

Digital Design

Chapter 6: Optimizations and Tradeoffs

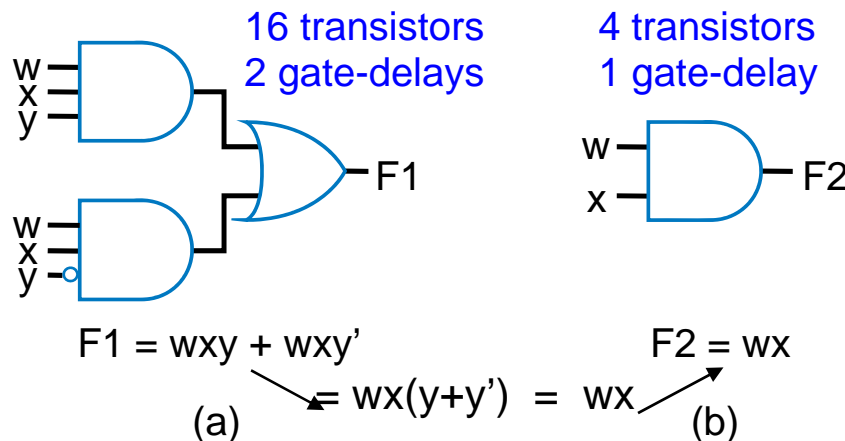
Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
<http://www.ddvahid.com>

Copyright © 2007 Frank Vahid

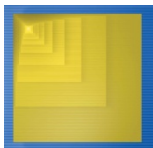
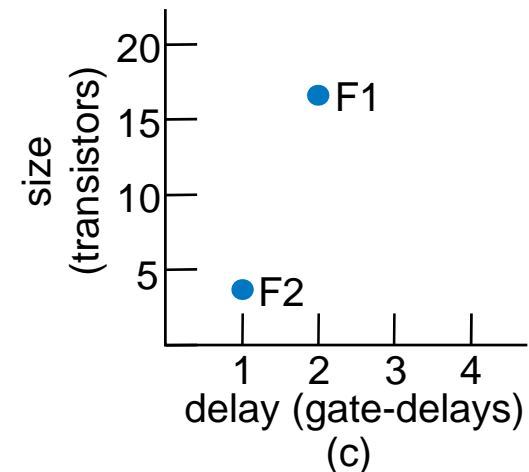
*Instructors of courses requiring Vahid's Digital Design textbook (published by John Wiley and Sons) have permission to modify and use these slides for customary course-related activities, subject to keeping this copyright notice in place and unmodified. These slides may be posted as unanimated pdf versions on publicly-accessible course websites.. PowerPoint source (or pdf with animations) may **not** be posted to publicly-accessible websites, but may be posted for students on internal protected sites or distributed directly to students by other electronic means. Instructors may make printouts of the slides available to students for a reasonable photocopying charge, without incurring royalties. Any other use requires explicit permission. Instructors may obtain PowerPoint source or obtain special use permissions from Wiley – see <http://www.ddvahid.com> for information.*

Introduction

- We now know how to build digital circuits
 - How can we build ***better*** circuits?
- Let's consider two important design criteria
 - **Delay** – the time from inputs changing to new correct stable output
 - **Size** – the number of transistors
 - For quick estimation, assume
 - Every gate has delay of “1 gate-delay”
 - Every gate *input* requires 2 transistors
 - Ignore inverters



