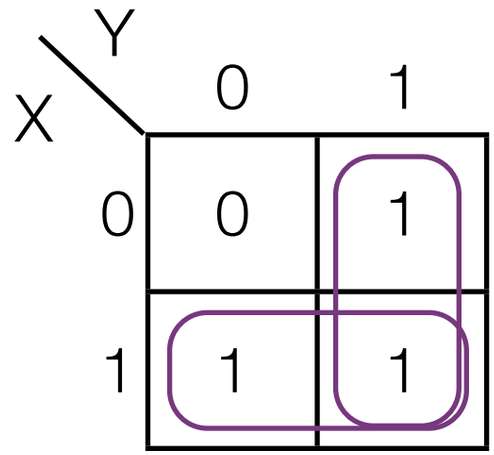
$F = \bar{X}Y + X\bar{Y} + XY$



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



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

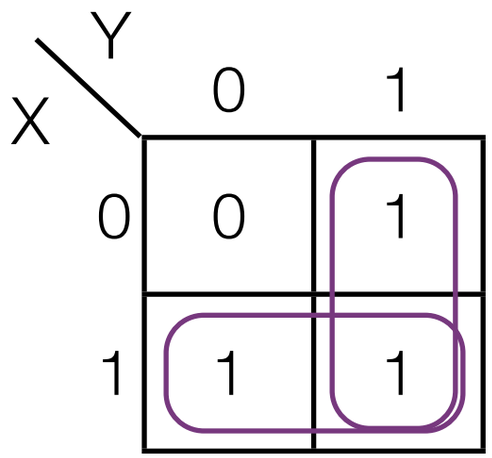


$F = X + Y$

Simplification using a k-map (2)

- Circle contiguous groups of 1s (circle sizes 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

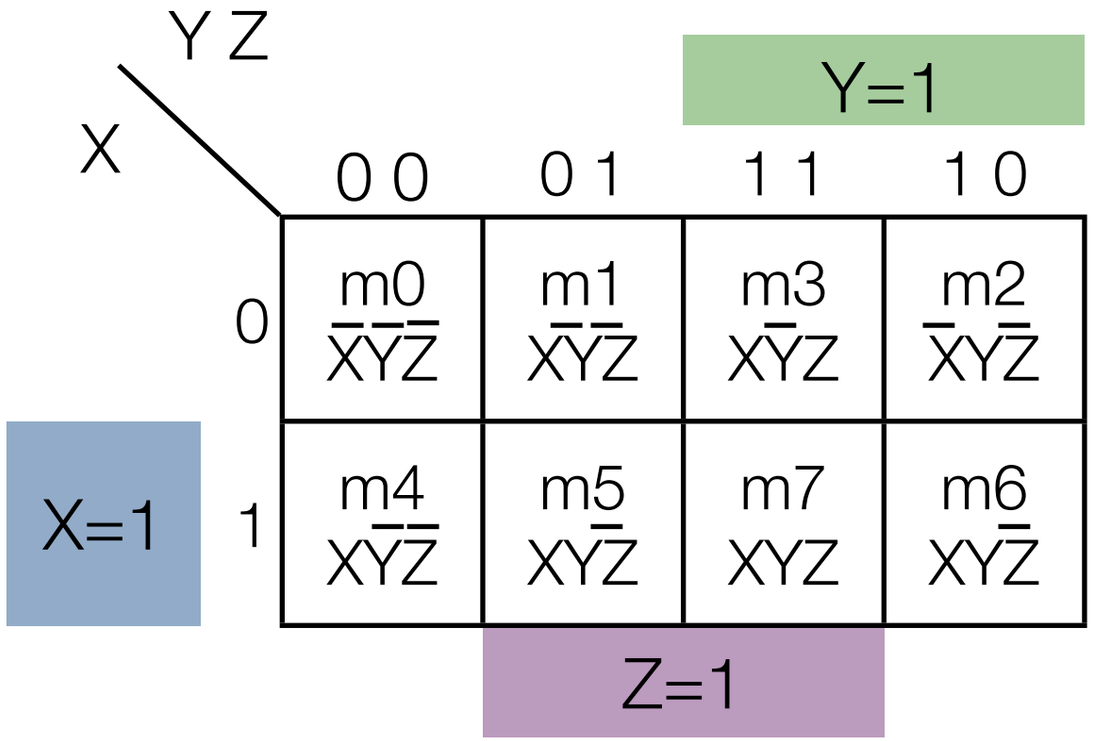
- Add circles (and terms) until all 1s on the k-map are circled



$F = X + Y$

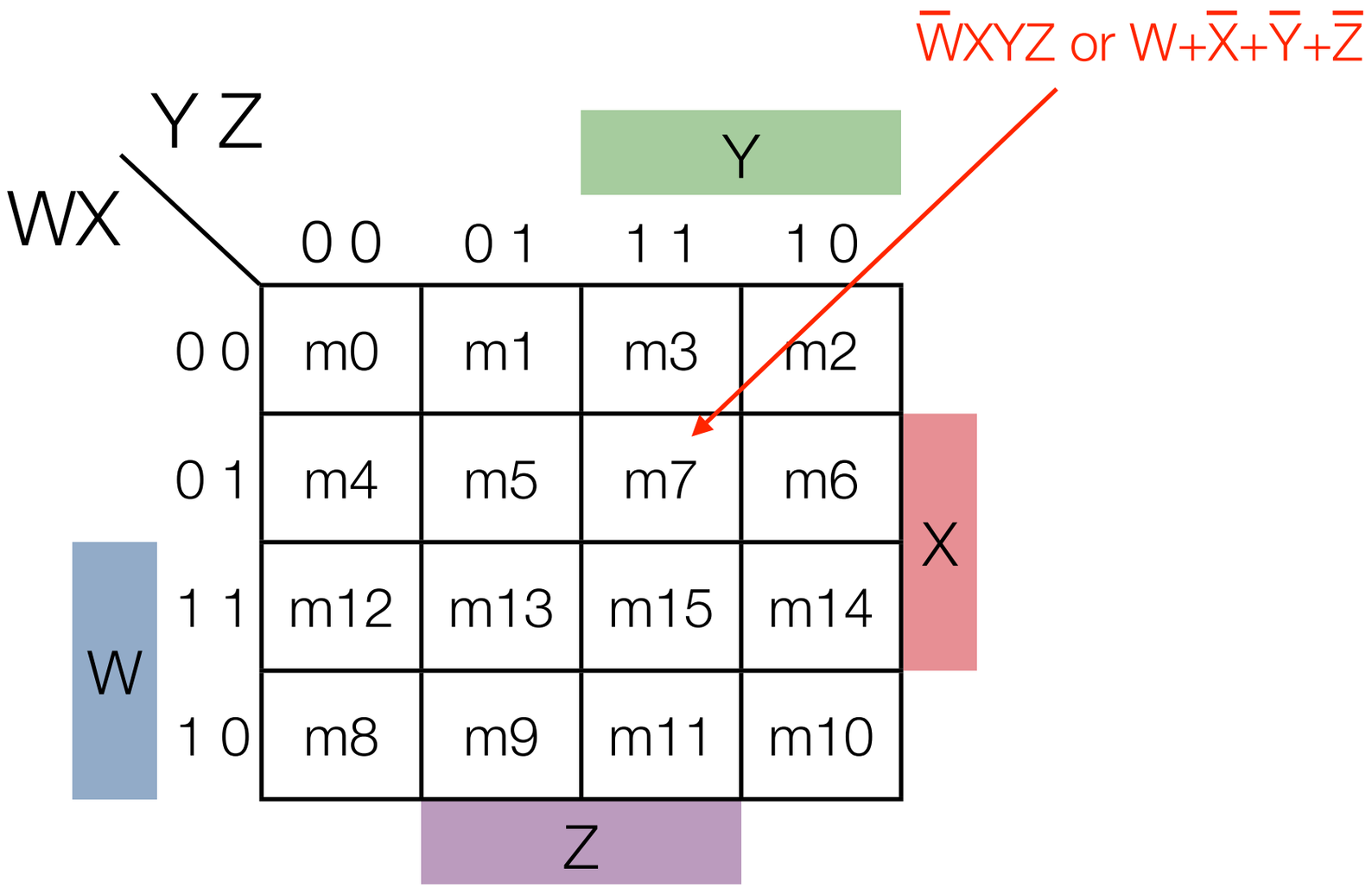
3-variable Karnaugh maps

- 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”)



