ECE 2300
Digital Logic & Computer Organization
Fall 2016

Combinational Logic Minimization

Cornell University

Example: Prime Number Detector (?)

- F = 1 if number xyz is prime

- Step 1: Lay out truth table

- Step 2: Derive canonical form
  - $F = \sum_{x,y,z}(1,2,3,5,7) = x'y'z + x'yz' + x'yz + xy'z + xyz$
  - $F = \prod_{x,y,z}(0,4,6) = (x+y+z)(x'+y+z)(x'+y'+z)$

- Step 3: Simplify expression
  - Algebraic simplification
  - Systematic minimization (next time)

xyz	F
000	0
001	1
010	1
011	1
100	0
101	1
110	0
111	1

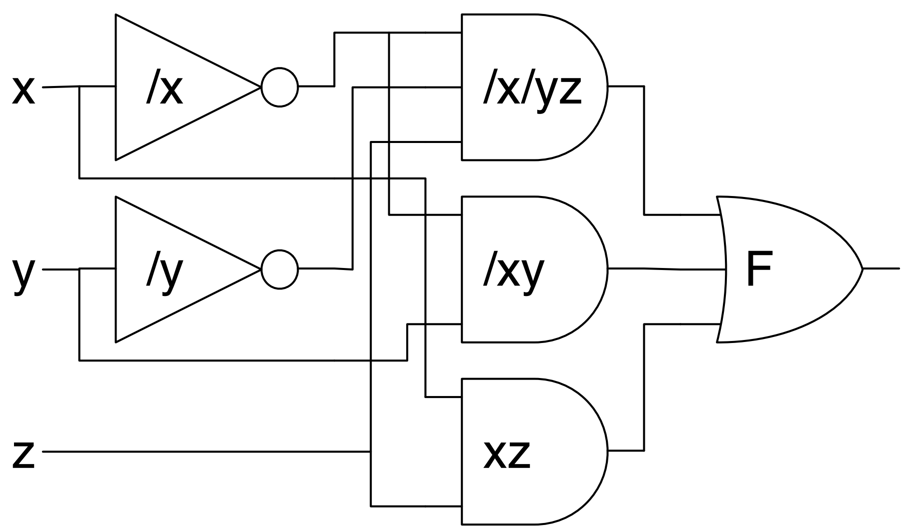
Algebraic Simplification

- $F = \sum_{x,y,z}(1,2,3,5,7)$

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

  $= x'y'z + x'y + xy'z + xyz$ [combining]

  $= x'y'z + x'y + xz$ [combining]



Combinational Logic

- Outputs depend only on current inputs
  - Example: Detect if the input is a prime number

- In contrast, *sequential logic* has “memory” or “state”
  - Example: Detect if the last two inputs are prime
  - We’ll cover sequential logic later

Algebraic Simplification Example

- 1-bit binary adder
  - inputs: A, B, Carry-in
  - outputs: Sum, Carry-out

- Truth Table → Canonical sum

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S = A'•B'•Cin + A'•B•Cin' + A•B'•Cin' + A•B•Cin

Cout = A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin

Idempotency and Combining Theorems

- Idempotency: $X+X = X$
- Combining: $X \cdot Y + X \cdot Y' = X$

Algebraic Simplification Example

Cout = A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin

= A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin + A•B•Cin (idempotency)

= B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin (combining)

= B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin + A•B•Cin (idempotency)

= B•Cin + A•Cin + A•B•Cin' + A•B•Cin (combining)

= B•Cin + A•Cin + A•B (combining)

We can apply these theorems in a more systematic fashion using a *Karnaugh Map*

Karnaugh Map

- Idea: Use combining and idempotency theorems *visually* to simplify canonical forms
- Multidimensional representation of a truth table
- Adjacent cells represent minterms (or maxterms) that differ by one variable
  - Cyclic encoding along each dimension
- At most two variables per dimension

Karnaugh Map for Cout

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Simplifying Cout Using a K Map

Cout = A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin
= A'•B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin + A•B•Cin (idempotency)
= B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin (combining)
= B•Cin + A•B'•Cin + A•B•Cin' + A•B•Cin + A•B•Cin (idempotency)
= B•Cin + A•Cin + A•B•Cin' + A•B•Cin (combining)
= B•Cin + A•Cin + A•B (combining)

Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Ovals result from applying combining theorem

Idempotency theorem allows ovals to overlap

Some K Map Definitions

- **1-cell**: Minterm of a canonical sum
- **Implicant**: Set of adjacent 1-cells
  - Must be a power of 2
- **Prime Implicant**: Implicant that cannot be contained in a larger implicant
- **Distinguished 1-cell**: 1-cell covered by only one prime implicant (*essential prime implicant*)

Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Minimization Using Karnaugh Map

