



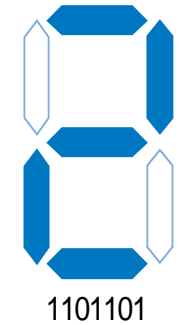
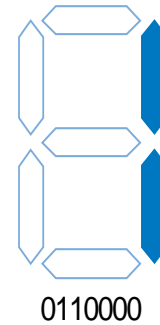
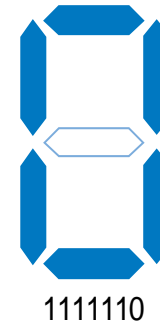
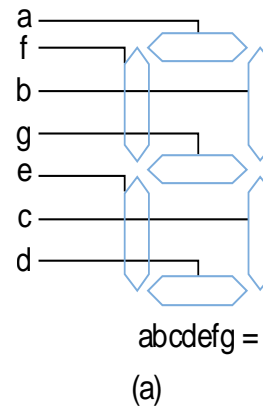
Slides to accompany the textbook *Digital Design*, First Edition,  
by Frank Vahid, John Wiley and Sons Publishers, 2007.  
<http://www.ddvahid.com>

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.

# K Map Problem

TABLE 2-4 4-bit binary number to seven-segment display truth table

w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



Write a boolean equation for the b segment using a Karnaugh map

# Don't Care Input Combinations

- What if particular input combinations can never occur?
  - e.g., Minimize  $F = xy'z'$ , given that  $x'y'z'$  ( $xyz=000$ ) can *never* be true, and that  $xy'z$  ( $xyz=101$ ) can *never* be true
  - So it doesn't matter what  $F$  outputs when  $x'y'z'$  or  $xy'z$  is true, because those cases *will never occur*
  - Thus, make  $F$  be 1 or 0 for those cases *in a way that best minimizes the equation*
- On K-map
  - Draw **X**s for don't care combinations
    - Include X in circle ONLY if minimizes equation
    - Don't include other Xs

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

Good use of don't cares

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

Unnecessary use of don't cares; results in extra term

# Minimization Example using Don't Cares

- Minimize:
  - $F = \underline{a'bc'} + abc' + a'b'c$
  - Given don't cares:  $\underline{a'bc}, abc$
- Note: Use don't cares with caution
  - Must be *sure* that we really don't care what the function outputs for that input combination
  - If we do care, then it is probably safer to set the output to 0

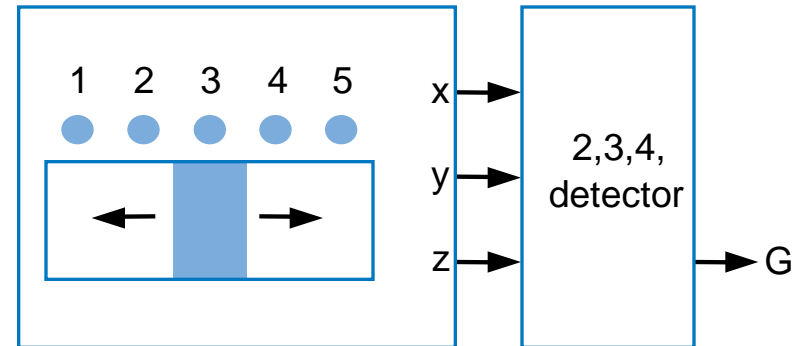
	bc	00	01	11	10
a	0	0	1	X	1
1		0	0	X	1

$$F = a'c + b$$



# Minimization with Don't Cares Example: Sliding Switch

- Switch with 5 positions
  - 3-bit value gives position in binary
- Want circuit that
  - Outputs 1 when switch is in position 2, 3, or 4
  - Outputs 0 when switch is in position 1 or 5
  - Note that the 3-bit input can never output binary 0, 6, or 7
    - Treat as don't care input combinations