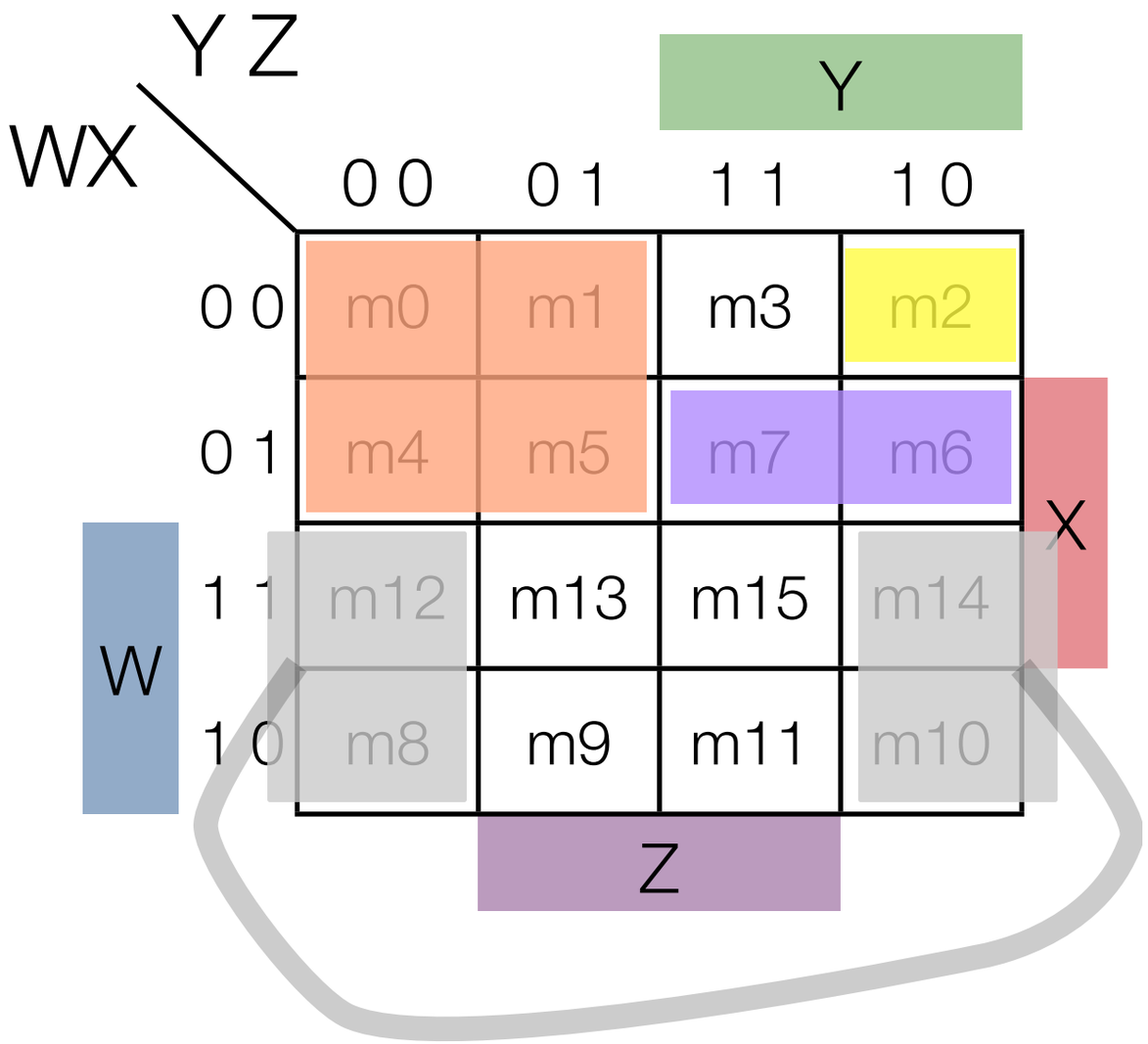
4-variable Karnaugh maps

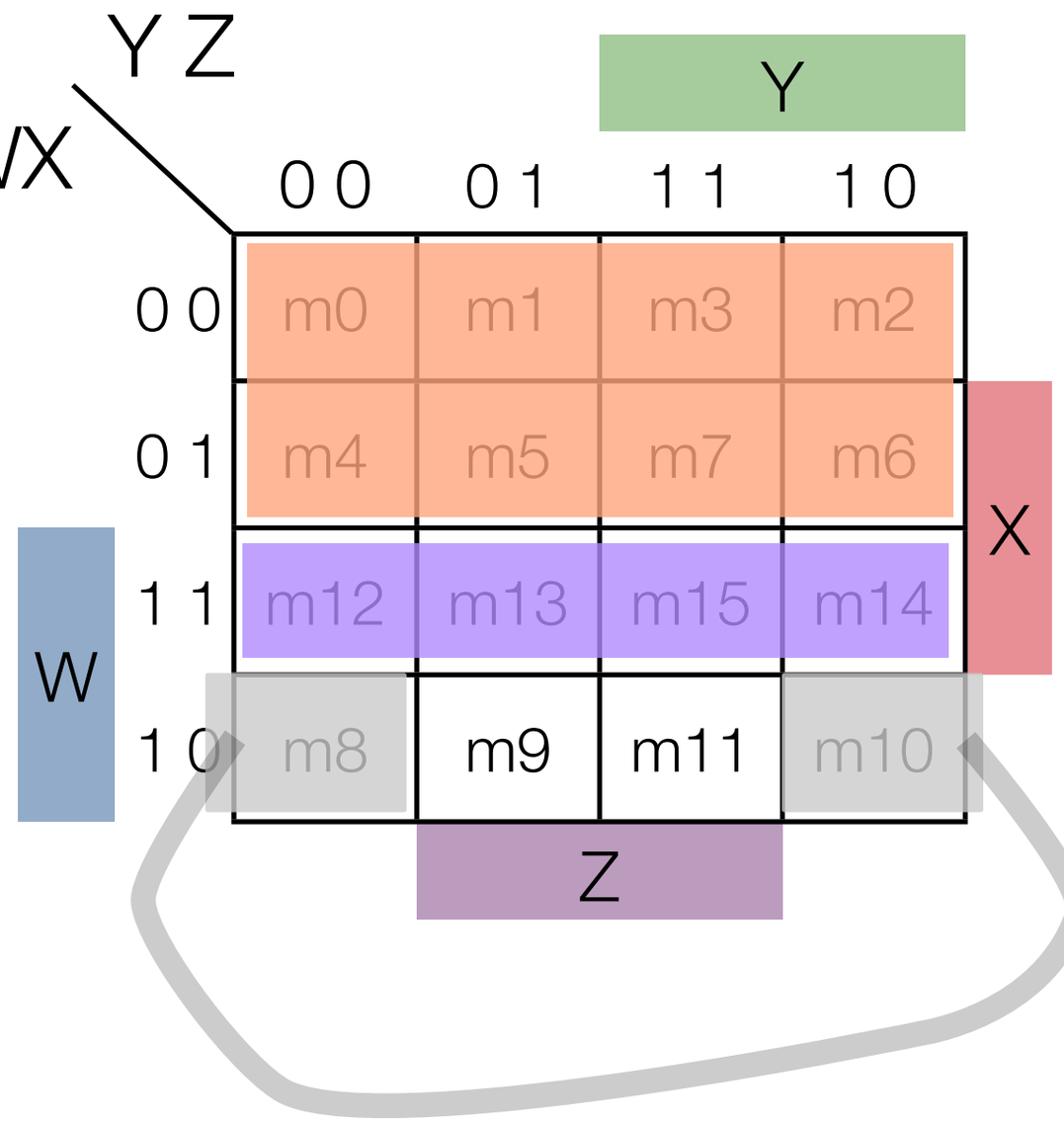
Extension of 3-variable maps



Implicants

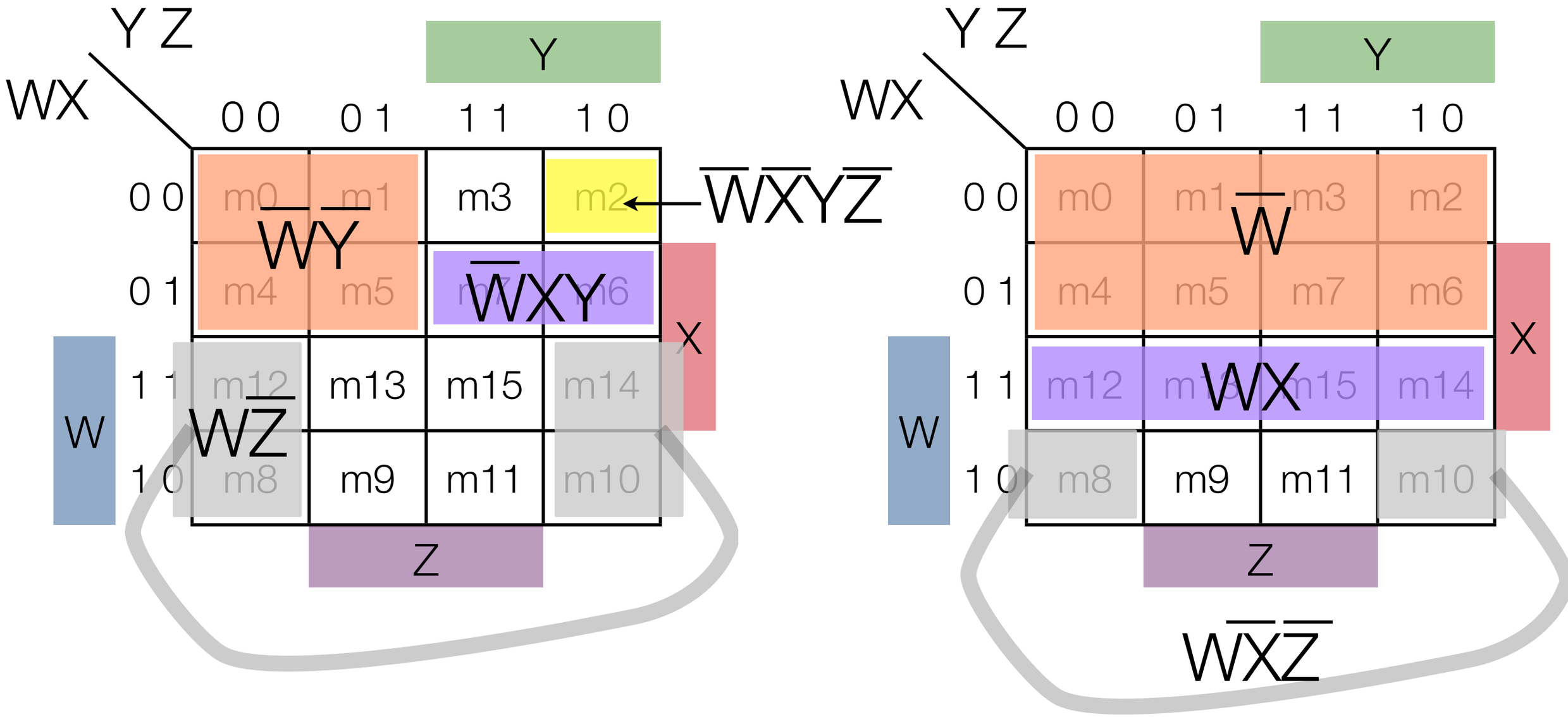
- **Implicant**: a product term, which, viewed in a K-Map is a 2^i x 2^j size "rectangle" (possibly wrapping around) where i=0,1,2, j=0,1,2





