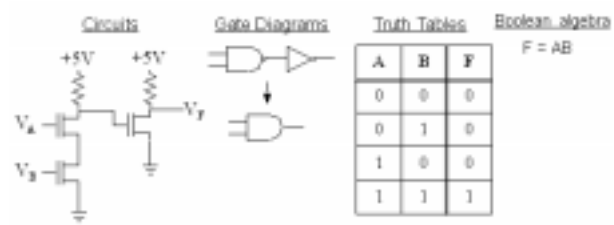
Topic 7: Combinational Circuits

Readings:
•Patterson & Hennessy Appendix B

Goals
•Boolean algebra
•Sum of products and products of sum form
•Karnaugh maps

Physical to Logical Level

4 Representations



•Circuits and gate diagrams correspond one to one
•We can map gate diagrams to TTs. How about TT's into gate diagrams?
Need a tool: boolean algebra
•How to manipulate TTs ?
Translate to boolean algebra and manipulate

Boolean Algebra

Algebra: Set of Elements, Set of Functions, Set of Axioms

•Elements: {0,1}
•Functions: AND (like multiply), OR (like add), NOT (bar)
•Identities:
$0X = 0 \quad 1 + X = 1$
$1X = X \quad 0 + X = X$

•Operator Axioms
• $XX = X \quad X + X = X$
$X\bar{X} = 0 \quad X + \bar{X} = 1$

•AND and OR are commutative:
• $A + B = B + A$, $AB = BA$

•AND and OR obey distributive law:
• $A(B + C) = AB + AC$
$A + (BC) = (A + B)(A + C)$

•AND takes precedence over OR
• $A + BC = A + (BC)$

•AND and OR obey associative law:
• $A + (B + C) = (A + B) + C$
$A(BC) = (AB)C$

DeMorgan's laws

Proving Boolean Equations

Method #1: Algebraic method $(X + Y)(X + Z) = X + YZ$

Method #2: Truth tables $(X + Y)(X + Z) = X + YZ$

X	Y	Z	X+Y	X+Z	LHS	YZ	RHS
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Generating Boolean Equations from Truth Tables

- Find all rows with F=1
- AND them together to make a product term. These are called minterms.
- OR the minterms together F = $\bar{A}BC + A\overline{BC} + ABC$

A	**B**	**C**	**F**	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}BC$
1	0	0	1	$A\overline{BC}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

Boolean Equations from Truth Tables (cont.)

Another Example F = $\overline{ABC}D + \overline{AB}CD + \bar{A}B\overline{CD} + \bar{A}BCD + A\overline{BCD} +$
$A\bar{B}C\bar{D} + A\bar{B}CD + AB\overline{CD} + ABCD$

A	B	C	D	F	
0	0	0	0	0	
0	0	0	1	1	$\overline{ABC}D$
0	0	1	0	0	
0	0	1	1	1	$\overline{AB}CD$
0	1	0	0	1	$\bar{A}B\overline{CD}$
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	1	$\bar{A}BCD$
1	0	0	0	1	$A\overline{BCD}$
1	0	0	1	0	
1	0	1	0	1	$A\bar{B}C\bar{D}$
1	0	1	1	1	$A\bar{B}CD$
1	1	0	0	1	$AB\overline{CD}$
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	$ABCD$

Canonical Form

- Truth Table provides a signature for the boolean function.
  Is there an equivalent algebraic signature?

Sum of Products form (SOP).
- Also known as disjoint normal form or minterm expansion.
- This is what we just did in the last two examples.

Product of Sums form (POS). AKA maxterm expansion.
- Find SOP for rows where F=0: Invert the entire expression, and apply De Morgan's law to get POS. Each sum is called a maxterm.

A	**B**	**C**	**F**	
0	0	0	0	$A+B+C$
0	0	1	0	$A+B+\bar{C}$
0	1	0	0	$A+\bar{B}+C$
0	1	1	1	
1	0	0	1	
1	0	1	0	$\bar{A}+B+\bar{C}$
1	1	0	0	$\bar{A}+\bar{B}+C$
1	1	1	1	

$$(A+B+C)(A+B+\bar{C})(A+\bar{B}+C)$$
$$(\bar{A}+B+\bar{C})(\bar{A}+\bar{B}+C)$$

Canonical Form (cont.)

A	**B**	**C**	**D**	**F**	
0	**0**	**0**	**0**	**0**	$A+B+C+D$
0	**0**	**0**	**1**	**1**	
0	**0**	**1**	**0**	**0**	$A+B+\bar{C}+D$
0	**0**	**1**	**1**	**1**	
0	**1**	**0**	**0**	**1**	
0	**1**	**0**	**1**	**0**	$A+\bar{B}+C+\bar{D}$
0	**1**	**1**	**0**	**0**	$A+\bar{B}+C+D$
0	**1**	**1**	**1**	**1**	
1	**0**	**0**	**0**	**1**	
1	**0**	**0**	**1**	**0**	$\bar{A}+B+C+\bar{D}$
1	**0**	**1**	**0**	**1**	
1	**0**	**1**	**1**	**1**	
1	**1**	**0**	**0**	**1**	
1	**1**	**0**	**1**	**0**	$\bar{A}+\bar{B}+C+\bar{D}$
1	**1**	**1**	**0**	**0**	$\bar{A}+\bar{B}+\bar{C}+D$
1	**1**	**1**	**1**	**1**	

$$(A+B+C+D)(A+B+\bar{C}+D)(A+\bar{B}+C+\bar{D})$$
$$(A+\bar{B}+C+D)(\bar{A}+B+C+\bar{D})(\bar{A}+\bar{B}+C+\bar{D})$$
$$(\bar{A}+\bar{B}+\bar{C}+D)$$

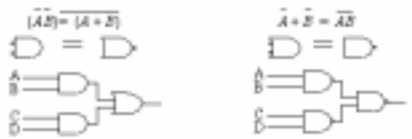
Implementing SOP using only NAND gates

Why just NAND gates?

