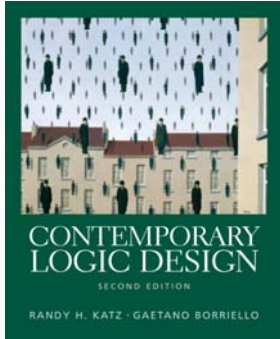


CSE 370 Spring 2006

Introduction to Digital Design

Lecture 5: Canonical Forms



Last Lecture

- Logic Gates
- Different Implementations
- Bubbles

Today

- Canonical Forms
- Sum of Products
- Product of Sums
- Boolean Cubes

Administrivia

- Homework 2 was modified Monday evening. See online for the modification. The two problems that were dropped will appear on the next homework.

Puzzle

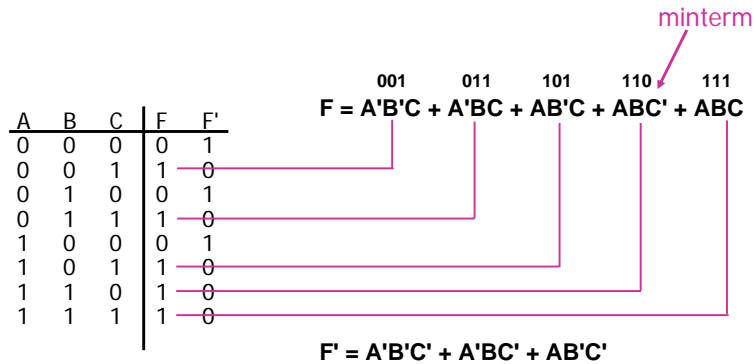
Suppose a light has to be lit by switching on simultaneously n switches. We may push any one of the n buttons at any time but we don't know if they are on or off. What is the smallest number of steps necessary to guarantee that we turn the light on starting from any initial configuration of the switches?

Canonical Forms

- Unique forms for Boolean functions
- Unique algebraic signatures:
 - Generically not the simplest
 - Can be simplified
- Two canonical forms
 - Sum of products
 - Product of sums

Sum of Products Canonical Form

- Also called disjunctive normal form (DNF)
 - Commonly called a **minterm expansion**



Minterms

- Variables appears exactly once in each minterm
 - In true or inverted form (but not both)

A	B	C	minterms
0	0	0	A'B'C' m0
0	0	1	A'B'C m1
0	1	0	A'BC' m2
0	1	1	A'BC m3
1	0	0	AB'C' m4
1	0	1	AB'C m5
1	1	0	ABC' m6
1	1	1	ABC m7

F in canonical form:

$$\begin{aligned}
 F(A,B,C) &= \Sigma m(1,3,5,6,7) \\
 &= m1 + m3 + m5 + m6 + m7 \\
 &= A'B'C + A'BC + AB'C + ABC' + ABC
 \end{aligned}$$

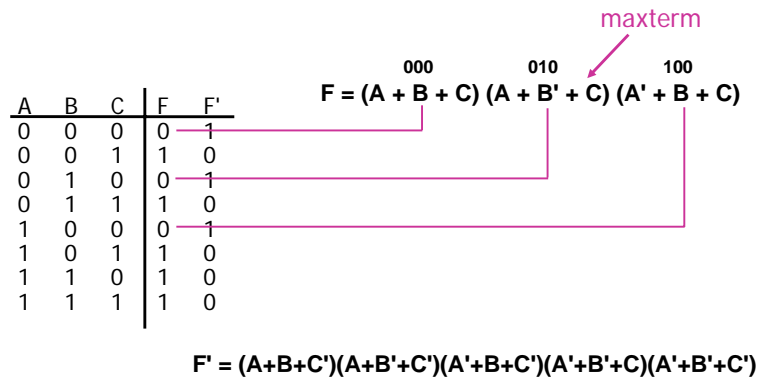
canonical form \rightarrow minimal form

$$\begin{aligned}
 F(A,B,C) &= A'B'C + A'BC + AB'C + ABC' + ABC \\
 &= (A'B' + A'B + AB' + AB)C + ABC' \\
 &= ((A' + A)(B' + B))C + ABC' \\
 &= ABC' + C \\
 &= AB + C
 \end{aligned}$$

short-hand notation

Product of Sums Canonical Form

- Also called conjunctive normal form (CNF)
 - Commonly called a **maxterm expansion**