Implicants

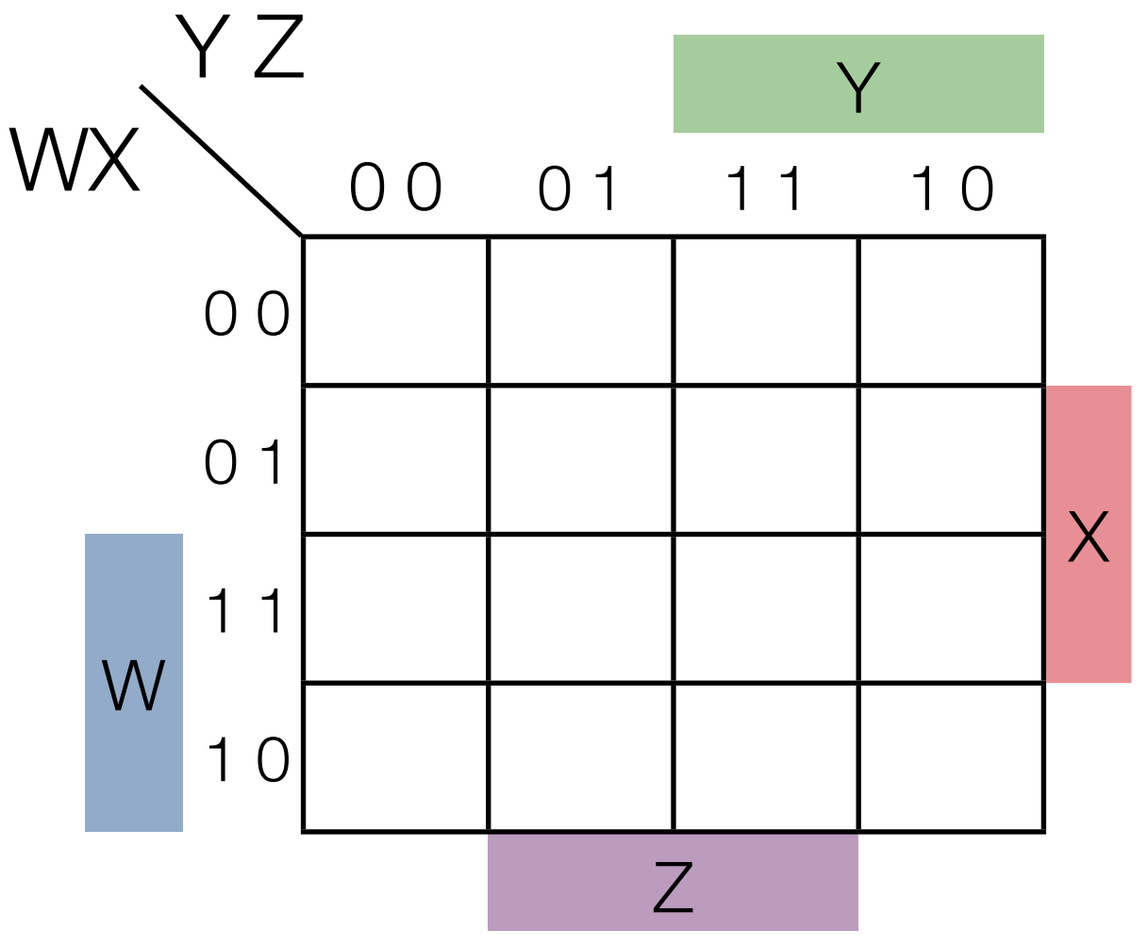
- **Implicant**: a product term, which, viewed in a K-Map is a 2^i x 2^j size "rectangle" (possibly wrapping around) where i=0,1,2, j=0,1,2