- <u>Goal</u>: Cover all 1-cells with the minimum number of Prime Implicants

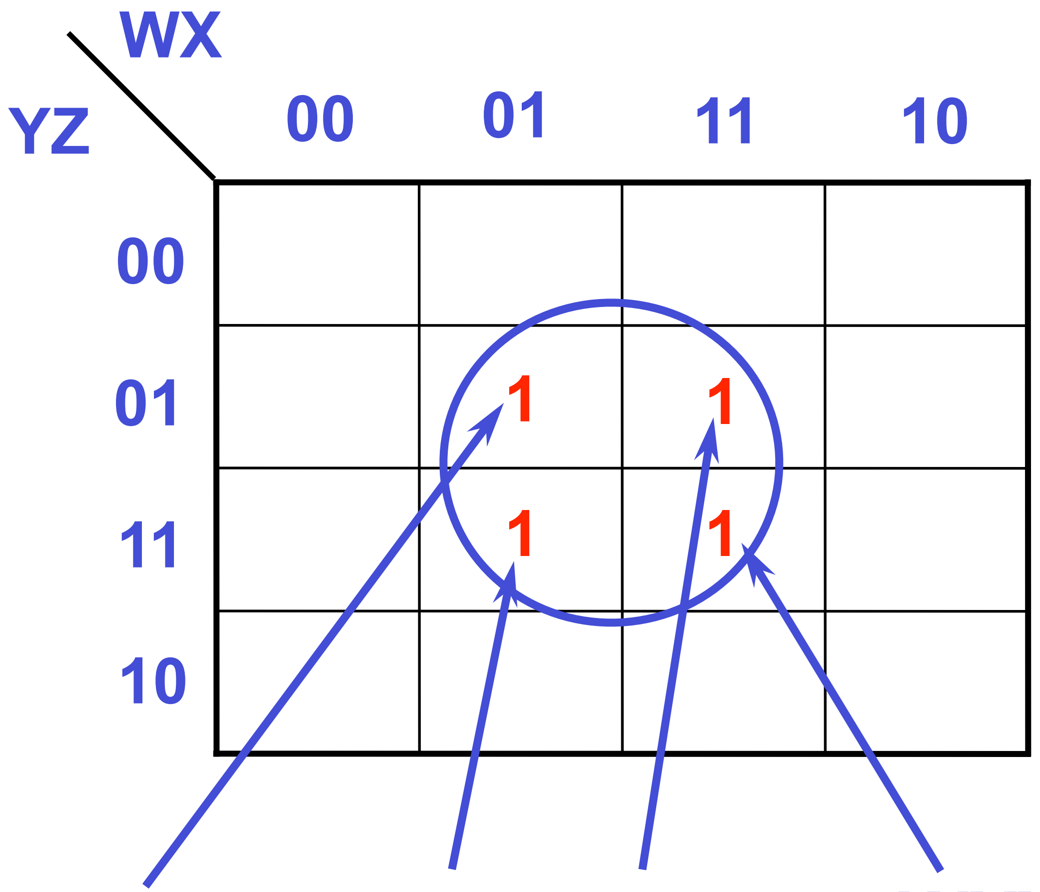
- Procedure
  - Plot 1's corresponding to minterms of function
  - Circle largest possible rectangular sets of 1's
    - Must be power of 2
    - Including "wrap-around" sets
  - Repeat until all minterms are covered
  - OR product terms derived from each circle