Maxterms

- Variables appears exactly once in each maxterm
 - In true or inverted form (but not both)

A	B	C	maxterms
0	0	0	A+B+C M0
0	0	1	A+B+C' M1
0	1	0	A+B'+C M2
0	1	1	A+B'+C' M3
1	0	0	A'+B+C M4
1	0	1	A'+B+C' M5
1	1	0	A'+B'+C M6
1	1	1	A'+B'+C' M7

F in canonical form:

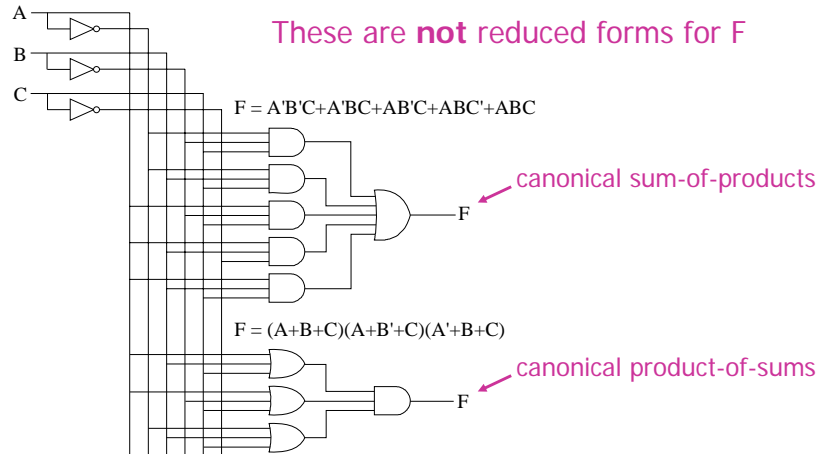
$$\begin{aligned}
 F(A,B,C) &= \Pi M(0,2,4) \\
 &= M0 \cdot M2 \cdot M4 \\
 &= (A+B+C)(A+B'+C)(A'+B+C)
 \end{aligned}$$

canonical form \rightarrow minimal form

$$\begin{aligned}
 F(A,B,C) &= (A+B+C)(A+B'+C)(A'+B+C) \\
 &= (A+B+C)(A+B'+C) \cdot \\
 &\quad (A+B+C)(A'+B+C) \\
 &= (A + C)(B + C)
 \end{aligned}$$

short-hand notation

Canonical Decompositions of $F=AB+C$



Exercise

Express the following binary function in canonical sum of products and product of sum form:

A	B	C	F
0	0	0	1
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	0

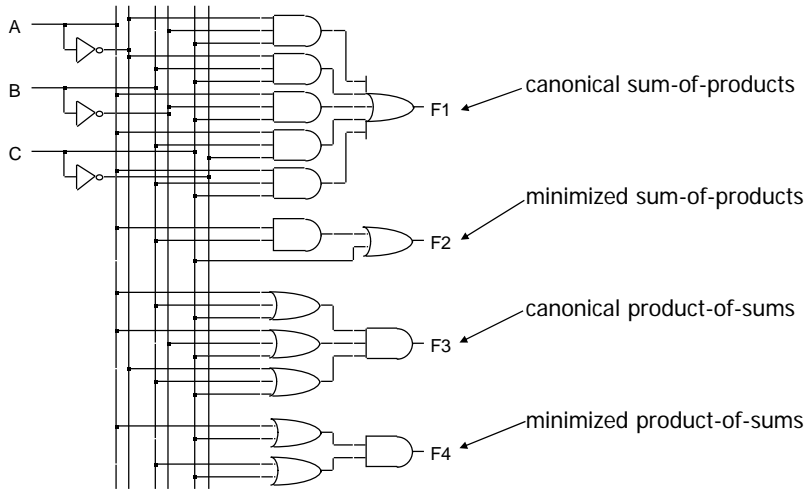
POS, SOP, and DeMorgan

- Sum-of-products
 - $F = A'B'C' + A'BC' + AB'C'$
- Apply de Morgan's to get POS
 - $(F')' = (A'B'C' + A'BC' + AB'C')$
 - $F = (A+B+C)(A+B'+C)(A'+B+C)$
- Product-of-sums
 - $F' = (A+B+C')(A+B'+C')(A'+B+C')(A'+B'+C)(A'+B'+C')$
- Apply de Morgan's to get SOP
 - $(F')' = ((A+B+C')(A+B'+C')(A'+B+C')(A'+B'+C)(A'+B'+C'))'$
 - $F = A'B'C + A'BC + AB'C + ABC' + ABC$

Conversions Between Canonical Forms

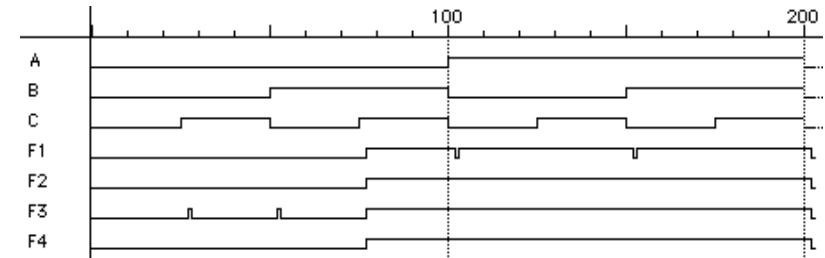
- Minterm to maxterm
 - Use maxterms that aren't in minterm expansion
 - $F(A,B,C) = \sum m(1,3,5,6,7) = \prod M(0,2,4)$
- Maxterm to minterm
 - Use minterms that aren't in maxterm expansion
 - $F(A,B,C) = \prod M(0,2,4) = \sum m(1,3,5,6,7)$
- Minterm of F to minterm of F'
 - Use minterms that don't appear
 - $F(A,B,C) = \sum m(1,3,5,6,7) \quad F'(A,B,C) = \sum m(0,2,4)$
- Maxterm of F to maxterm of F'
 - Use maxterms that don't appear
 - $F(A,B,C) = \prod M(0,2,4) \quad F'(A,B,C) = \prod M(1,3,5,6,7)$

Alternative Implementations of $F=AB+C$



Waveforms of Four Alternatives

- Waveforms are essentially identical
 - except for timing hazards (glitches)
 - delays almost identical (modeled as a delay per level, not type of gate or number of inputs to gate)



Incompletely Specified Functions

- Example: binary coded decimal increment by 1
 - BCD digits encode the decimal digits 0 – 9 in the bit patterns 0000 – 1001

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	1	1	1
0	0	1	1	1	0	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	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
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

Annotations:

- off-set of W: points to the 0 in the W column for input 0001.
- on-set of W: points to the 1 in the W column for input 0010.
- don't care (DC) set of W: points to the X's in the W column for inputs 1000-1111.
- these inputs patterns should never be encountered in practice - "don't care" about associated output values, can be exploited in minimization: points to the X's in the W, X, Y, Z columns for inputs 1000-1111.