Note: bigger rectangles = fewer literals

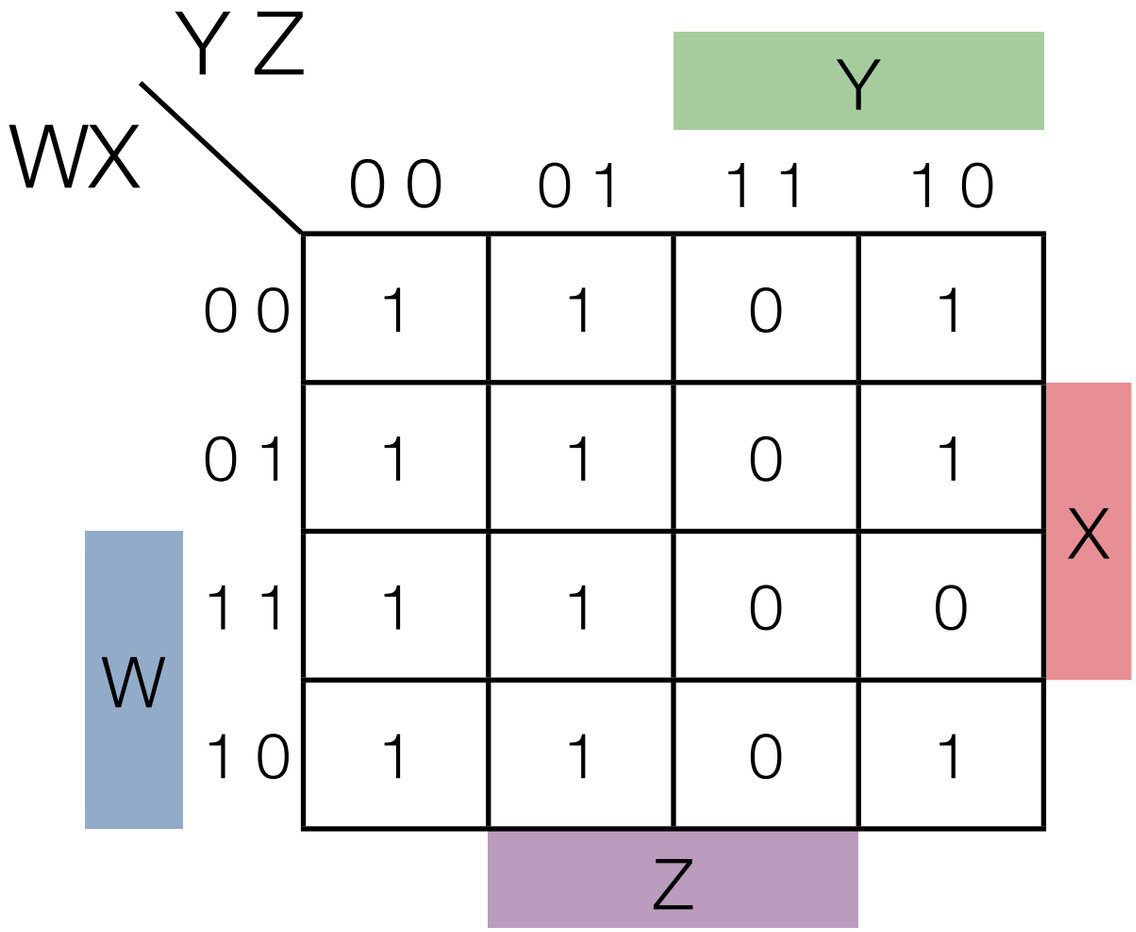
4-variable Karnaugh map example

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



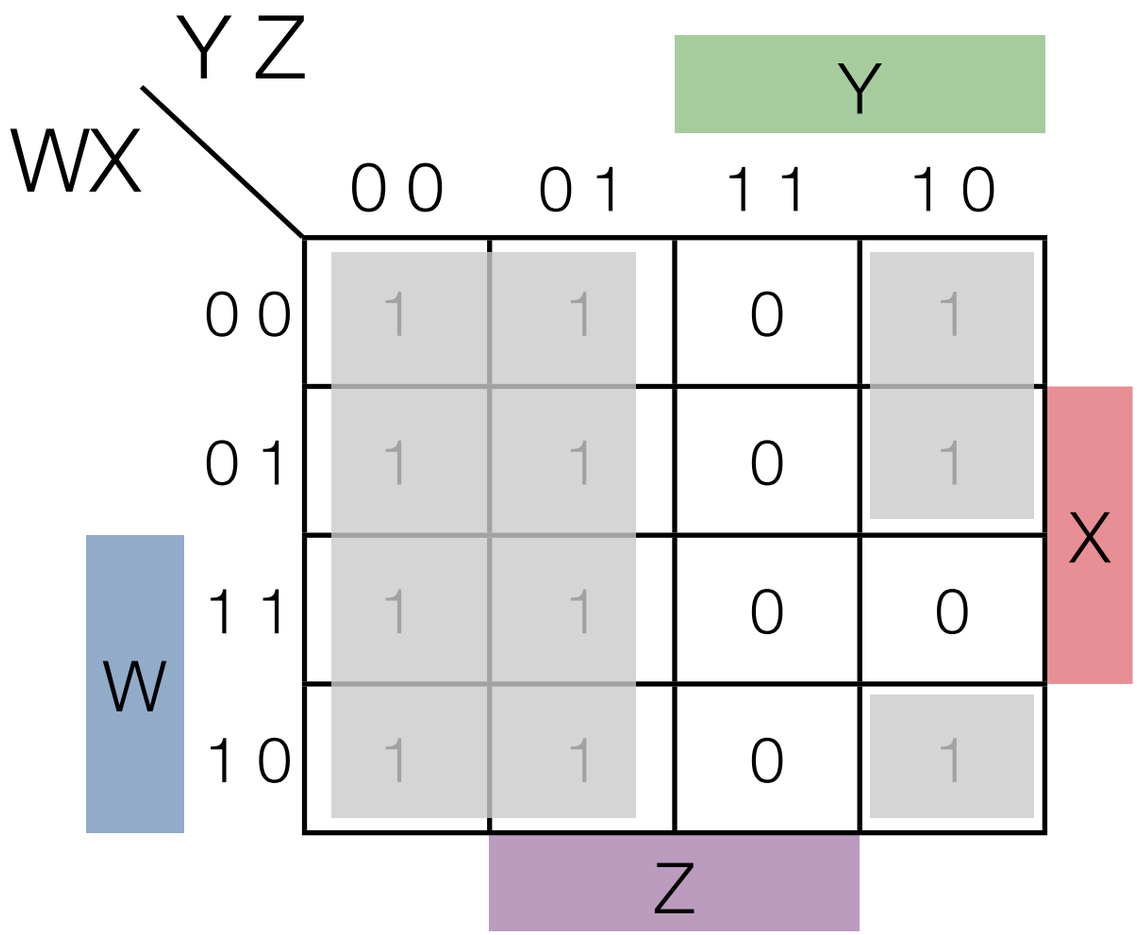
4-variable Karnaugh map example

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



4-variable Karnaugh map 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



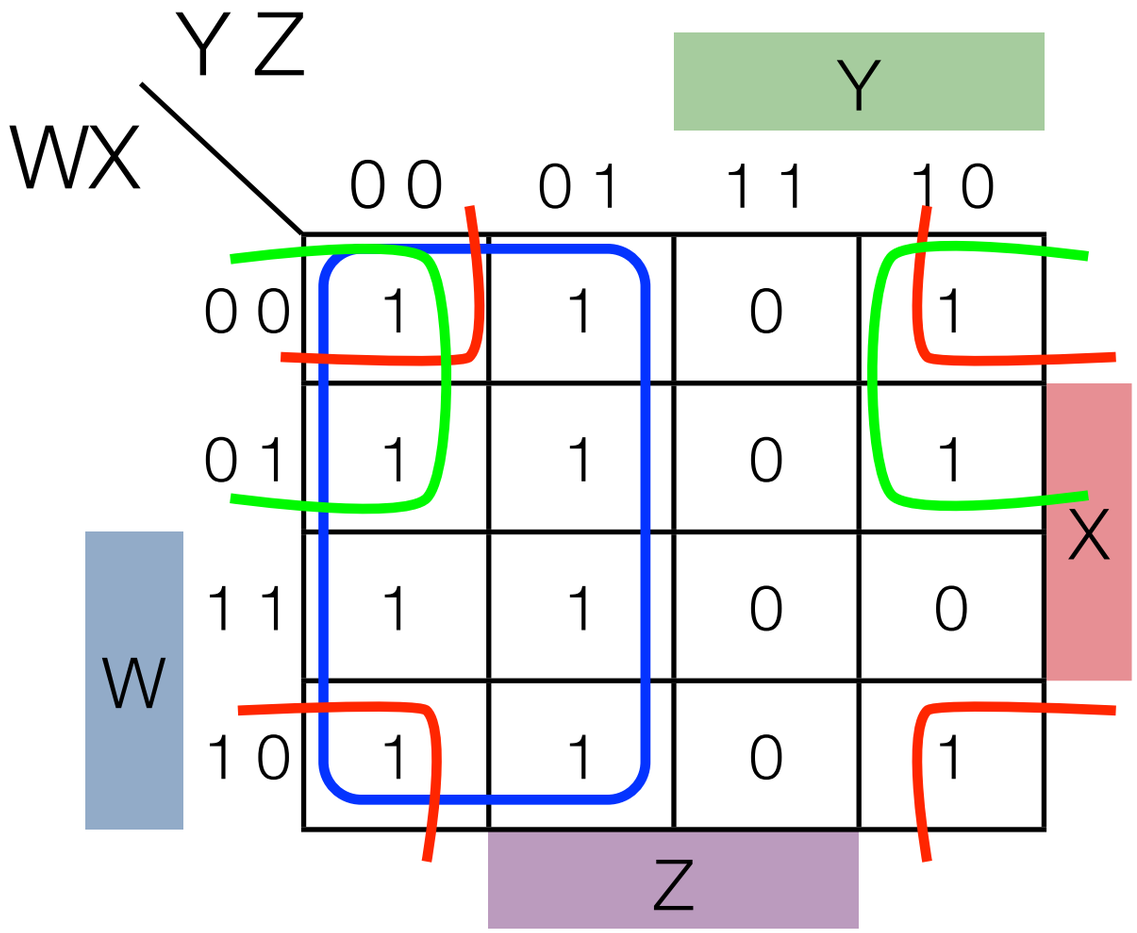
K-maps make F expressed as SoP easy to see, e.g.,

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

Can the expression for F be simplified further?

4-variable Karnaugh map 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



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

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

More implicant terminology

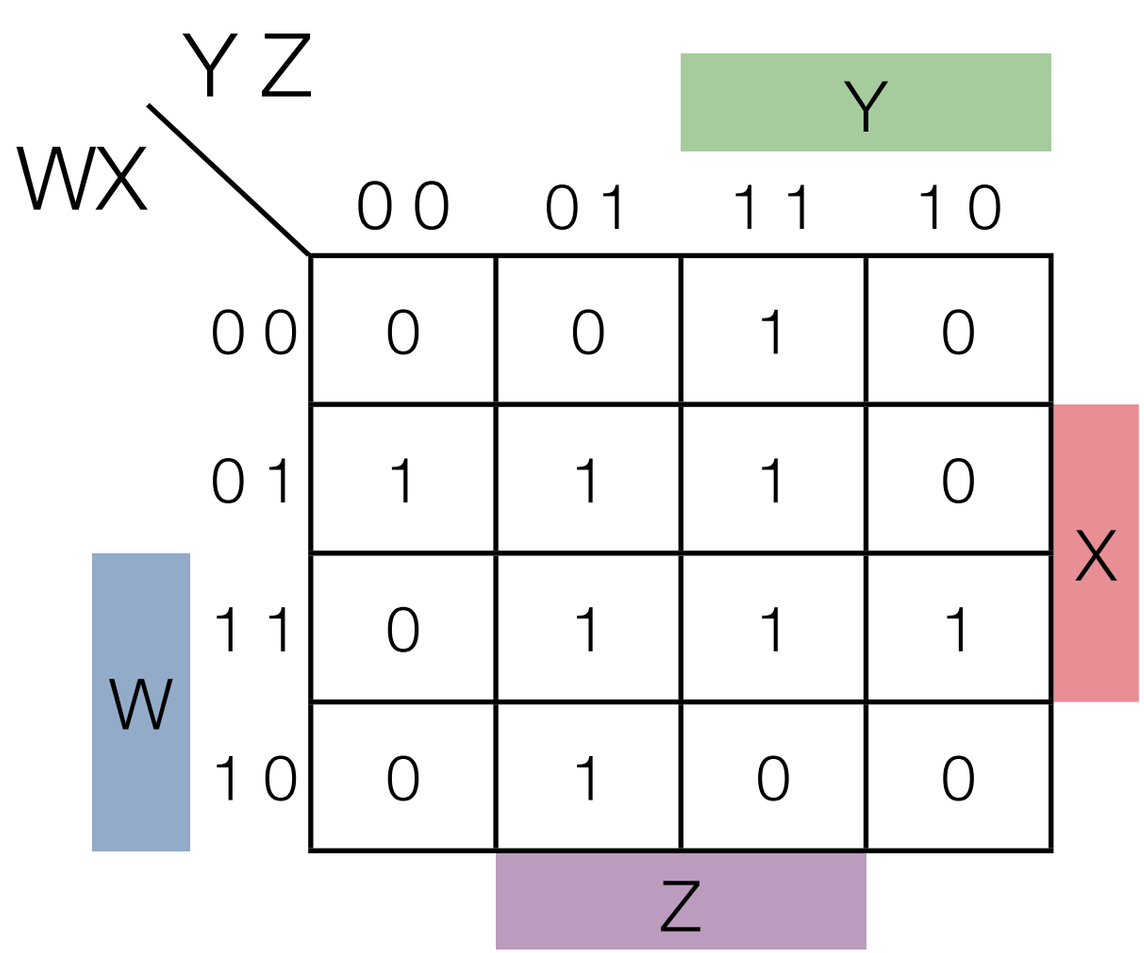
- **implicant**: a product term, which, viewed in a K-Map is a 2^i x 2^j size "rectangle" (possibly wrapping around) where i=0,1,2, j=0,1,2

- **prime implicant**: An implicant not contained within another implicant.

- **essential prime implicant**: a **prime implicant** that is the **only prime implicant** to cover some minterm.

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?



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

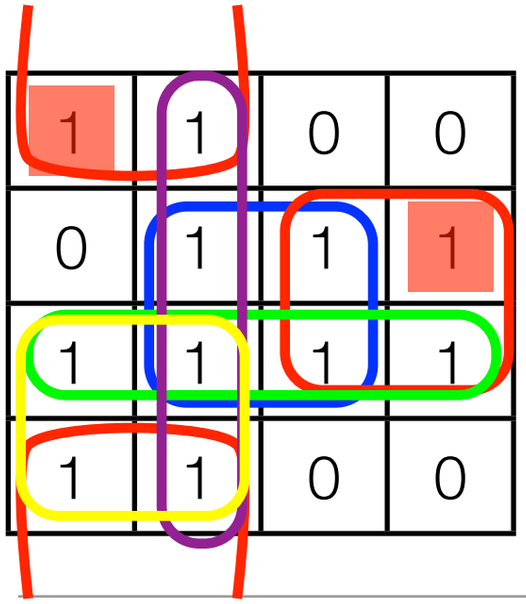
1	1	1	0
0	1	1	0
1	1	1	1
1	1	0	1

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

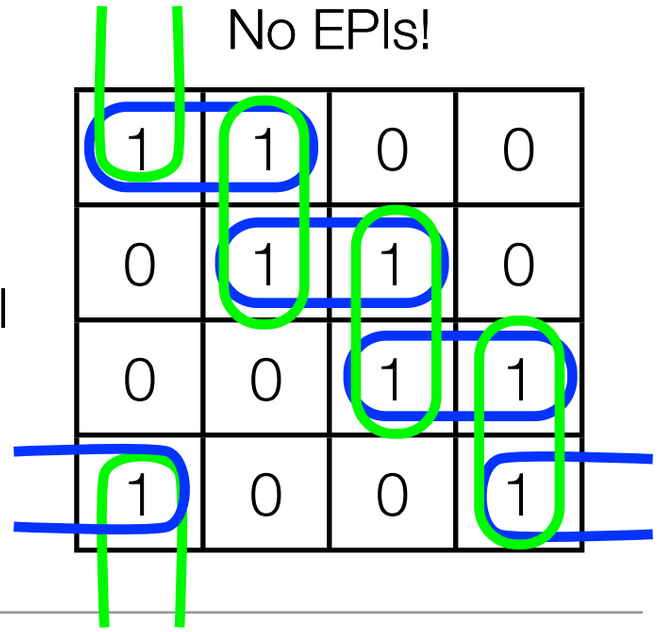
Red bounds are EPIs (solo-covered minterm shown in red)