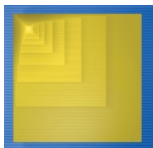
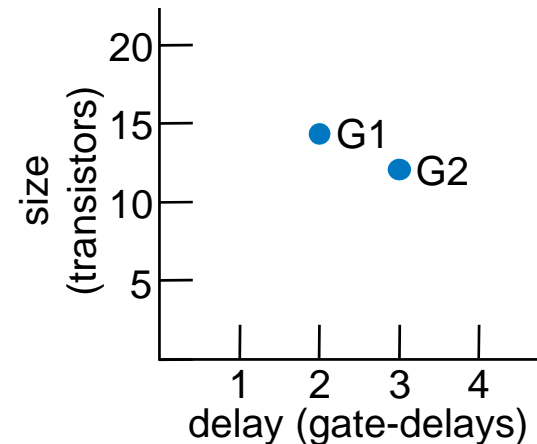
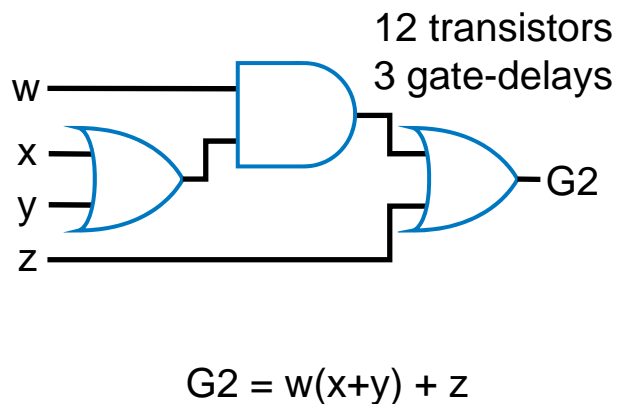
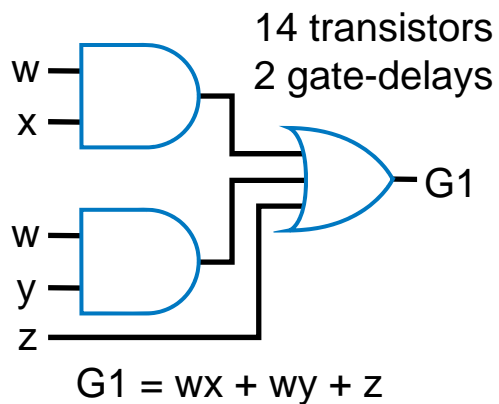
Transforming F1 to F2 represents an **optimization**: Better in all criteria of interest



Introduction

- Tradeoff
 - Improves some, but worsens other, criteria of interest

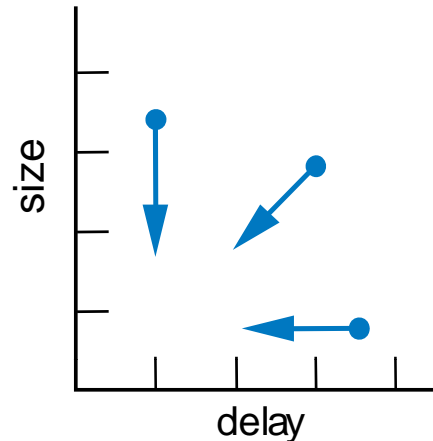
Transforming G1 to G2 represents a **tradeoff**. Some criteria better, others worse.



Introduction

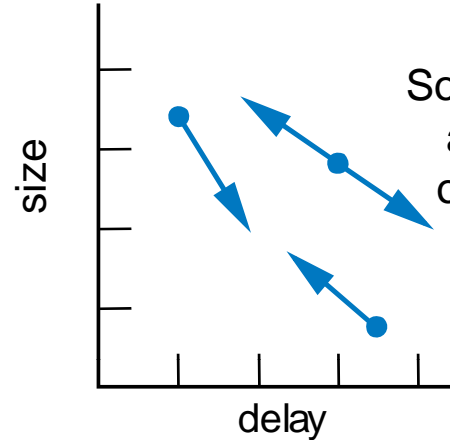
Optimizations

All criteria of interest are improved (or at least kept the same)

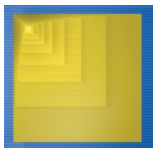


Tradeoffs

Some criteria of interest are improved, while others are worsened



- We obviously prefer optimizations, but often must accept tradeoffs
 - You can't build a car that is the most comfortable, and has the best fuel efficiency, and is the fastest – you have to give up something to gain other things.



