# Outline

- Last time:
  - Field-Programmable Logic Arrays
  - FPGA-Based FSM Design
  - Intro. to the Xilinx FPGA
- This lecture:
  - Cube Notation
  - Postulates and Theorems of Boolean Algebra
  - De Morgan's Laws
  - Algebraic Simplification
  - Standard Forms, Min-terms, Short Form
  - Design Example: Simplify algebraically
  - Define implicant, prime implicant, cover, prime cover



# Design Example: Seatbelt

**Define variables:**
**S = car starts, B = belts are buckled, D = Door is closed, P = brake is on**

| B | D | P | S′ | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

**S′ = B′D′P′ + B′DP′ + B′DP**
**+ BD′P′ + BD′P + BDP**

**S = B′D′P + BDP′**

# Standard Forms: Definitions

**A *literal* is a variable *or* the complement of a variable. (e.g. X, Y, X')**

**A *product term* is a single literal or the logical product of two or more literals. (e.g. X•Y, W•X'•Z, X)**

**A *sum term* is a single literal or the logical sum of two or more literals. (e.g. X+Y, X, W+X'+Y)**

**A *sum-of-products* expression is a logical sum of product terms.**
**(e.g. X•Y + W'•Y'•Z)**

**A *product-of-sums* expression is a logical product of sum terms.**
**(e.g. (X+Y)•(Z+W+V) )**

**A *normal term* is a product or sum term in which no variable appears more than once.**

**An n-variable *minterm* is a normal product term with n literals.**
**(e.g. X•Y•Z, W•X'•Y)**

**An n-variable *maxterm* is a normal sum term with n literals.**
**(e.g. W'+X'+Y, X+Y+Z)**



# Minterms and Maxterms

- Use the *ones* of the function ("on-set") to determine *minterms* and the *zeros* of the function to determine *maxterm* form.

| X | Y | Z |
|---|---|---|
| 0 | **0** | **0** |
| 0 | **1** | **1** |
| 1 | **0** | **1** |
| 1 | **1** | **0** |

XOR

$$Z(X,Y) = X'\bullet Y + X\bullet Y'$$
$$= \Sigma m(1,2)$$

$$Z'(X,Y) = X'\bullet Y' + X\bullet Y$$
$$Z(X,Y) = (X'\bullet Y' + X\bullet Y)'$$
$$= (X+Y)\bullet(X'+Y')$$
$$= \Pi M(0,3)$$

# Standard Forms

| | B | D | P | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |

$S(B,D,P) = \Sigma m(\quad)$

$S(B,D,P) = \Pi M(\quad)$

- **The notation above is called the *short-form representation* of the Sum-of-Products and Product-of-Sums *canonical forms* respectively.**



# De Morgan's Laws

Given a switching function f(X,Y,...), to obtain its complement f':

(1) Change all + operations to •.

(2) Change all • operations to +.

(3) Complement each variable

e.g. f'(X,Y,Z) = X + X•Y'•Z'•(X•Y + Z•Y')

f(X,Y,Z) = X'•(X'+Y+Z) + (X'+Y')•(Z'+Y)

# Product of Sums: Conjunctive Normal Form

| A | B | C | Maxterms |
|---|---|---|---|
| 0 | 0 | 0 | $A+B+C = M_0$ |
| 0 | 0 | 1 | $A+B+\overline{C} = M_1$ |
| 0 | 1 | 0 | $A+\overline{B}+C = M_2$ |
| 0 | 1 | 1 | $A+\overline{B}+\overline{C} = M_3$ |
| 1 | 0 | 0 | $\overline{A}+B+C = M_4$ |
| 1 | 0 | 1 | $\overline{A}+B+\overline{C} = M_5$ |
| 1 | 1 | 0 | $\overline{A}+\overline{B}+C = M_6$ |
| 1 | 1 | 1 | $\overline{A}+\overline{B}+\overline{C} = M_7$ |

Maxterm Shorthand Notation
for a Function of Three Variables

→ *Maxterm:*

ORed sum of literals in which each variable appears exactly once in either true or complemented form, but not both!

→ *Maxterm form:*

Find truth table rows where *f* is 0
0 in input column implies true literal
1 in input column implies complemented literal

**$f$(A,B,C) = ΠM(0,1,2)**
**= (A + B + C) (A + B + C') (A + B' + C)**

**$f'$(A,B,C) = ΠM(3,4,5,6,7)**
**= (A + B' + C) (A' + B + C) (A' + B + C') (A' + B' + C) (A' + B' + C')**



# Design Example: The Hot-tub

**George is concerned as to whether he should warm up the hot-tub for tonight's party. Under what conditions will someone use the tub?:**

**(a) Alice never bathes**

**(b) Bill bathes, iff both Alice and Don are present**

**(c) Cindy bathes under all conditions - even by herself!**

**(d) Don bathes iff Alice is not present.**

# Design Example: The Hot-tub

George is concerned as to whether he should warm up the hot-tub for tonight's party. Under what conditions will someone use the tub?:

(a) Alice never bathes

(b) Bill bathes, iff both Alice and Don are present

(c) Cindy bathes under all conditions - even by herself!