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				

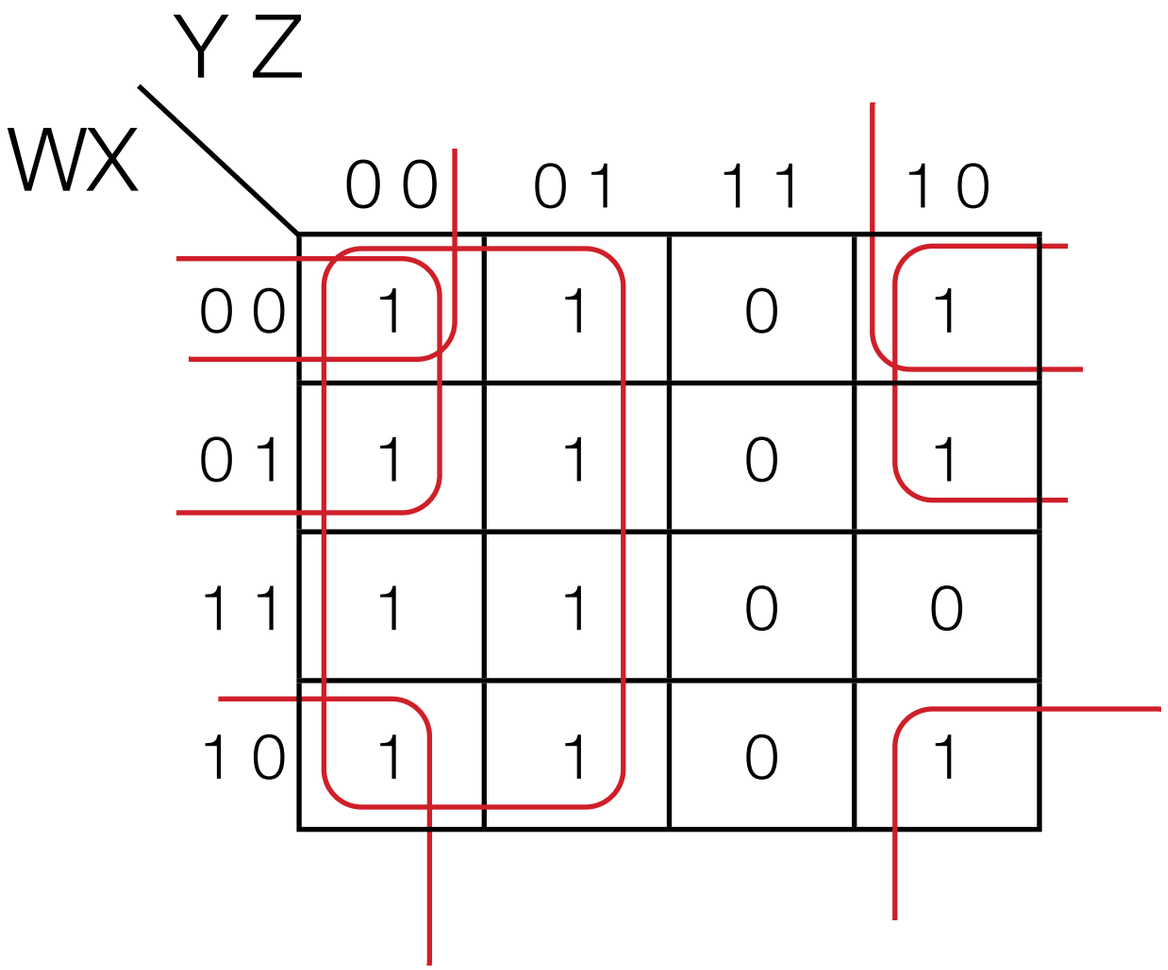
two 2-bit #'s multiplied together to give a 4-bit solution

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

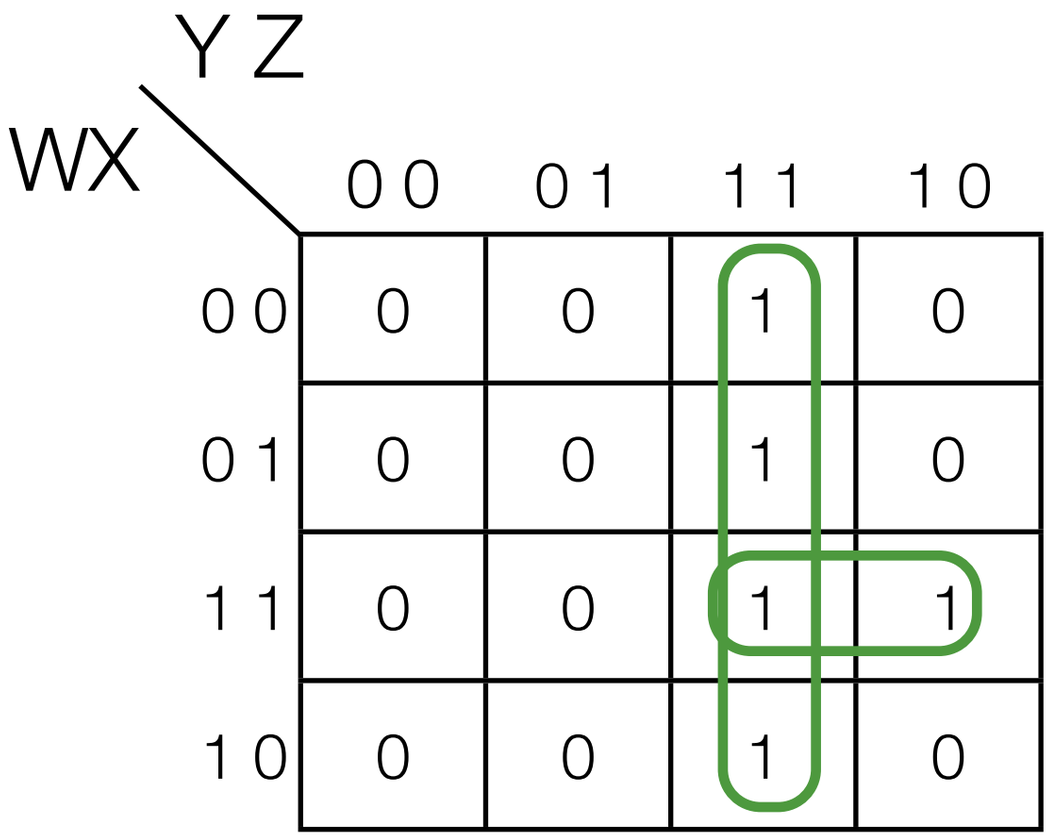
K-Maps: Complements, PoS, don’t care conditions

Finding $\overline{F}$

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



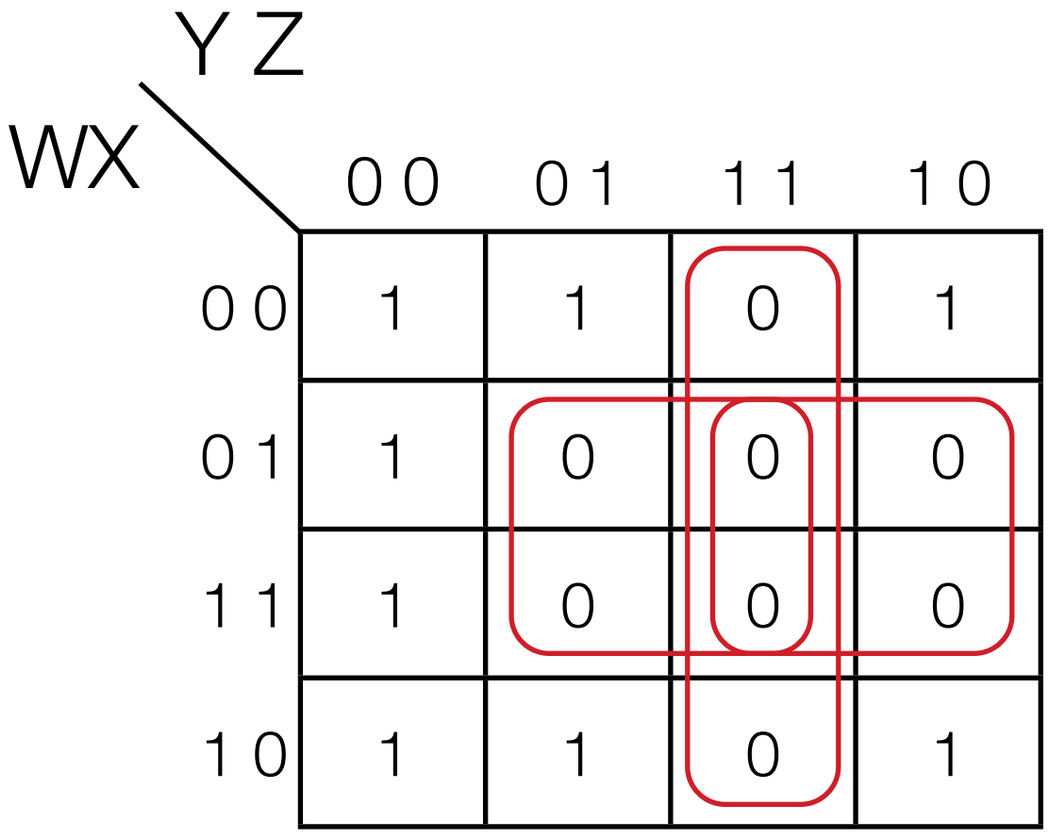
$$F = \overline{Y} + \overline{X}\overline{Z} + \overline{W}\overline{Z}$$