- Minimizes number of gates and gate inputs

4 Variable Karnaugh Map

YZ \ WX	00	01	11	10
00				
01				
11				
10				

Combining Theorem in Action

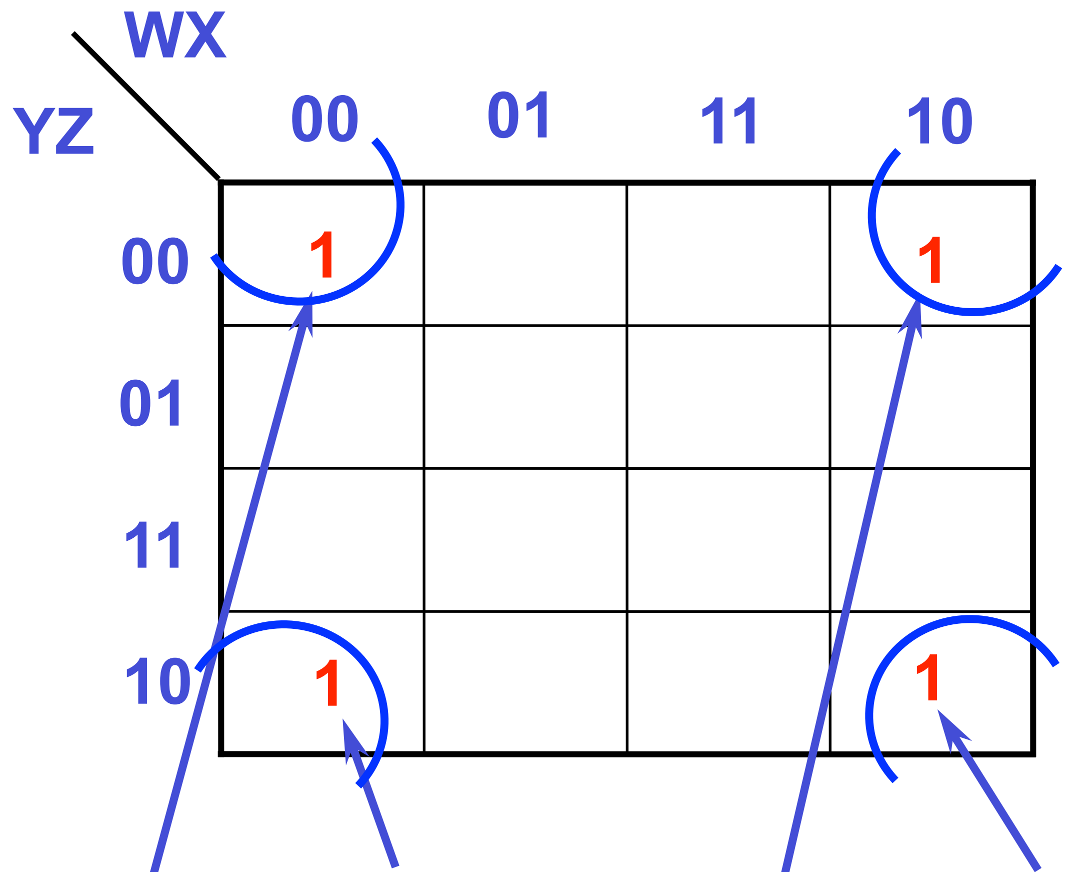


W'XY'Z + W'XYZ + WXY'Z + WXYZ

W'XZ + WXZ

XZ

Combining Theorem in Action



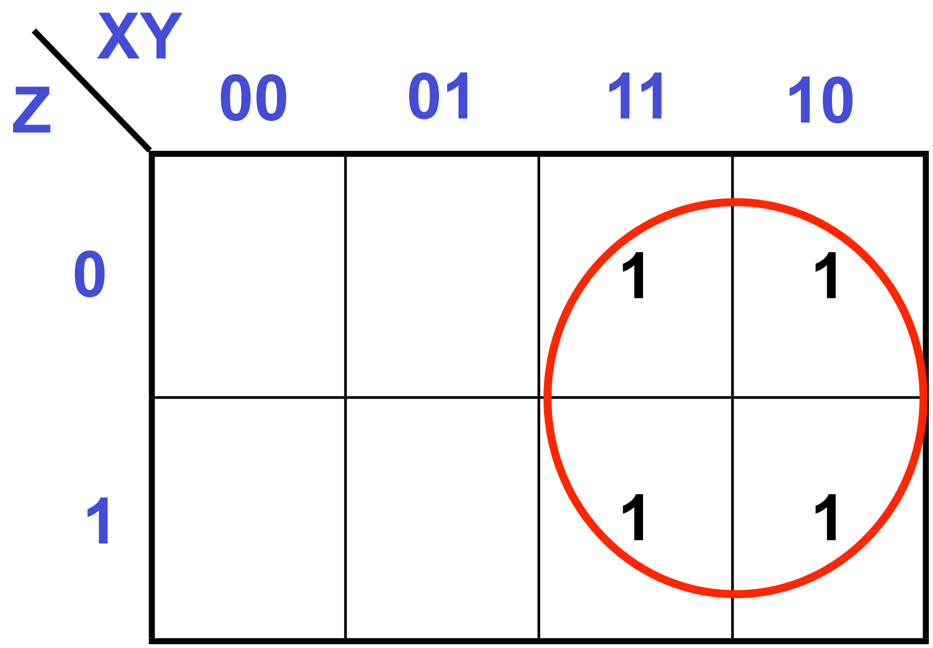
W'X'Y'Z' + W'X'YZ' + WX'Y'Z' + WX'YZ'

W'X'Z' + WX'Z'

X'Z'