Combinational Logic Optimization and Tradeoffs

- Two-level size optimization using algebraic methods
 - Goal: circuit with only two levels (ORed AND gates), with minimum transistors
 - Though transistors getting cheaper (Moore's Law), they still cost something
- Define problem algebraically
 - Sum-of-products yields two levels
 - $F = abc + abc'$ is sum-of-products; $G = w(xy + z)$ is not.
 - Transform sum-of-products equation to have fewest literals and terms
 - Each literal and term translates to a gate input, each of which translates to about 2 transistors (see Ch. 2)
 - Ignore inverters for simplicity

Example

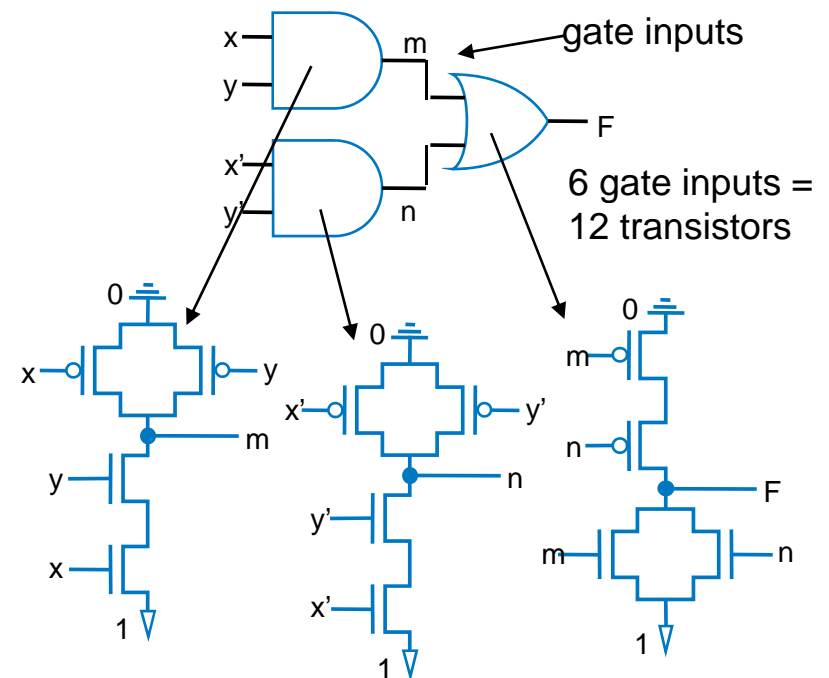
$$F = xyz + xyz' + x'y'z' + x'y'z$$

$$F = xy(z + z') + x'y'(z + z')$$

$$F = xy*1 + x'y'*1$$

$$F = xy + x'y'$$

4 literals + 2 terms = 6



Note: Assuming 4-transistor 2-input AND/OR circuits; in reality, only NAND/NOR are so efficient.



Algebraic Two-Level Size Minimization

- Previous example showed common algebraic minimization method

- (Multiply out to sum-of-products, then)
- Apply following as much possible
 - $ab + ab' = a(b + b') = a \cdot 1 = a$
 - “Combining terms to eliminate a variable”
 - (Formally called the “Uniting theorem”)

$$\begin{aligned} F &= xy\mathbf{z} + xy\mathbf{z'} + x'y'\mathbf{z'} + x'y'\mathbf{z} \\ F &= xy(\mathbf{z + z'}) + x'y'(\mathbf{z + z'}) \\ F &= xy \cdot \mathbf{1} + x'y' \cdot \mathbf{1} \\ F &= xy + x'y' \end{aligned}$$

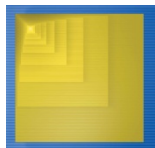
- Duplicating a term sometimes helps

- Note that doesn't change function
 - $c + d = c + d + d = c + d + d + d + d \dots$

$$\begin{aligned} F &= x'y'z' + \mathbf{x'y'z} + x'yz \\ F &= x'y'z' + \mathbf{x'y'z} + \mathbf{x'y'z} + x'yz \\ F &= x'y'(z+z') + x'z(y'+y) \\ F &= x'y' + x'z \end{aligned}$$