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

PoS expressions from a k-map

Find $\overline{F}$ as SoP and then apply DeMorgan's



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

DeMorgan's

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

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 if we are told that a and b will be non-0? We “don’t care” what the output looks like for the input cases that should 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:

- During minimization can be treated as either a 1 or a 0

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

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 that a disallowed input combo is A=1, B=0
- 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

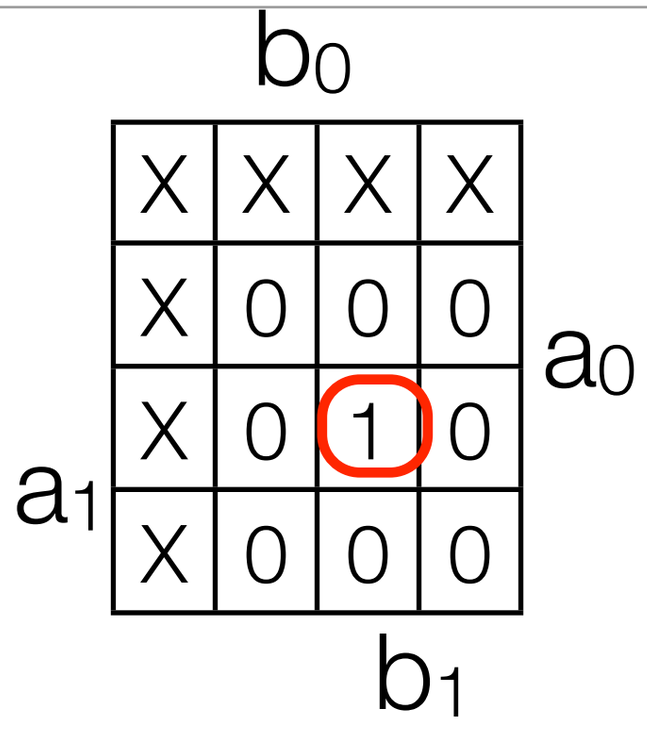
Inputs will not occur

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

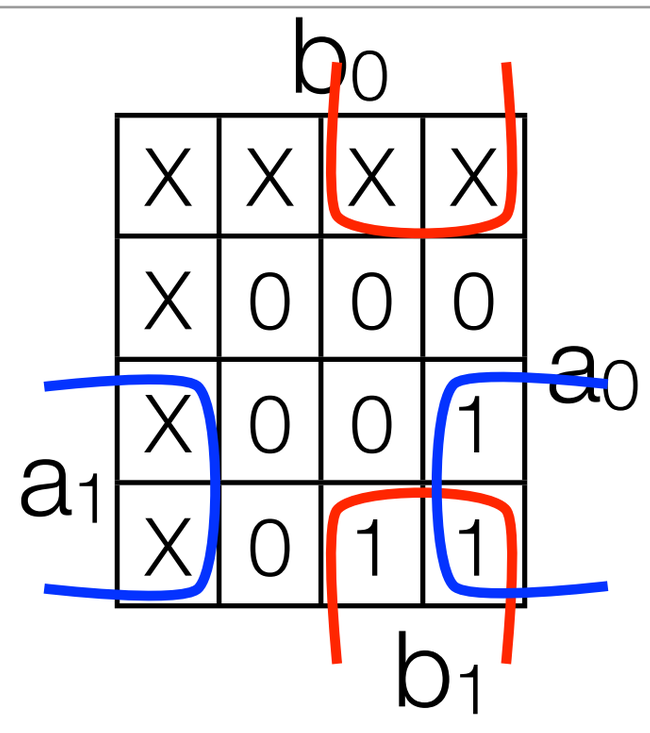
- Both F & G are appropriate functions for H
- G can substitute for F for valid input combinations

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



$z_3 = a_1 a_0 b_1 b_0$



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

(vs. $a_1\overline{a_0}b_1 + a_1b_1\overline{b_0}$)

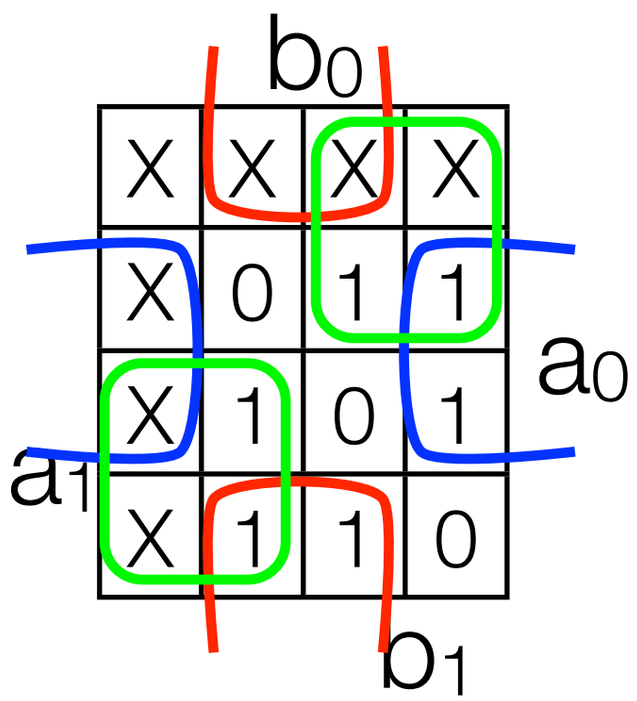
1's must be covered
0's must not be covered
X's are optionally covered

2-bit multiplier non-0 multiplier (SOLUTION)

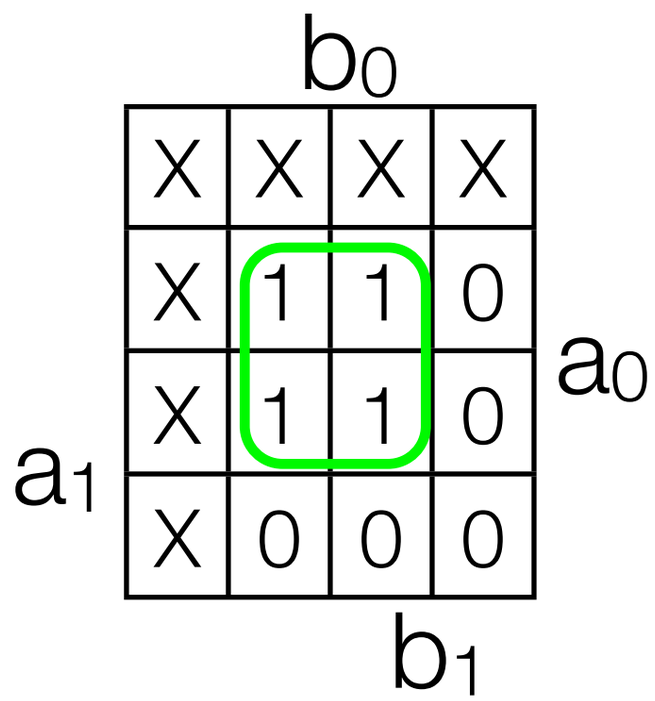
a1	a0	b1	b0	z3	z2	z1	z0
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	X	X	X	X
0	1	0	0	X	X	X	X
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	X	X	X	X
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	X	X	X	X
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Still have prime and essential prime implicants

z_1 = (exercise)



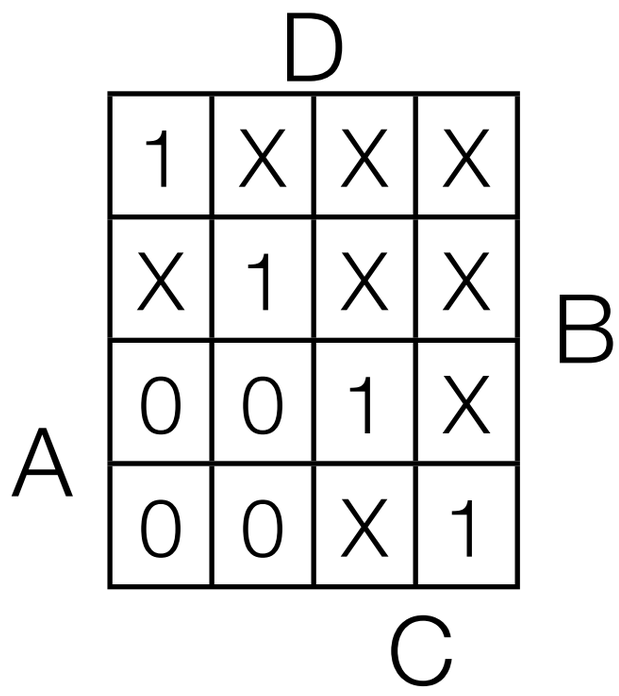
$z_0 = a_0 b_0$



All above prime implicants are essential

Final thoughts on Don’t care conditions

Sometimes “don’t cares” greatly simplify circuitry



$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + ABCD + A\bar{B}C\bar{D}$ vs. $\bar{A} + C$

Timing, Glitches, and Hazards

Timing

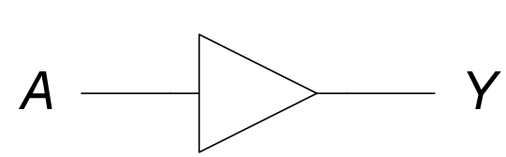
There is a delay between changes in circuit input and the output changing in response