Incompletely Specified Functions: Notation

- Don't cares and canonical forms
 - so far, only represented on-set
 - also represent don't-care-set
 - need two of the three sets (on-set, off-set, dc-set)
- Canonical representations of the BCD increment by 1 function:
 - $Z = m_0 + m_2 + m_4 + m_6 + m_8 + d_{10} + d_{11} + d_{12} + d_{13} + d_{14} + d_{15}$
 - $Z = \Sigma [m(0,2,4,6,8) + d(10,11,12,13,14,15)]$
 - $Z = M_1 \cdot M_3 \cdot M_5 \cdot M_7 \cdot M_9 \cdot D_{10} \cdot D_{11} \cdot D_{12} \cdot D_{13} \cdot D_{14} \cdot D_{15}$
 - $Z = \Pi [M(1,3,5,7,9) \cdot D(10,11,12,13,14,15)]$

Simplification of Two Level Logic

- Find a minimal sum of products or product of sums realization
 - exploit don't care information in the process
- Algebraic simplification
 - not an algorithmic/systematic procedure
 - how do you know when the minimum realization has been found?
- Computer-aided design tools
 - precise solutions require very long computation times, especially for functions with many inputs (> 10)
 - heuristic methods employed – "educated guesses" to reduce amount of computation and yield good if not best solutions
- Hand methods still relevant
 - to understand automatic tools and their strengths and weaknesses
 - ability to check results (on small examples)

Uniting Theorem

- Key tool to simplification: $A(B' + B) = A$
- Essence of simplification of two-level logic
 - find two element subsets of the ON-set where only one variable changes its value – this single varying variable can be eliminated and a single product term used to represent both elements

$$F = A'B' + AB' = (A' + A)B' = B'$$

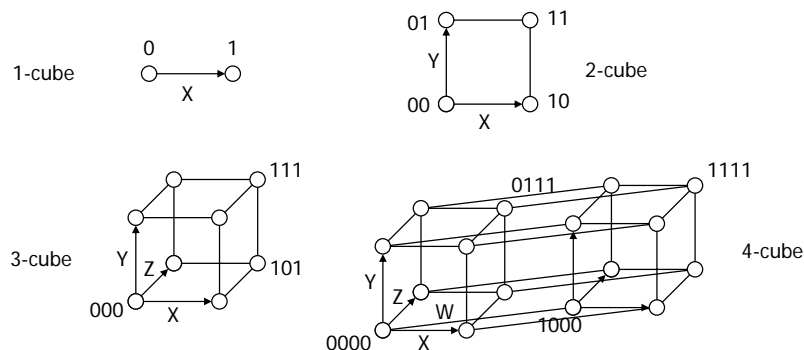
A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

B has the same value in both on-set rows
– B remains

A has a different value in the two rows
– A is eliminated

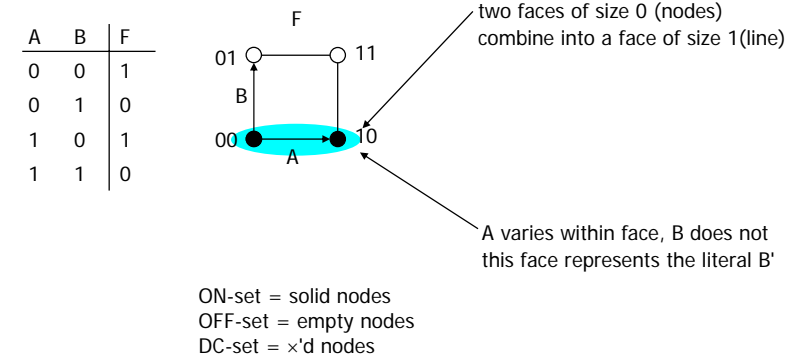
Boolean Cubes

- Visual technique for identifying when the uniting theorem can be applied
- n input variables = n-dimensional "cube"