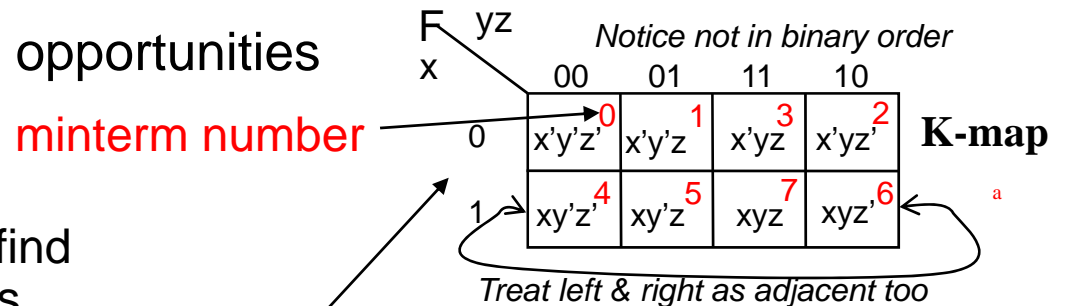
- Sometimes after combining terms, can combine resulting terms

$$\begin{aligned} G &= xy'\mathbf{z'} + xy'\mathbf{z} + xyz + xyz' \\ G &= xy'(\mathbf{z'+z}) + xy(\mathbf{z+z'}) \\ G &= xy' + xy \quad (\text{now do again}) \\ G &= x(\mathbf{y'+y}) \\ G &= x \end{aligned}$$

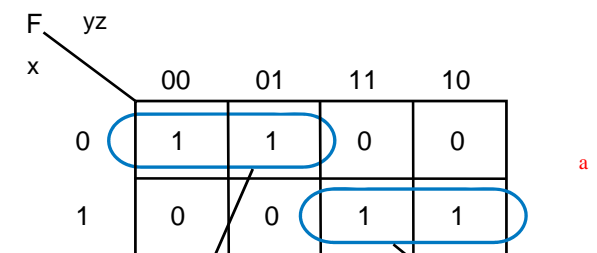
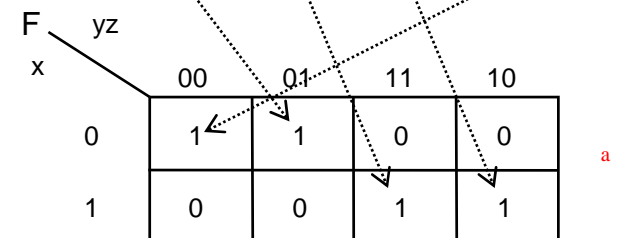


Karnaugh Maps for Two-Level Size Minimization

- Easy to miss “seeing” possible opportunities to combine terms
- **Karnaugh Maps (K-maps)**
 - **Graphical** method to help us find opportunities to combine terms
 - Minterms differing in one variable are *adjacent* in the map
 - Can clearly see opportunities to combine terms – look for adjacent 1s
 - For F, clearly two opportunities
 - Top left circle is *shorthand* for $x'y'z' + x'y'z = x'y'(z' + z) = x'y'(1) = x'y'$
 - Draw circle, write term that has all the literals except the one that changes in the circle
 - Circle xy, $x=1$ & $y=1$ in both cells of the circle, but z changes ($z=1$ in one cell, 0 in the other)
 - Minimized function: OR the final terms



$$F = x'y'z + xyz + xyz' + x'y'z'$$



$$F = x'y' + xy$$

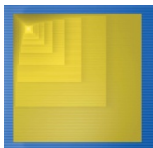
$$F = xyz + xyz' + x'y'z' + x'y'z$$

$$F = xy(z + z') + x'y'(z + z')$$

$$F = xy*1 + x'y'*1$$

$$F = xy + x'y'$$

Easier than all that algebra:



K-maps

- Four adjacent 1s means two variables can be eliminated
 - Makes intuitive sense – those two variables appear in all combinations, so one *must* be true
 - Draw one big circle – *shorthand* for the algebraic transformations above

$$G = xy'z' + xy'z + xyz + xyz'$$

$$G = x(y'z' + y'z + yz + yz') \text{ (must be true)}$$

$$G = x(y'(z' + z) + y(z + z'))$$

$$G = x(y' + y)$$

$$G = x$$

Diagram illustrating a Karnaugh map (K-map) for a function of three variables, x , y , and z . The map is a 2x4 grid. The columns are labeled with yz values: 00, 01, 11, and 10. The rows are labeled with x values: 0 and 1. The cell values are:

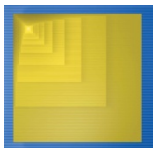
	00	01	11	10
0	0	0	0	0
1	1	1	1	1

The bottom row (where $x = 1$) is circled in blue, indicating a group of four adjacent 1s. This group represents the simplified expression $G = x$.