- AND gates are usually just NAND gates with an inverter.
- INVERTER can be made by wiring together inputs of NAND gates

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

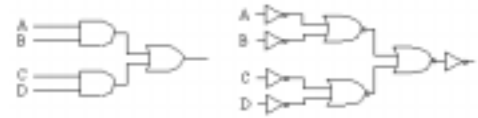
"Pushing a Bubble" through an AND changes it to an OR, and vice versa



Implementing SOP using only NOR gates

What goes for NAND gates, goes for NOR gates:

- Inverters can be made by wiring gates
- OR gate is sometimes implemented as NOR plus INVERTER
- Unfortunately, circuits are not very clean



What about POS using only NOR? $F = (B+\bar{C})(\bar{B}+C)(A+C)$

Canonical Form is not minimal form

Example:

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

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

$F = A + BC$

Question: how to find minimal form?

Finding the minimal form

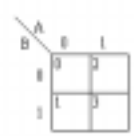
Goals is to reduce the number of literals in a boolean equation.
A literal is a variable and its complement in an equation.

- A1: Use Boolean Algebra. Hard to know when you're "done."
- A2: Use Karnaugh maps

Karnaugh Maps

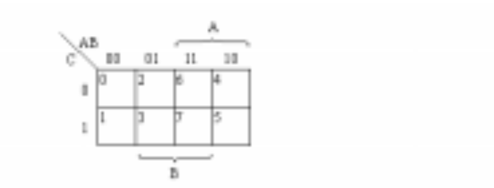
Graphical way to unify terms

Example #1: $F = A\bar{B} + AB$



More Karnaugh Maps

Example #2: $F = \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$

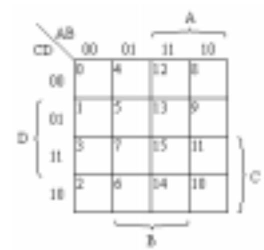


Goal:

- circle as few rectangles of 1's as possible, covering all 1's
- but: rectangle sides must be power-of-two in size
(e.g., 1x1, 1x2, 2x2, 1x4, 2x4, 4x4)

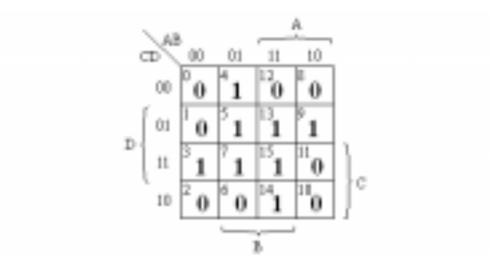
More Karnaugh Maps

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Rules of thumb for finding minimal expressions

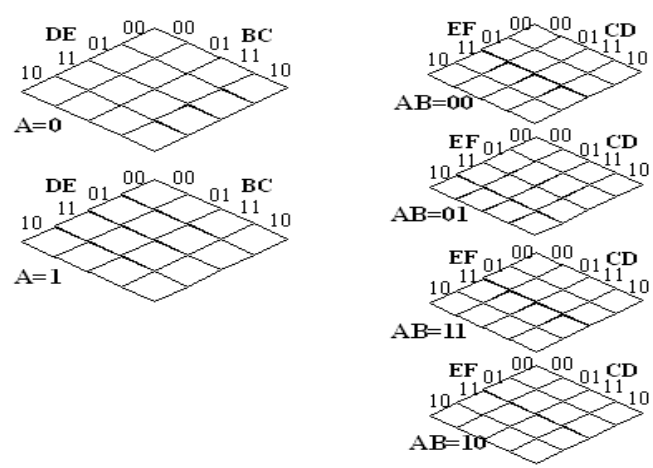
Suppose I have this K-Map



- Largest subcube to smallest?
- Smallest subcube to largest?

5 and 6 variable K-maps

Just a stack of 4-var K-maps



Mapping real problems to boolean equations

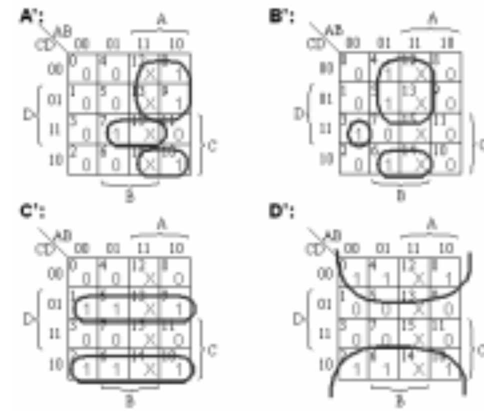
"Design a circuit for a digital clock that computes the next hour on its output, given the current hour as input."

Assumption: Hours are represented as 4 bit binary number

A	B	C	D	A'	B'	C'	D'
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	0
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

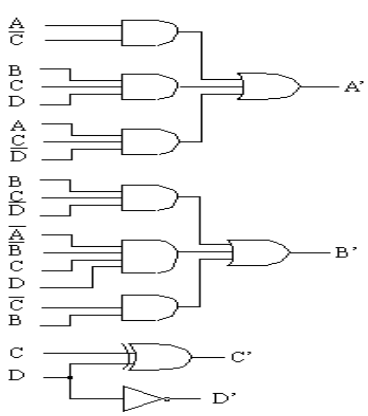
Digital Clock Example

Karnaugh Maps:



not the best!

Digital Clock Circuit



Summary