(d) Don bathes iff Alice is not present.

---

A = Alice is present, B = Bill is present, C = Cindy is present, D = Don is present, T = someone will use the tub.

T = BAD + C + DA'

= ?

Word Statement: The hot-tub will be used if



# Fundamental Theorems & Postulates of Boolean Algebra

| | | | | |
|---|---|---|---|---|
| **Identities:** | **(1)** | **X + 0 = X** | **(1D)** | **X•1 = X** |
| **Null Elements:** | **(2)** | **X + 1 = 1** | **(2D)** | **X•0 = 0** |
| **Indempotency:** | **(3)** | **X + X = X** | **(3D)** | **X•X = X** |
| **Involution (Double Negation):** | | | **(4)** | **(X')' = X** |
| **Complements:** | **(5)** | **X + X' = 1** | **(5D)** | **X•X' = 0** |
| **Commutativity:** | **(6)** | **X + Y = Y + X** | **(6D)** | **X•Y = Y•X** |
| **Associativity:** | **(7)** | **(X+Y)+Z = X+(Y+Z)** | **(7D)** | **(X•Y)•Z = X•(Y•Z)** |
| **Distributivity:** | **(8)** | **X•Y+X•Z = X•(Y+Z)** | **(8D)** | **(X+Y)•(X+Z) = X+Y•Z** |
| **Combining:** | **(9)** | **X•Y + X•Y' = X** | **(9D)** | **(X+Y)•(X+Y') = X** |
| **Covering:** | **(10)** | **X + X•Y = X** | **(10D)** | **X•(X+Y) = X** |
| **DeMorgan's Laws:** | **(12)** | **(X•Y•Z)' = X'+Y'+Z'** | **(12D)** | **(X+Y+Z)' = X'•Y'•Z'** |
| **Consensus:** | **(17)** | **X•Y+X'•Z+Y•Z = X•Y+X'•Z** | **(17D)** | **(X+Y)•(X'+Z)•(Y+Z) = (X+Y)•(X'+Z)** |

**Shannon Expansion:** **(18)** **F(X,Y,Z) = X•F(1,Y,Z) + X'•F(0,Y,Z)**

**(18D)** **F(X,Y,Z) = (X+F(0,Y,Z))•(X'+F(1,Y,Z))**

# Design Example: The Hot-tub

Reduce using fundamental theorems & postulates (label equation on right side from table)

T = BAD + C + DA′

=

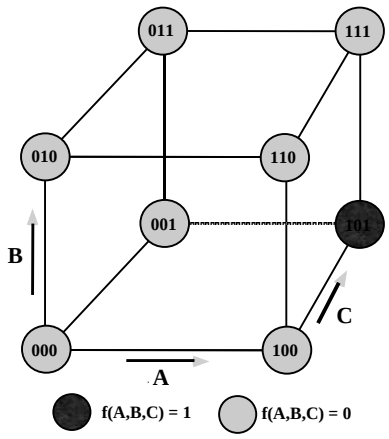
=

=

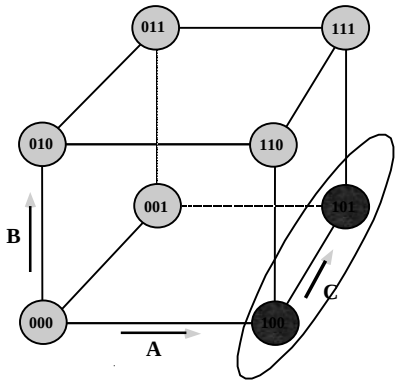
Word statement: The tub should be warmed up iff:



# Cubes as Product Terms

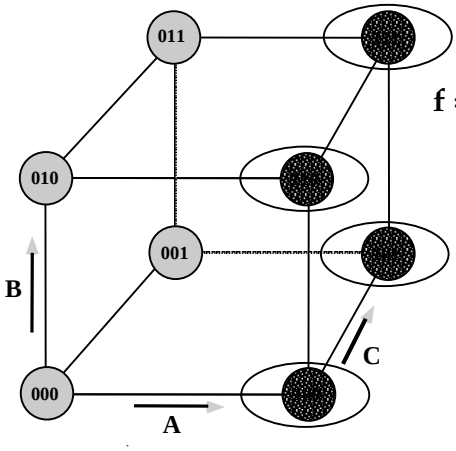



**f = AB′C**




**f = AB′C + AB′C′**
**= AB′**

# f (A,B,C) = A
# Using Minterms (4 cubes)




**f = AB'C' + AB'C + ABC' + ABC**

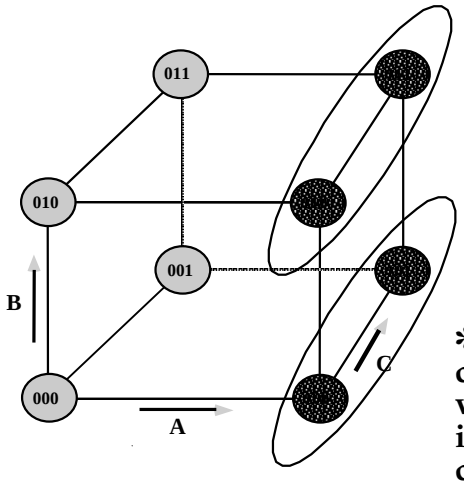
Alternate format:

| A | B | C | f |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**ON-set or ON-cover**