Draw the biggest circle possible, or you'll have more terms than really needed

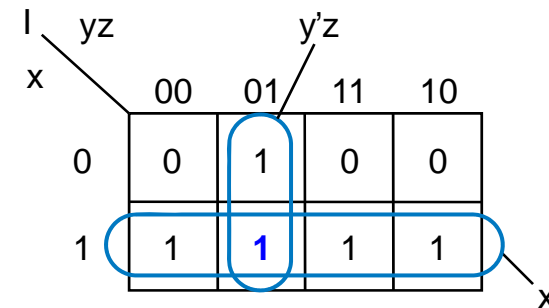
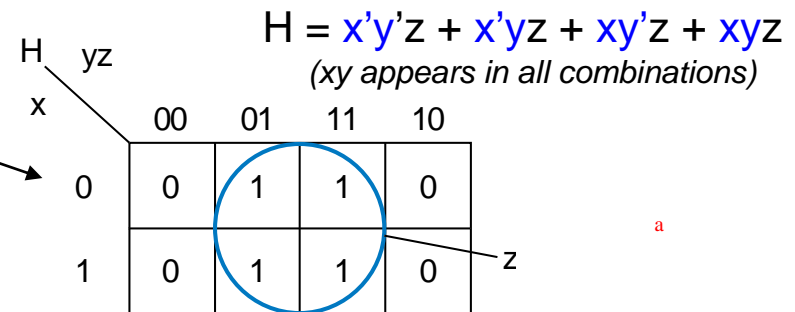
Diagram illustrating a 2x4 Karnaugh map (K-map) for variables x , y , and z . The map shows four adjacent 1s in the bottom row ($x=1$), which are circled in blue to indicate a group. The top row ($x=0$) contains all 0s. The columns are labeled yz (00, 01, 11, 10). The rows are labeled x (0, 1). The map is also labeled with G and xy' on the left, and xy on the right.

		yz				
	x		00	01	11	10
0		0	0	0	0	0
1		1	1	1	1	



K-maps

- Four adjacent cells can be in shape of a square
- OK to cover a 1 twice
 - Just like duplicating a term
 - Remember, $c + d = c + d + d$
- No *need* to cover 1s more than once
 - Yields extra terms – not minimized

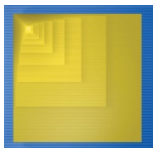
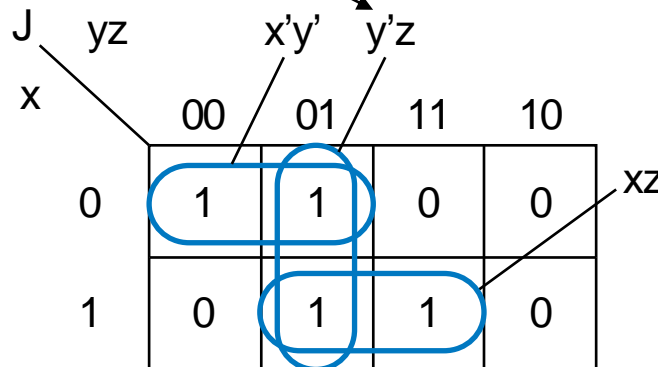


The two circles are shorthand for:

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

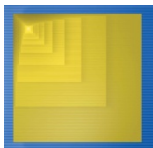
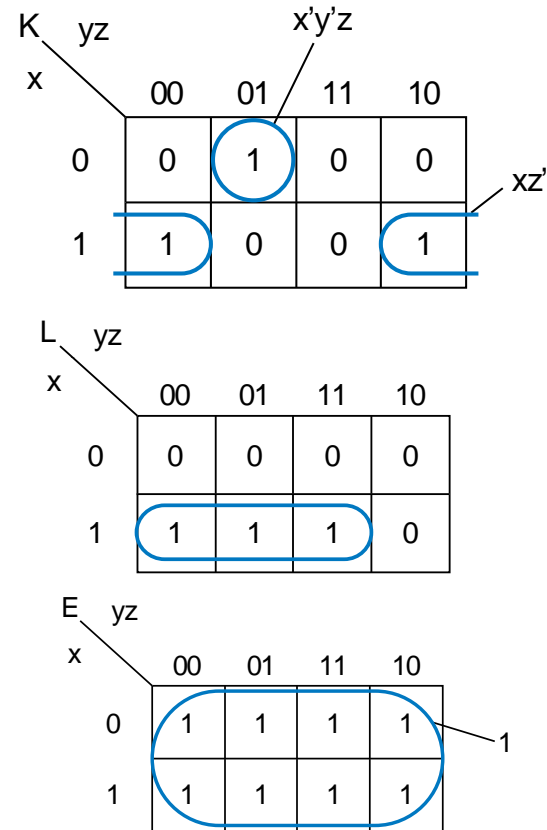
$$I = x'y'z + xy'z + xy'z' + xy'z + xyz + xyz'$$

$$I = (x'y'z + xy'z) + (xy'z' + xy'z + xyz + xyz')$$

$$I = (y'z) + (x)$$


K-maps

- Circles can cross left/right sides
 - Remember, edges are adjacent
 - Minterms differ in one variable only
- Circles must have 1, 2, 4, or 8 cells – 3, 5, or 7 not allowed
 - 3/5/7 doesn't correspond to algebraic transformations that combine terms to eliminate a variable
- Circling all the cells is OK
 - Function just equals 1



Two-Level Size Minimization Using K-maps

General K-map method

1. Convert the function's equation into sum-of-products form
2. Place 1s in the appropriate K-map cells for each term
3. Cover all 1s by drawing the fewest largest circles, with every 1 included at least once; write the corresponding term for each circle
4. OR all the resulting terms to create the minimized function.

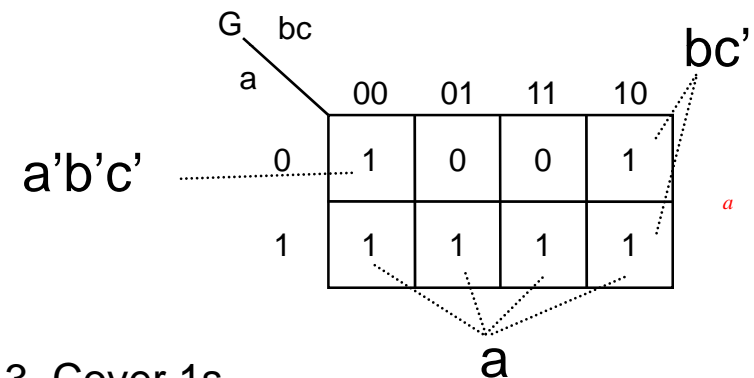
Example: Minimize:

$$G = a + a'b'c' + b'(c' + bc')$$

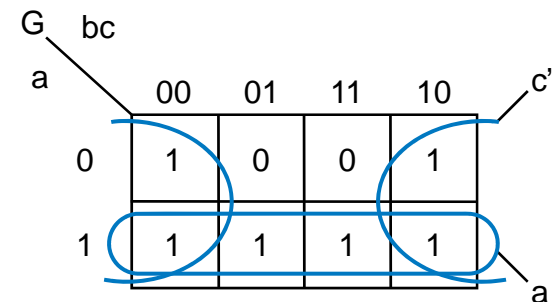
1. Convert to sum-of-products

$$G = a + a'b'c' + bc' + bc'$$

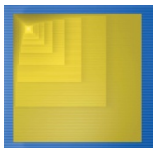
2. Place 1s in appropriate cells



3. Cover 1s



4. OR terms: **$G = a + c'$**



K-maps for Four Variables

- Four-variable K-map follows same principle
 - Adjacent cells differ in one variable
 - Left/right adjacent
 - Top/bottom also adjacent
- 5 and 6 variable maps exist
 - But hard to use
- Two-variable maps exist
 - But not very useful – easy to do algebraically by hand

Diagram of a 2-variable K-map for function F with variables y and z.