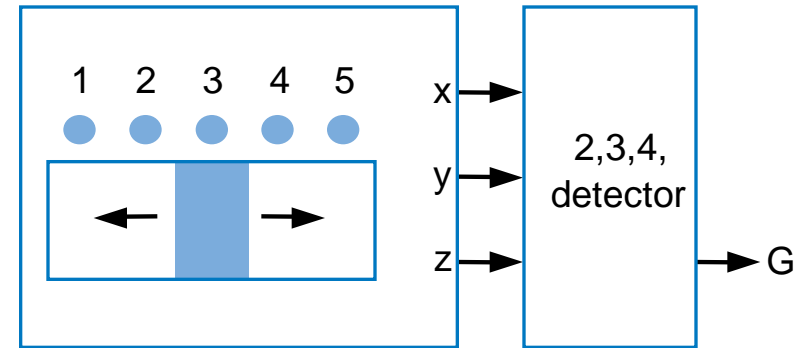
a

a



# Minimization with Don't Care Example: Sliding Switch

- Switch with 5 positions
  - 3-bit value gives position in binary
- Want circuit that
  - Outputs 1 when switch is in position 2, 3, or 4
  - Outputs 0 when switch is in position 1 or 5
  - Note that the 3-bit input can never output binary 0, 6, or 7
    - Treat as don't care input combinations



Without don't cares:

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

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

Labels:  $x'y$  (points to 11, 10),  $xy'z'$  (points to 00, 01)

With don't cares:

$$F = y + z'$$

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

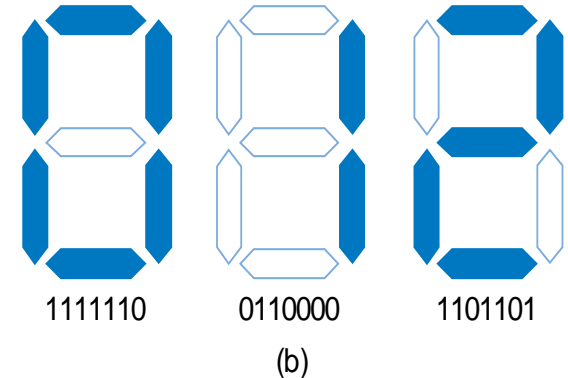
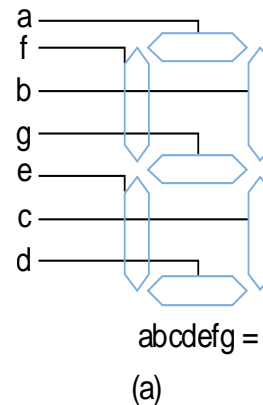
Labels:  $y$  (points to 11, 10),  $z'$  (points to 00, 01)



# K Map Problem

TABLE 2-4 4-bit binary number to seven-segment display truth table

w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



Assume binary-coded decimal input (BCD) only, i.e., 0 – 9. Write a boolean equation for the b segment using a Karnaugh map assuming “don’t care” conditions for 1010 to 1111

# Design Problem

- Problem statement: design the logic circuit for a day/night, setback thermostat.
  - temperature during the day is maintained at 68 degrees
  - temperature during the night is maintained at 58 degrees
- Identify inputs/outputs
  - inputs: DAY, <68, <58
  - outputs: HEAT
- Formulate problem representation
  - Function table
- Formulate Boolean logic using appropriate methods
- Transform logic equations to logic circuit



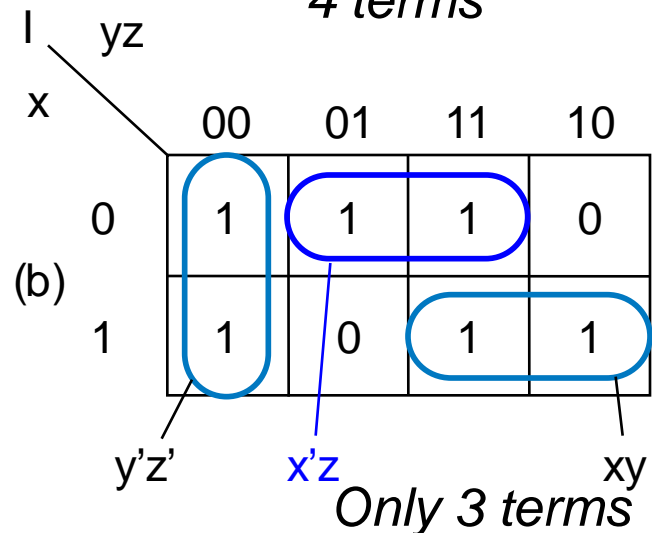
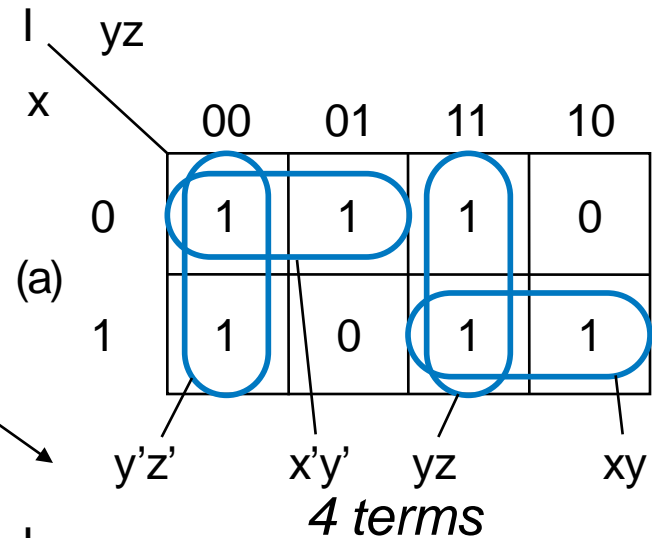
# Temperature Setback