3 Variable Karnaugh Map Example

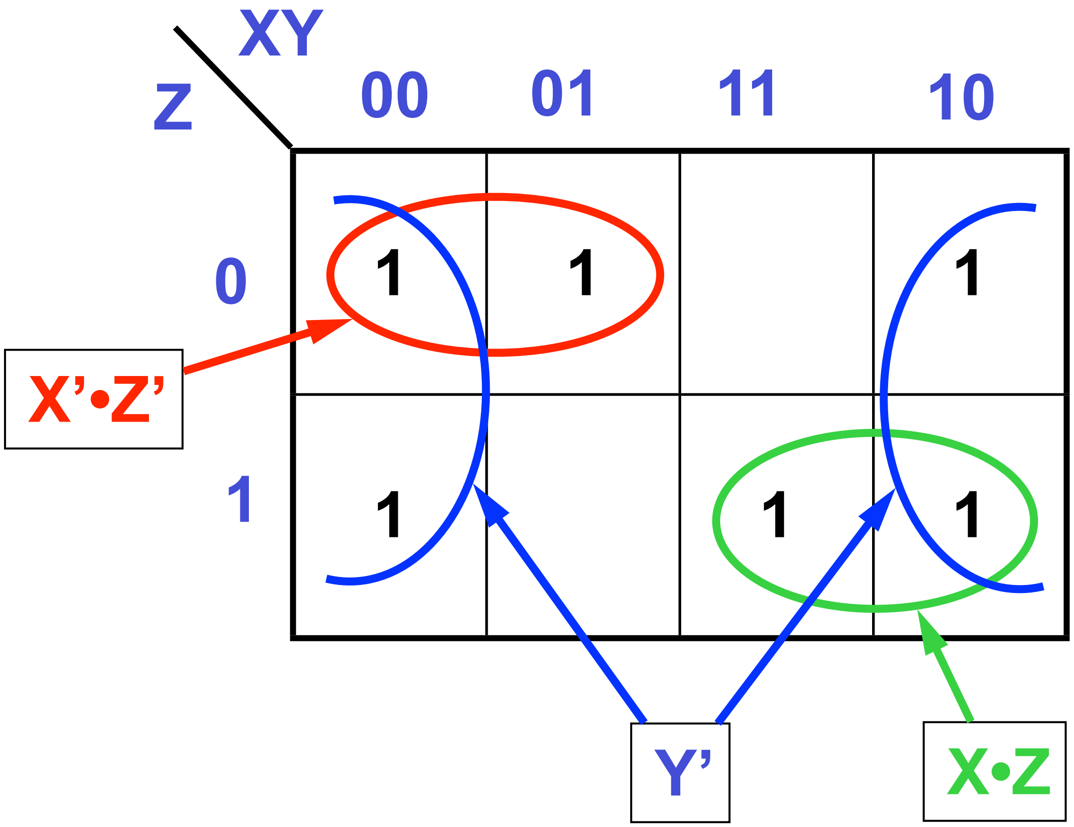
$F=\Sigma_{X,Y,Z}(4,5,6,7)$



F = X

Another Example

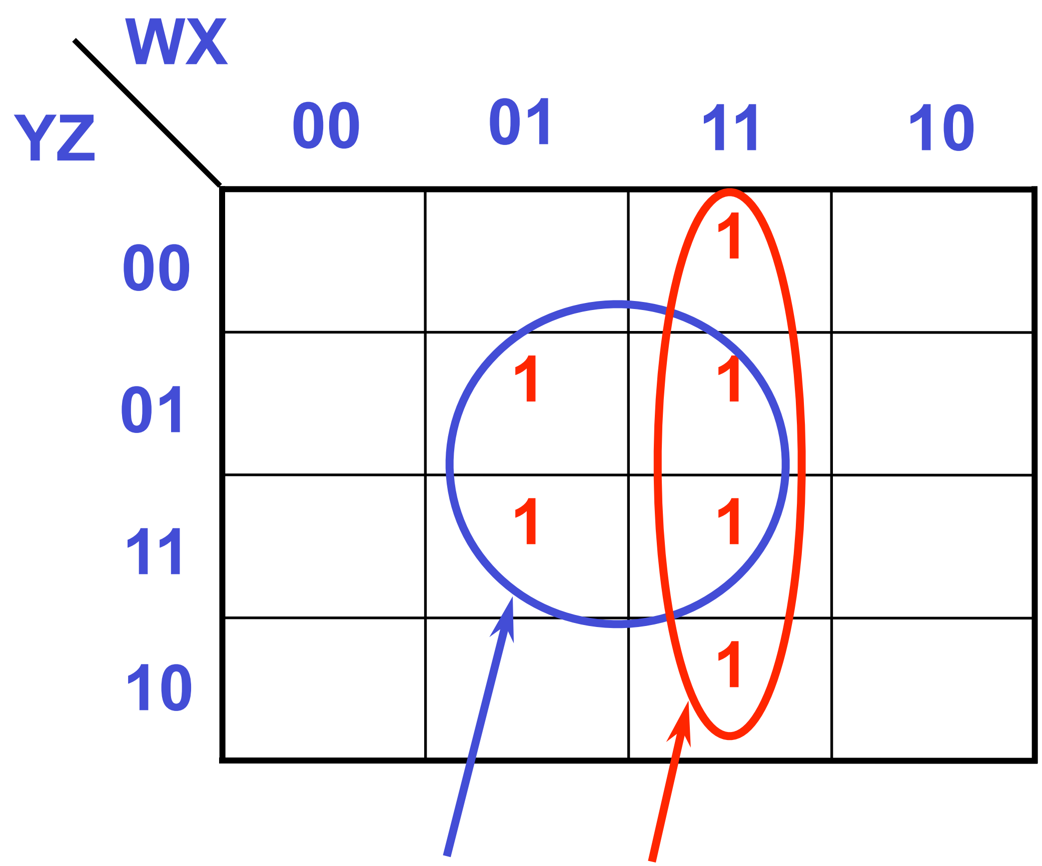
$F=\Sigma_{X,Y,Z}(0,1,2,4,5,7)$



F = X'•Z' + Y' + X•Z

4 Variable Karnaugh Map Example