# f(A,B,C) = A
# Two cubes




**f = AB + AB'**

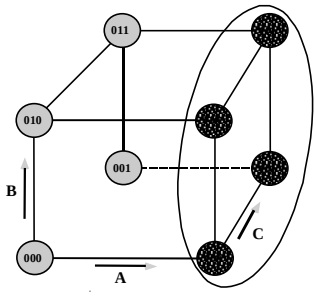
**Alternate format:**

| A | B | C | f |
|---|---|---|---|
| 1 | **1** | * | **1** |
| 1 | **0** | * | **1** |

**Input "don't cares"**

*** = "don't care": The cube is independent of that variable; we "don't care" what it's value is - it doesn't matter c.f. PLA "don't connects" from last week!**

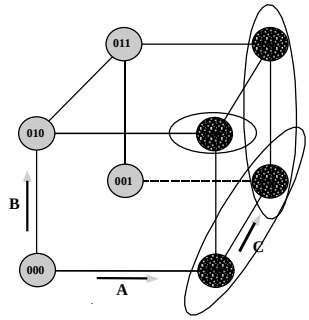
# f(A,B,C) = A
# One Cube and Mixed Cover




**f = A**

**Alternate format:**

| A | B | C | f |
|---|---|---|---|
| 1 | * | * | 1 |



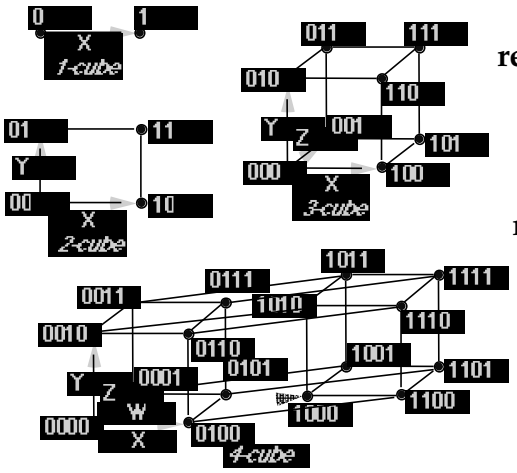

**f = ?**

**Alternate format:**

| A | B | C | f |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 1 | 0 | * | 1 |
| 1 | * | 1 | 1 |



# Boolean Hypercubes

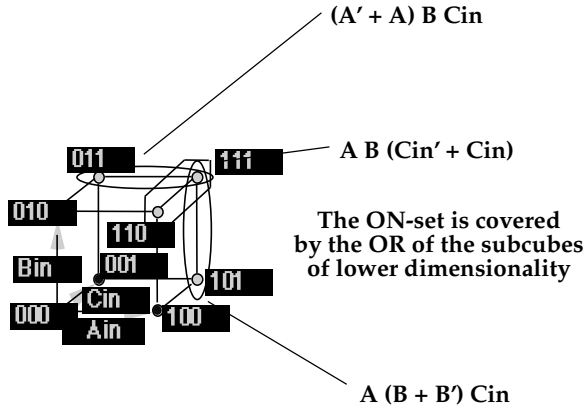



**Just another way to represent the truth table**

**n input variables = n dimensional "cube"**

# Relationship to Cube Notation: Full Adder

| Ain | Bin | Cin | Cout |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

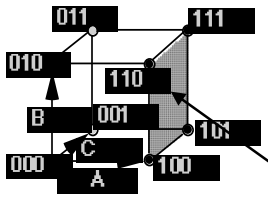



***Cout = B Cin + A B + A Cin***



# Sub-cubes of Higher Dimensionality

**$f$(A,B,C) = Σm(4,5,6,7)**




**On-set forms a rectangle, i.e., a cube of two dimensions**

***represents an expression in one variable i.e., 3 dimensions - 2 dimensions***

**A is asserted and unchanged B and C vary**

**This subcube represents the literal A**

# Describing Logic Functions

❍**Definitions:**

- ➜ Adjacent pairs, quadruples, octuples, ... of minterms (a vertex marked 1) in the Boolean space are called *implicants.*
- ➜ An implicant is called a *prime implicant* if it is not a subset of another implicant of the function.
- ➜ Any prime implicant which includes a 1 vertex that is not included in any other prime implicant is called an *essential prime implicant*.
- ➜ A set of prime implicants of a function is called a *minimal sum-of-products form (minimal "cover")* of the function if the following conditions are satisfied:
  - (a) It includes the entire on-set (1 verticies) of the function.
  - (b) Condition (a) will not be satisfied if any one of the prime implicants is removed.
  - (c) None of the prime implicants can be replaced with a simpler prime implicant (i.e. one containing fewer variables).