D	<	<	H
A	6	5	E
Y	8	8	A
			T
0	0	0	0
0	0	1	?
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	?
1	1	0	1
1	1	1	1

D	<	<	H
A	6	5	E
Y	8	8	A
			T
0	0	0	0
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	X
1	1	0	1
1	1	1	1

# Automating Two-Level Logic Size Minimization

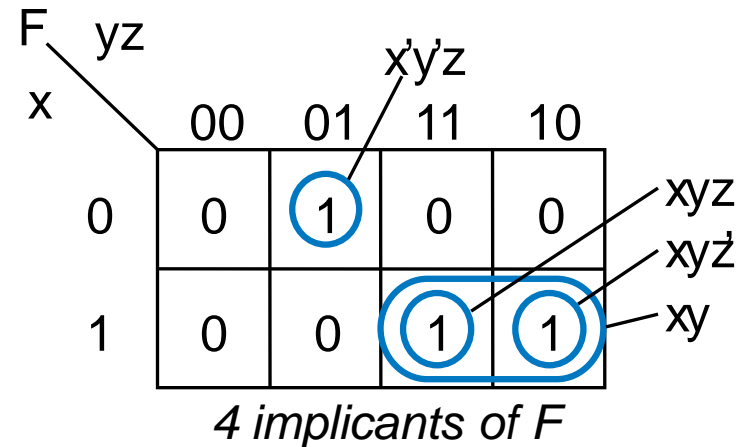
- Minimizing by hand
  - Is hard for functions with 5 or more variables
  - May not yield minimum cover depending on order we choose
  - Is error prone
- Minimization thus typically done by automated tools
  - **Exact algorithm**: finds optimal solution
  - **Heuristic**: finds good solution, but not necessarily optimal



# Basic Concepts Underlying Automated Two-Level Logic Minimization

- Definitions

- **On-set**: All minterms that define when  $F=1$
- **Off-set**: All minterms that define when  $F=0$
- **Implicant**: Any product term (minterm or other) that when 1 causes  $F=1$ 
  - On K-map, any legal (but not necessarily largest) circle
  - Cover: Implicant  $xy$  **covers** minterms  $xyz$  and  $xyz'$
- **Expanding** a term: removing a variable (like larger K-map circle)
  - $xyz \rightarrow xy$  is an expansion of  $xyz$



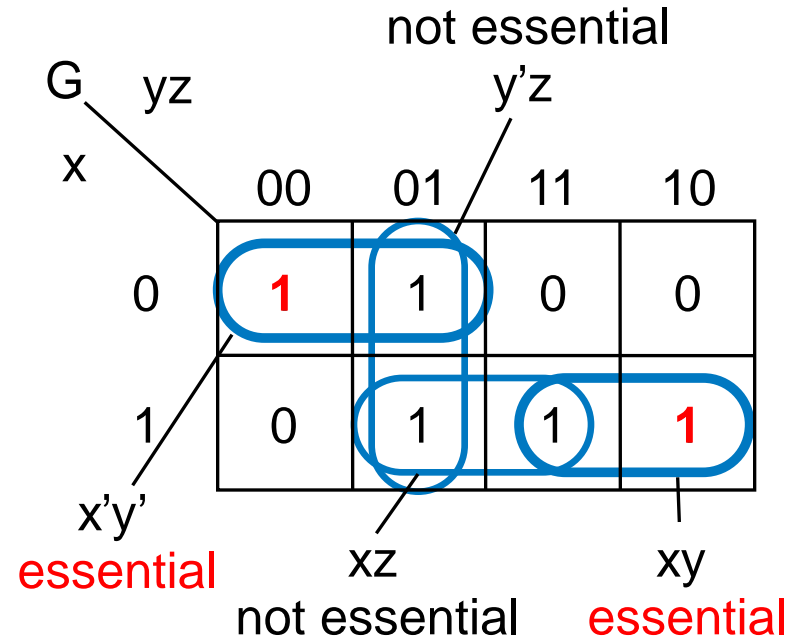
*Note: We use K-maps here just for intuitive illustration of concepts; automated tools do **not** use K-maps.*