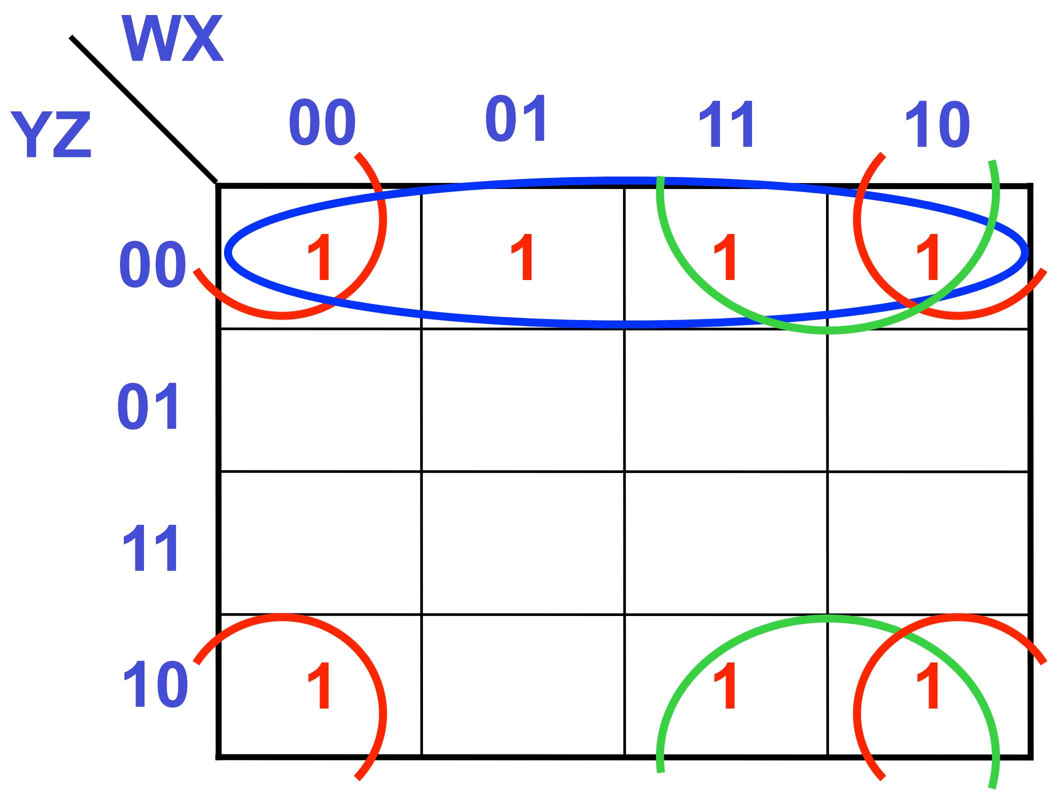
$$F=\Sigma_{W,X,Y,Z}(5,7,12,13,14,15)$$



F = X•Z + W•X

Another Example

- Detect all even digits from 0 to 15 except 6
  - $F=\Sigma_{W,X,Y,Z}(0,2,4,8,10,12,14)$



F = X'Z' + WZ' + Y'Z'