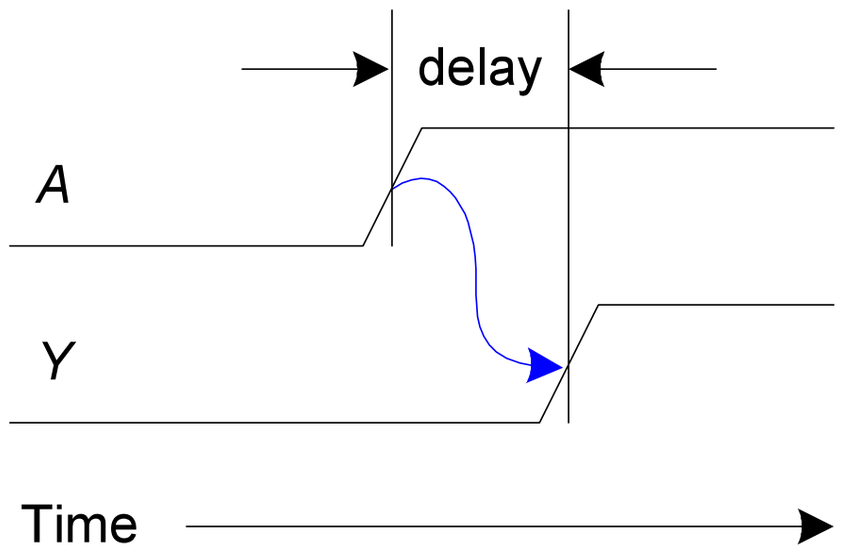
The challenge is to build fast circuits

Delay is caused by

Capacitance and resistance in a circuit

Speed of light limitation

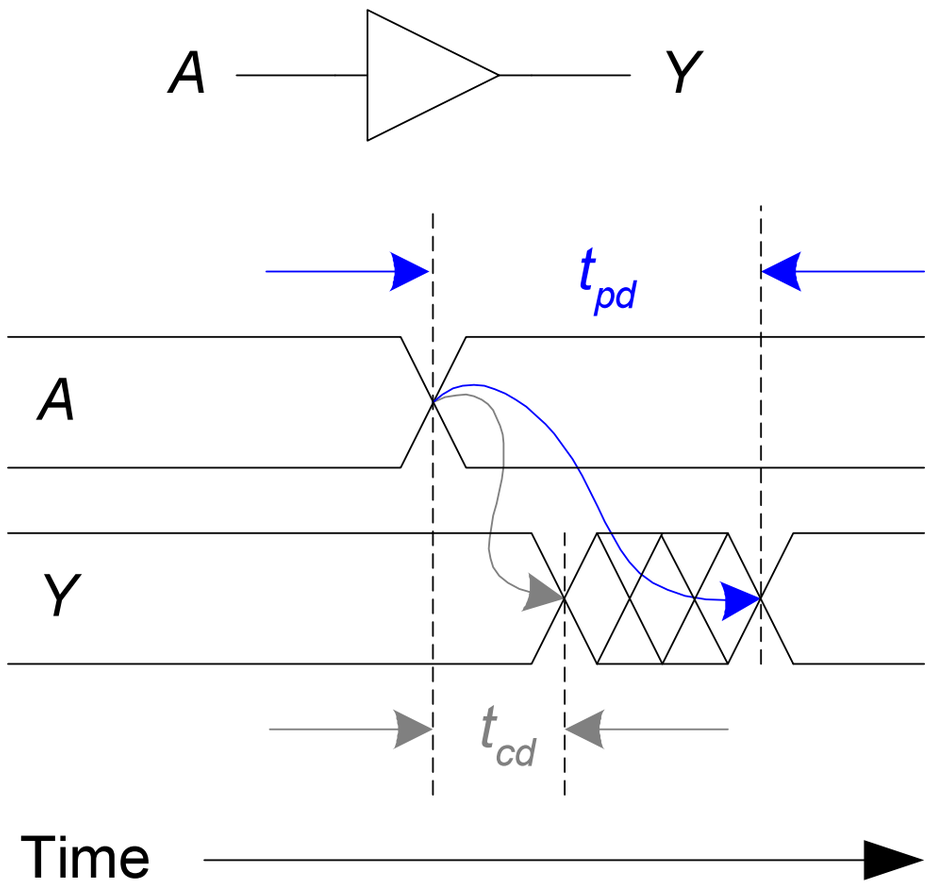




Propagation and Contamination Delay

Propagation delay: t_{pd} = max delay from input to output

Contamination delay: t_{cd} = min delay from input to output



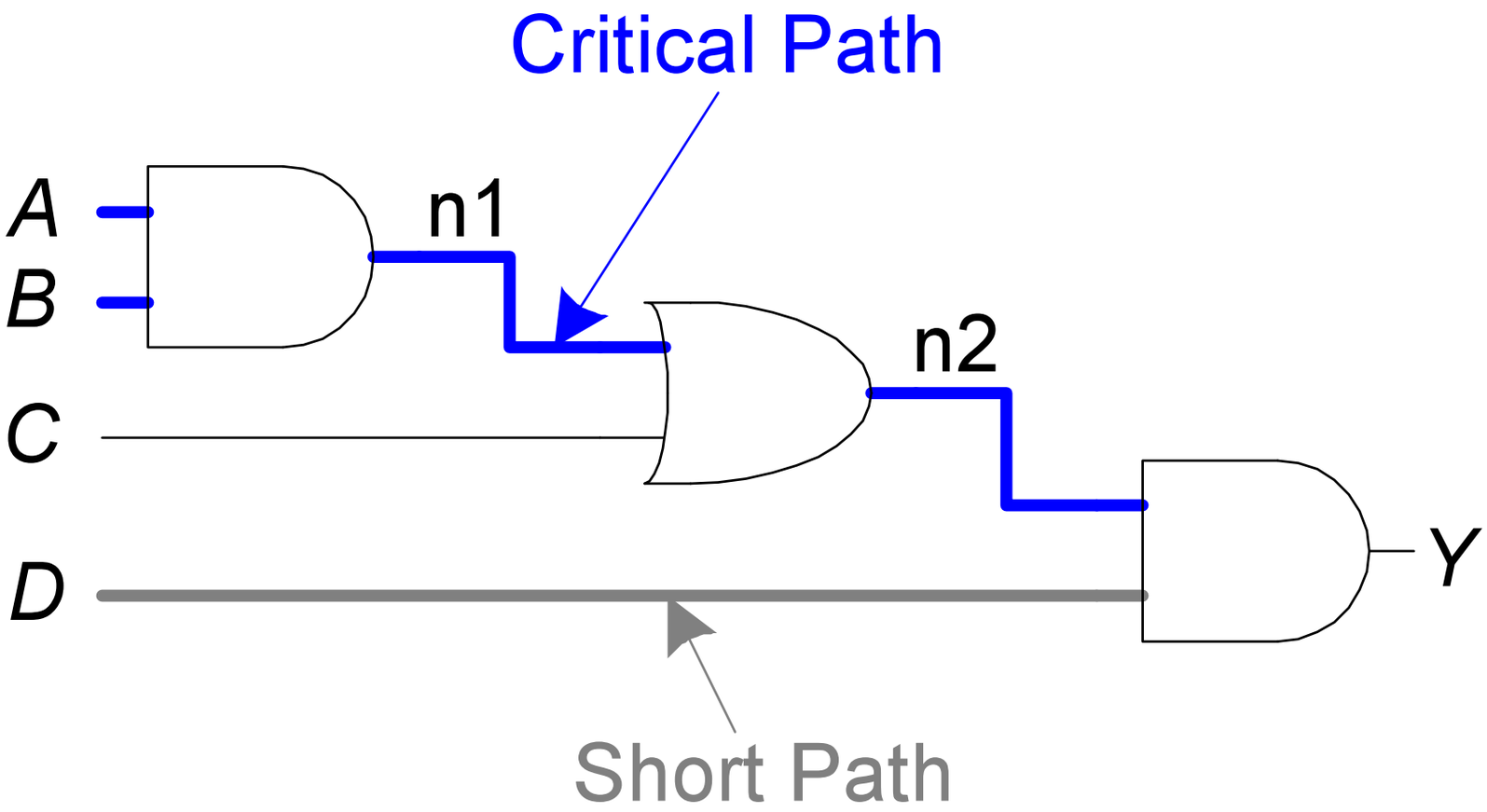
Reasons why t_{pd} and t_{cd} may be different:

Different rising and falling delays

Multiple inputs and outputs, some of which are faster than others

Circuits slow down when hot and speed up when cold

Critical (Long) and Short Paths

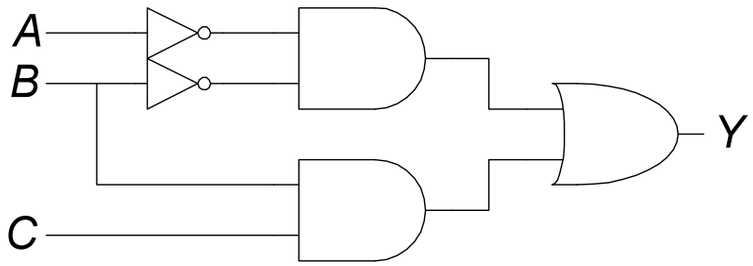


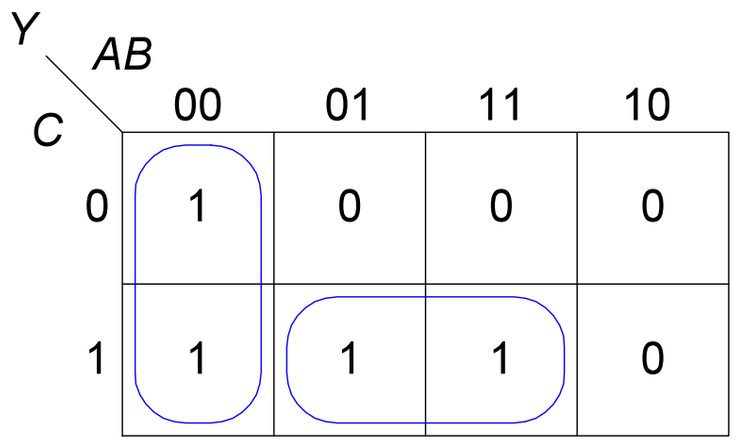
Critical (Long) Path: $t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$