	0	1
0		
1		

Diagram of a 4-variable K-map for function F with variables wx and yz.

	yz	00	01	11	10
wx	00	0	0	1	0
	01	1	1	1	0
w'xy'	11	0	0	1	0
	10	0	0	1	0

Groupings in the 4-variable K-map for F:

- A horizontal group of two cells (01, 11) in row 01, labeled $w'xy'$.
- A vertical group of four cells (01, 11) in column 11, labeled yz .

Equation: $F = w'xy' + yz$

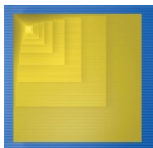
Diagram of a 4-variable K-map for function G with variables wx and yz.

	yz	00	01	11	10
wx	00	0	1	1	0
	01	0	1	1	0
	11	0	1	1	0
	10	0	1	1	0

Groupings in the 4-variable K-map for G:

- A vertical group of four cells (01, 11) in column 11, labeled z .

Equation: $G = z$



Two-Level Size Minimization Using K-maps

– Four Variable Example

- Minimize:
 - $H = a'b'(cd' + c'd') + ab'c'd' + ab'cd' + a'bd + a'bcd'$

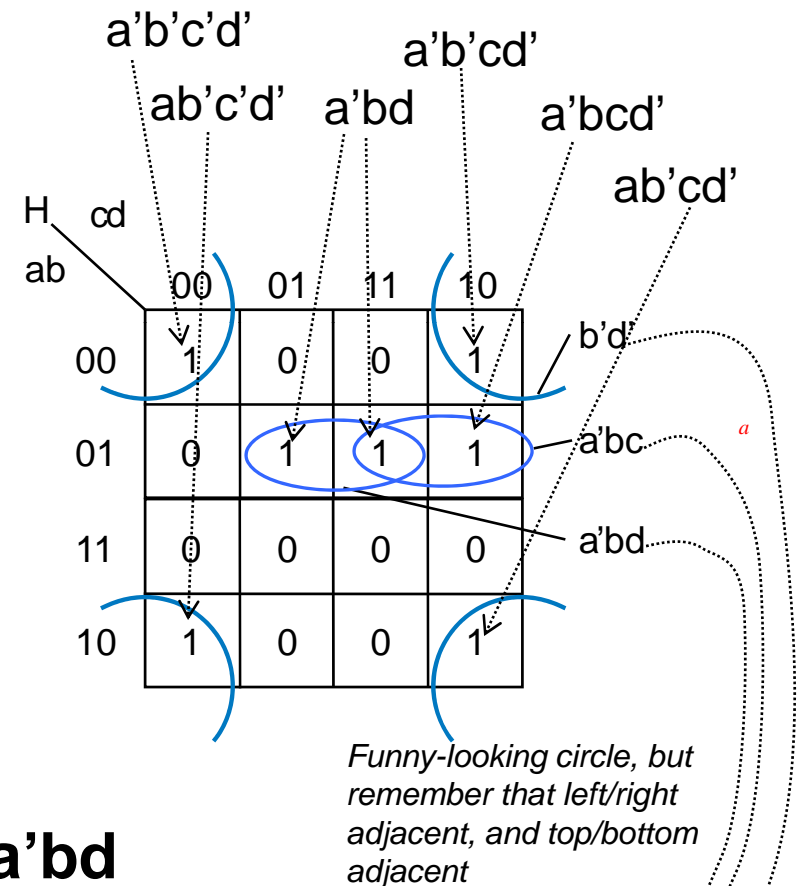
1. Convert to sum-of-products:

$$H = a'b'cd' + a'b'c'd' + ab'c'd' + ab'cd' + a'bd + a'bcd'$$

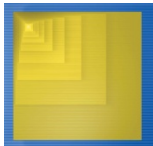
2. Place 1s in K-map cells

3. Cover 1s

4. OR resulting terms



$$H = b'd' + a'bc + a'bd$$



Homework

- Chapter 6: 3, 4, 5
- Homework is due on Thursday, January 29