Yet Another

- **Minimize** **$F = \Sigma_{A,B,C,D}(0, 1, 4, 5, 6, 7, 9, 13, 14, 15)$**

CD \ AB	00	01	11	10
00	1	1		
01	1	1	1	1
11		1	1	
10		1	1	

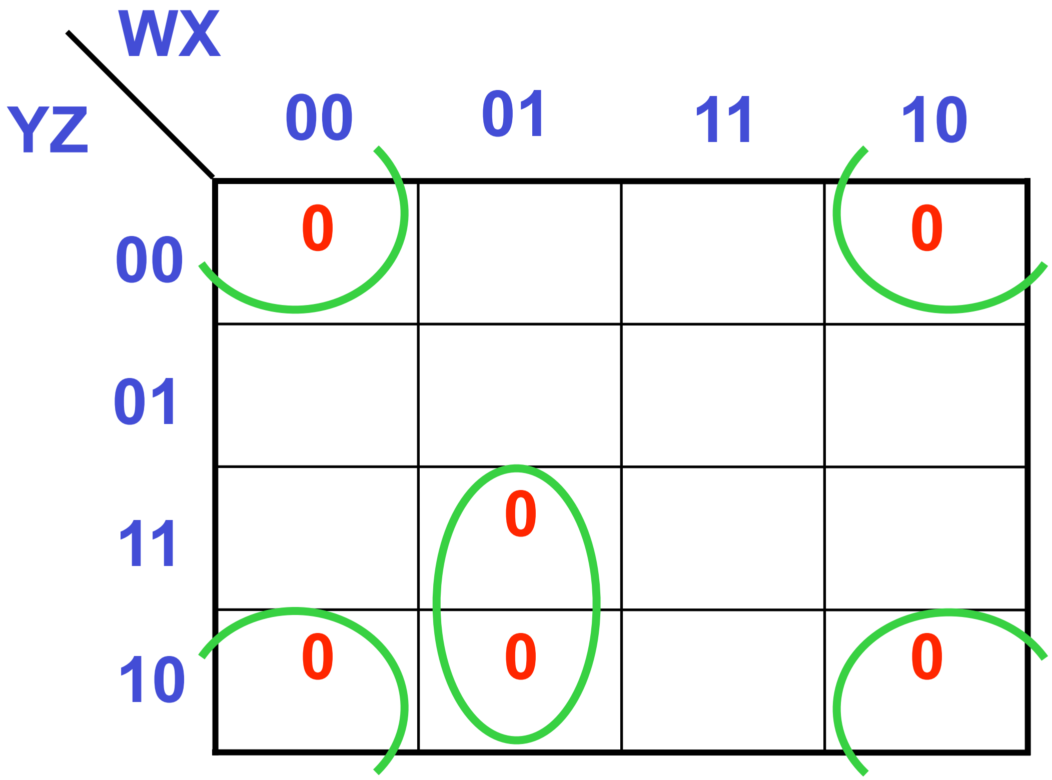
F = A'•C' + B•C + C'•D

Minimizing Product-of-Sums

- Procedure
  - Plot 0’s corresponding to maxterms of function
  - Circle largest possible rectangular sets of 0’s
    - Must be power of 2
    - Including “wrap-around” sets
  - Repeat until all maxterms are covered
  - AND sum terms derived from each circle

Product-of-Sums Example

$F = \Pi_{W,X,Y,Z}(0, 2, 6, 7, 8, 10)$



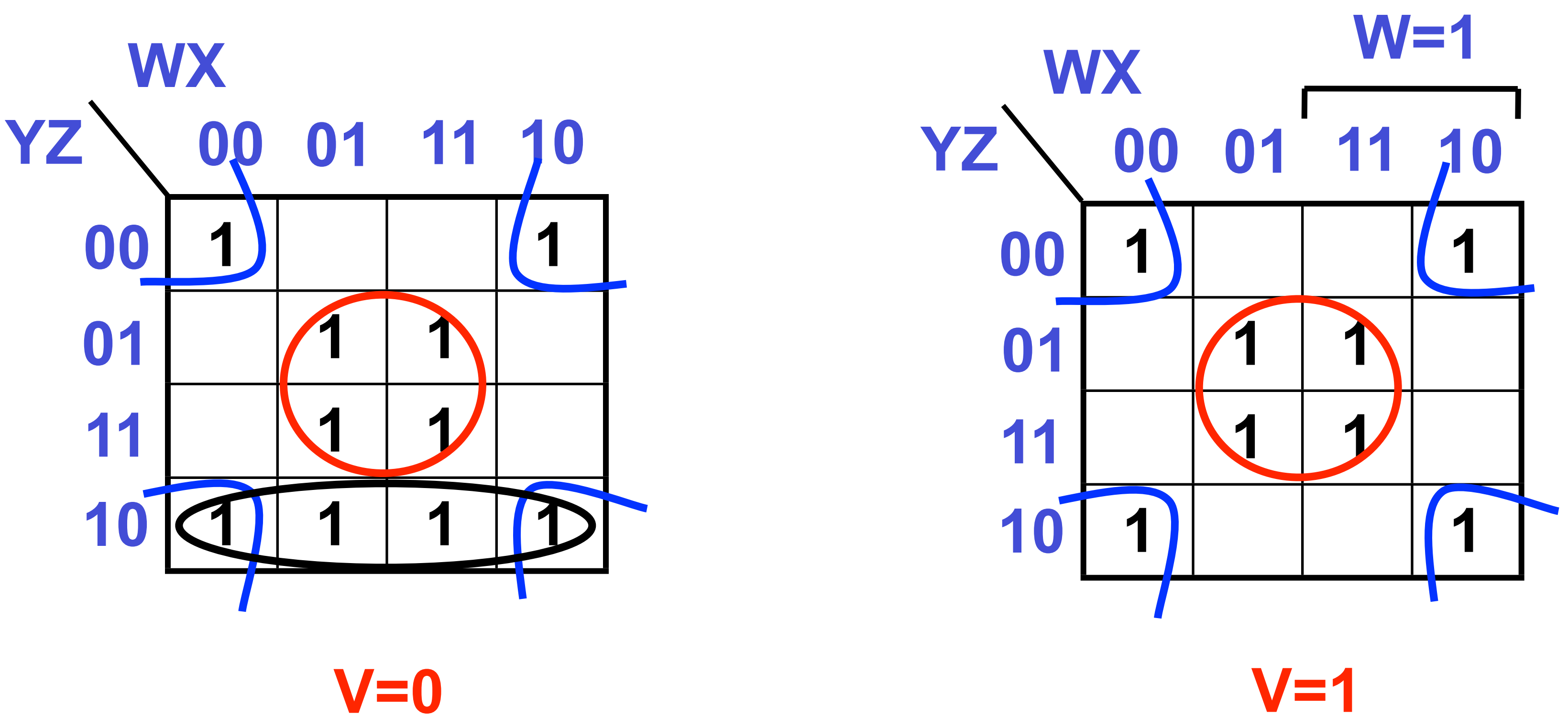
Corners:
X+Z

Other:
W+X'+Y'