Short Path: $t_{cd} = t_{cd\_\mathrm{AND}}$

Glitches

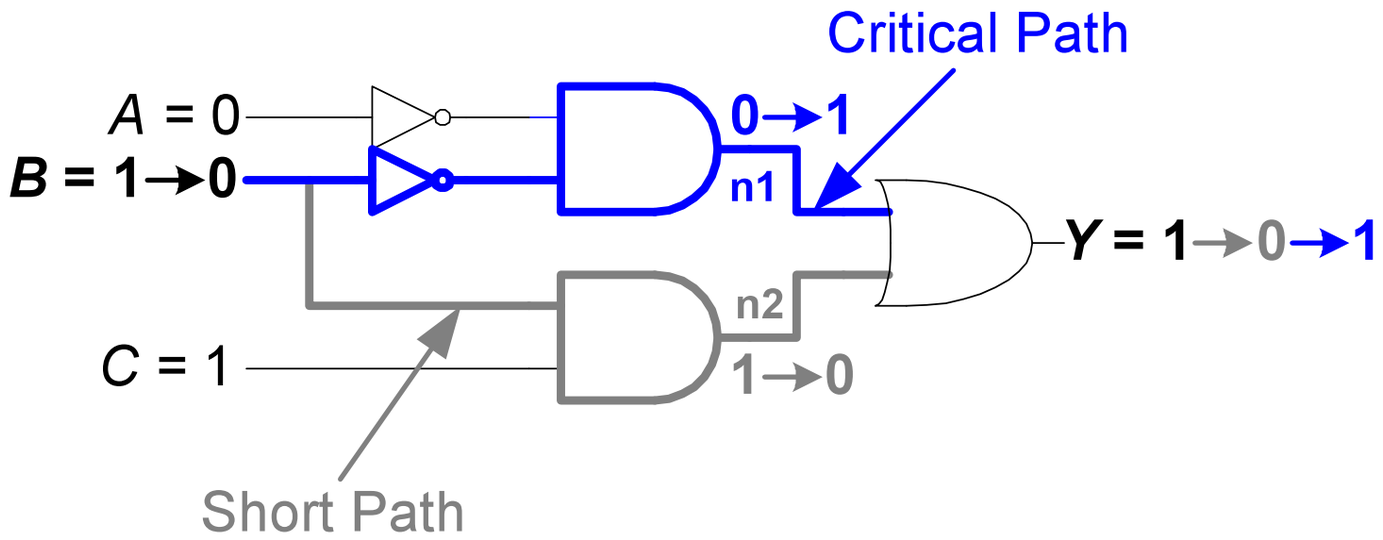
- Glitch: when a single input change causes multiple output changes
- Glitches don't cause problems because of synchronous design conventions (which we'll talk about in a bit)
- But it's important to recognize a glitch when you see one in timing diagrams
- Example: *what happens when A=0, C=1, and B falls?*

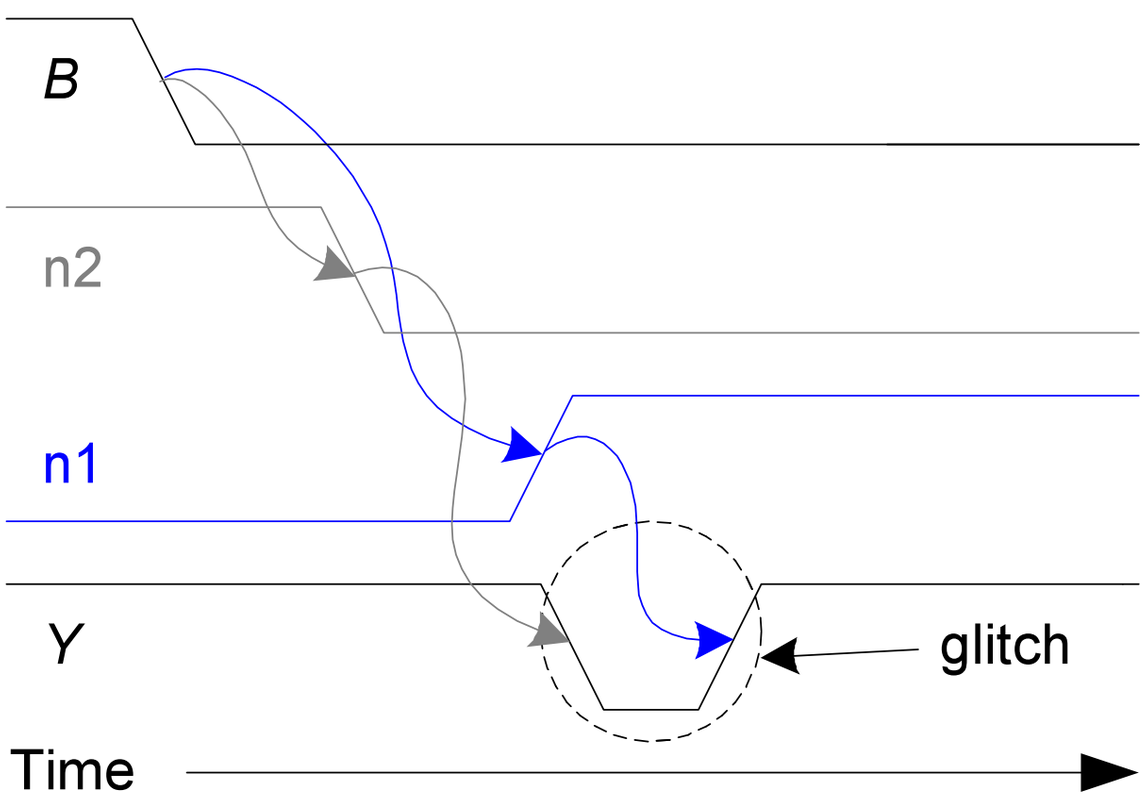




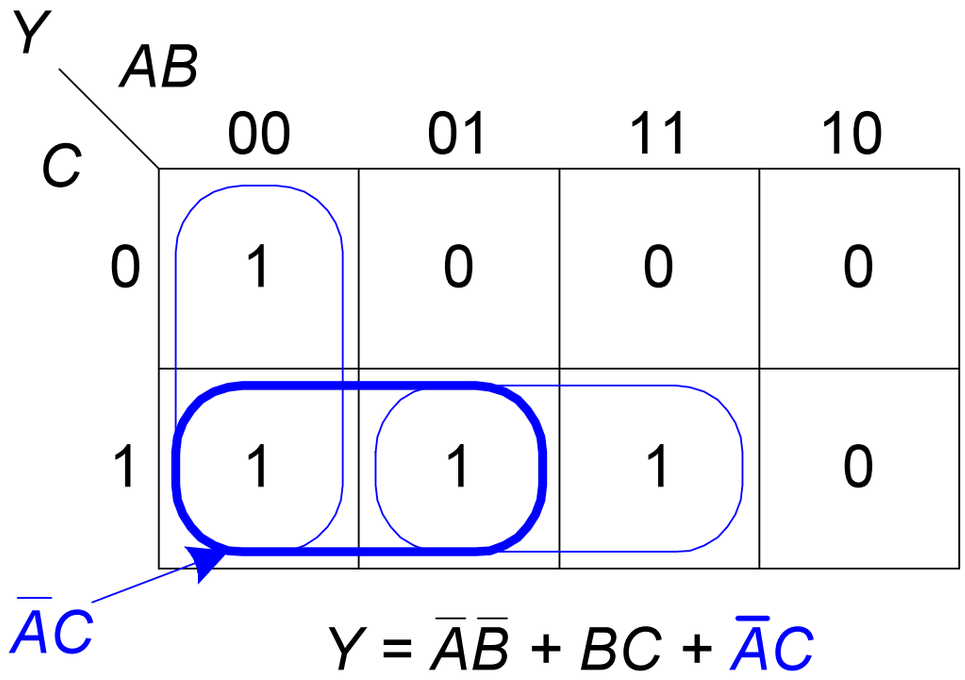
$Y = \bar{A}\bar{B} + BC$

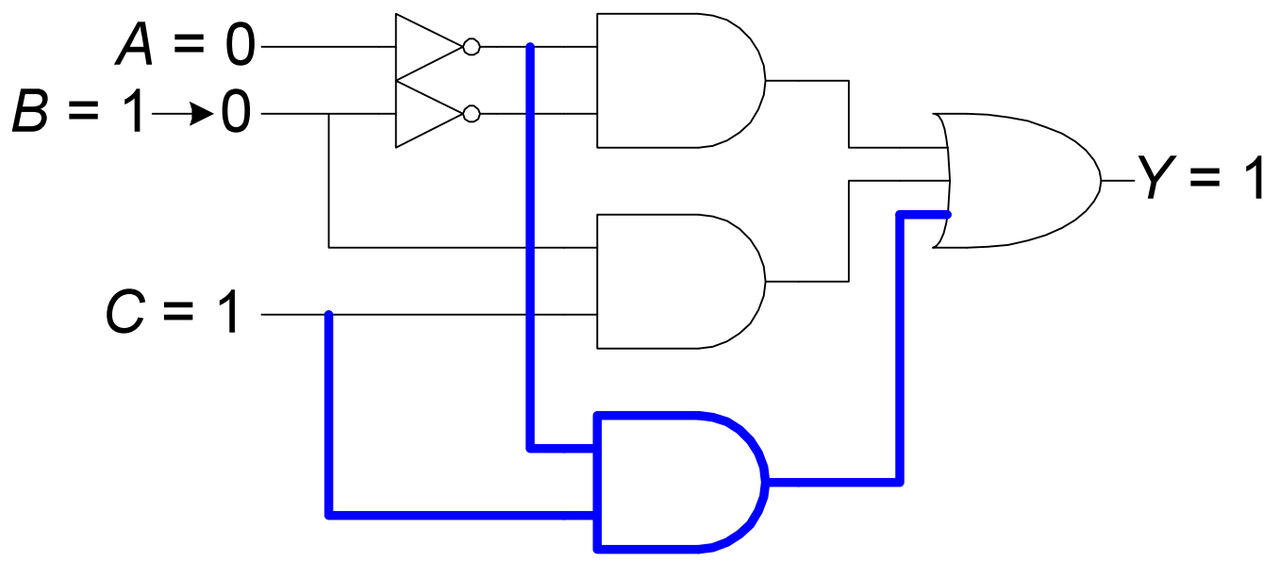
Glitches Example (cont.)





Fixing the Glitch





NB: Can't get rid of all glitches – simultaneous transitions on multiple inputs can also cause glitches