- **Prime implicant**: Maximally expanded implicant – any expansion would cover 1s not in on-set
  - $x'y'z$ , and  $xy$ , above
  - But not  $xyz$  or  $xyz'$  – they can be expanded



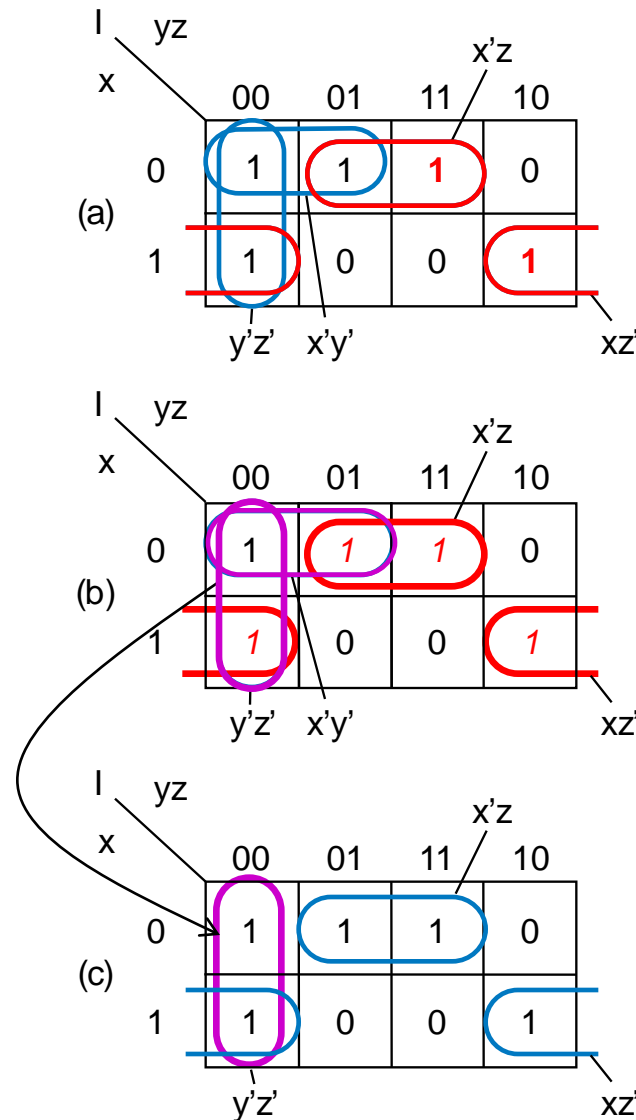
# Basic Concepts Underlying Automated Two-Level Logic Minimization

- Definitions (cont)
  - **Essential prime implicant**: The only prime implicant that covers a particular minterm in a function's on-set
    - Importance: We **must** include **all** essential PIs in a function's cover
    - In contrast, some, but not all, non-essential PIs will be included



# Example of Automated Two-Level Minimization

- 1. Determine all prime implicants
- 2. Add **essential PIs** to cover
  - Italicized 1s are thus already covered
  - Only one uncovered 1 remains
- 3. Cover remaining minterms with non-essential PIs
  - Pick among the two possible PIs



# Problem with Methods that Enumerate all Minterms or Compute all Prime Implicants

- Too many minterms for functions with many variables
  - Function with 32 variables:
    - $2^{32} = 4$  billion possible minterms.
    - Too much compute time/memory
- Too many computations to generate all prime implicants
  - Comparing every minterm with every other minterm, for 32 variables, is  $(4 \text{ billion})^2 = 1$  quadrillion computations
  - Functions with many variables could requires days, months, years, or more of computation – unreasonable



# Homework

- Chapter 6: 6, 7
- Homework is due on Thursday, February 4