$F = (X+Z) \cdot (W+X'+Y')$

5 Variable Karnaugh Maps

$F = \Sigma_{VWXYZ}(0,2,5,6,7,8,10,13,14,15,16,18,21,23,24,26,29,31)$



$F = X' \cdot Z' + X \cdot Z + V' \cdot Y \cdot Z'$

6 Variable Karnaugh Maps

YZ \ WX	00	01	11	10
00				
01				
11				
10				

U,V=0,0

YZ \ WX	00	01	11	10
00				
01				
11				
10				

U,V=0,1

YZ \ WX	00	01	11	10
00				
01				
11				
10				

U,V=1,0

YZ \ WX	00	01	11	10
00				
01				
11				
10				

U,V=1,1

Don’t-Care Combinations

- Sometimes the output for a particular input combination is irrelevant
  - Such as an input combination that will never happen
- Can be used as 1- or 0-cells as needed
- Represent as a ‘d’ in the Karnaugh Map
- Only circle if doing so creates a larger Prime Implicant (and thus a more minimal expression)

Don’t-Care Example

- Detect all even decimal digits except 6
- Only inputs 0-9 appear
- 10-15 are “don’t care” values

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

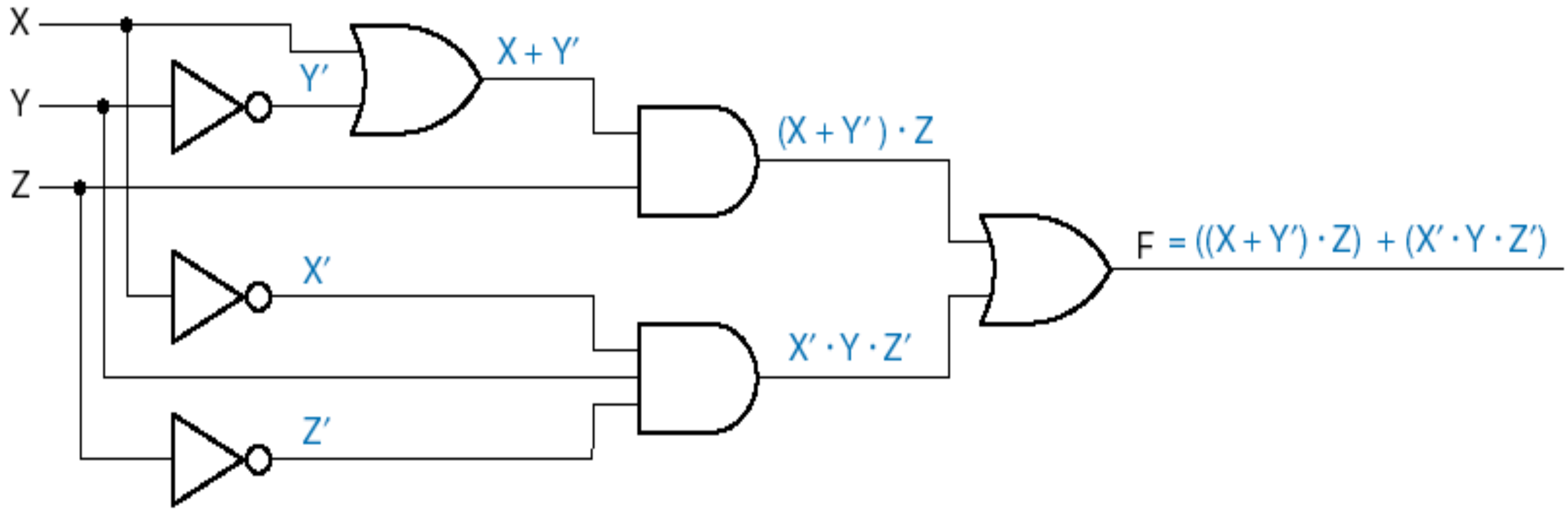
F= Y’•Z’ + X’•Z’

Deriving a Circuit Output Function

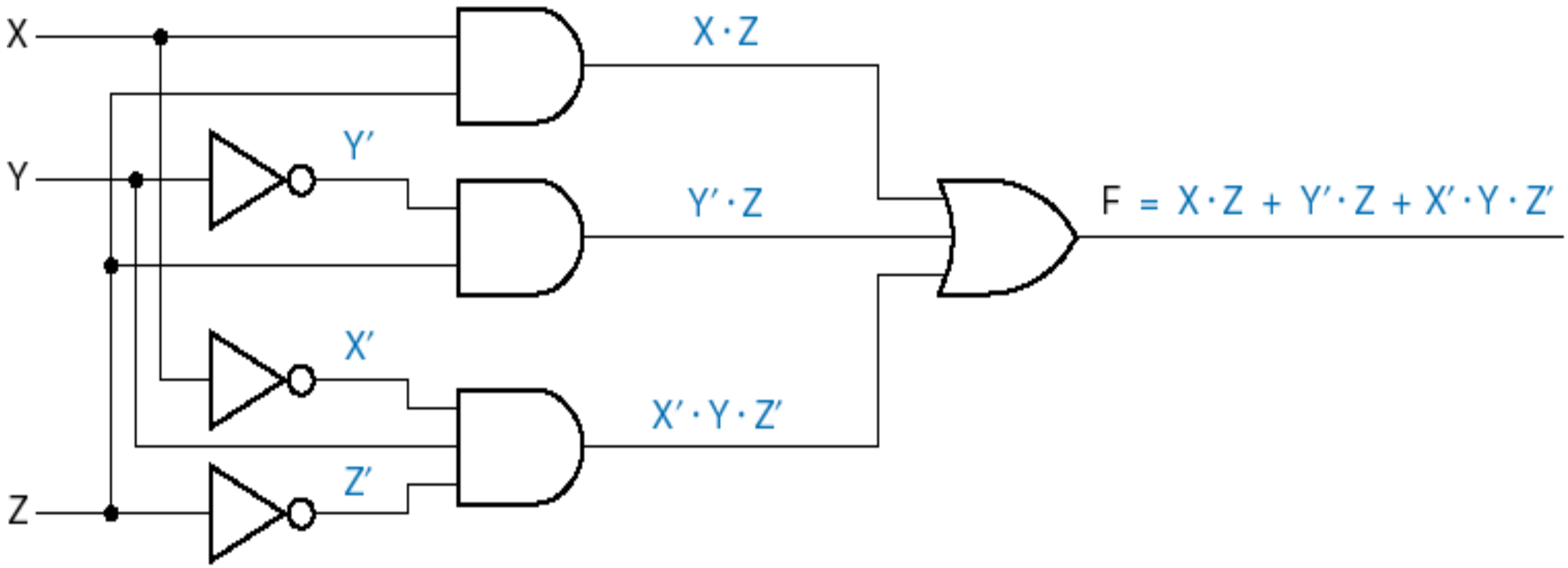
Start from the inputs

Derive intermediate outputs

Carry through to the circuit output



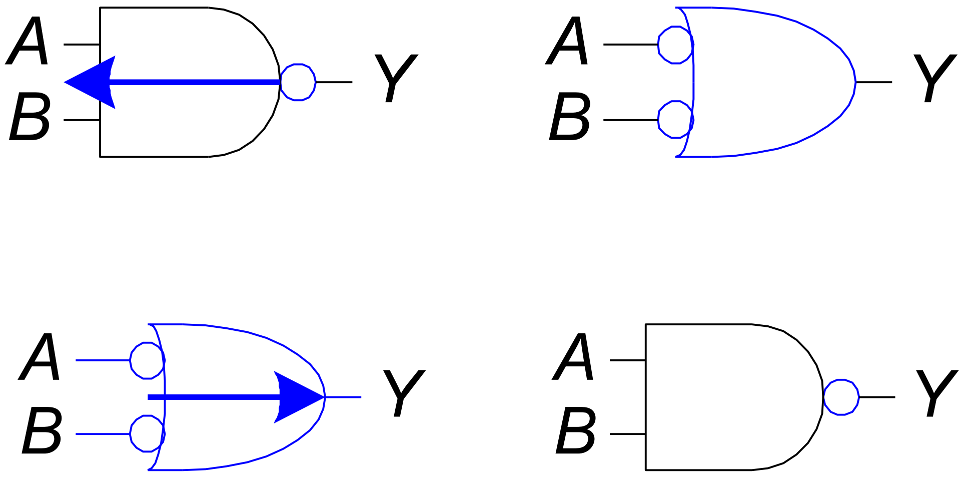
Deriving a Circuit Output Function



Bubble Pushing

- Aid for deriving circuit function

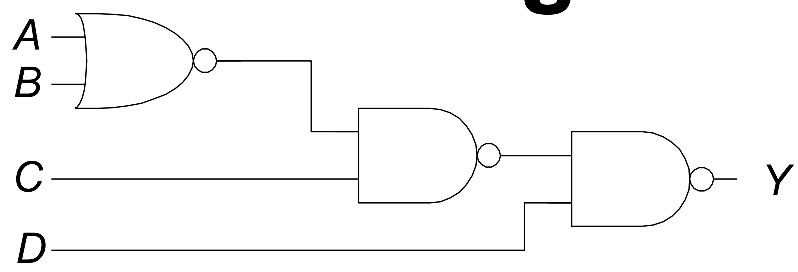
- Makes use of De Morgan to push bubbles from output to inputs, or vice-versa



Bubble Pushing Rules

- Work from output back to inputs
- Push bubble on final output back
- Draw gates in a form so that bubbles cancel

Bubble Pushing Example



$Y = A'B'C + D'$

Before Next Class

- H&H 1.7, 2.8

Next Time

CMOS Logic

Logic Functions