Truth Tables To Boolean Cubes

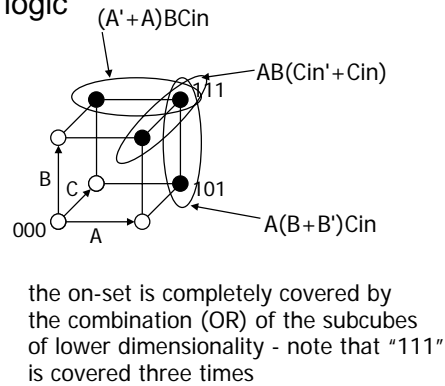
- Uniting theorem combines two "faces" of a cube into a larger "face"
- Example:



Three Variable Example

Binary full-adder carry-out logic

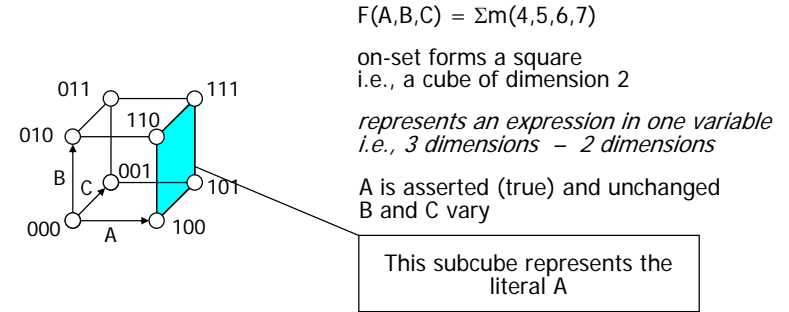
A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



$$Cout = BCin + AB + ACin$$

Three Variable Example

Sub-cubes of higher dimension than 2

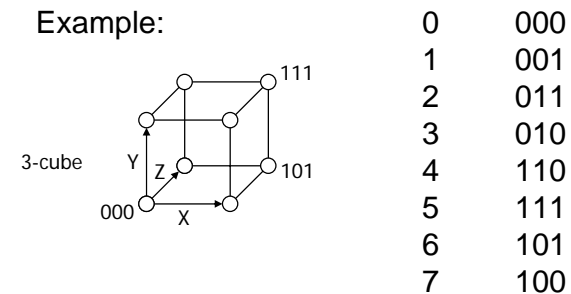
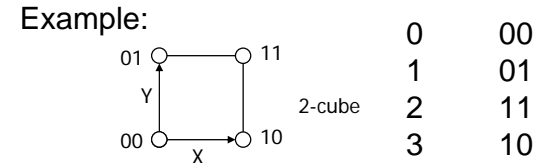


m-dimensional cubes in a n-dimensional Boolean space

- In a 3-cube (three variables):
 - a 0-cube, i.e., a single node, yields a term in 3 literals
 - a 1-cube, i.e., a line of two nodes, yields a term in 2 literals
 - a 2-cube, i.e., a plane of four nodes, yields a term in 1 literal
 - a 3-cube, i.e., a cube of eight nodes, yields a constant term "1"
- In general,
 - an m-subcube within an n-cube ($m < n$) yields a term with $n - m$ literals

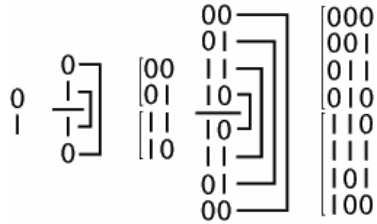
Gray Codes

- A listing of the vertices of the Boolean cube in which we only move along the edges.



Gray Codes

- Binary reflected Gray codes



Gray code on n bits => Gray code on n+1 bits

Append 0 to Gray code of n bits, followed by appending 1 to the reverse-ordered Gray code of n bits

Gray Code Uses

- Use Gray codes to help visualize higher dimensional Boolean cubes.

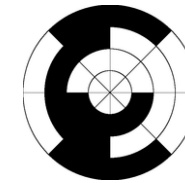
- Avoid spurious intermediate states

We want $001 \Rightarrow 110$

We get $001 \Rightarrow 000 \Rightarrow 010 \Rightarrow 110$

Gray codes help avoid synchronizing errors

- Example: measuring angles



Puzzle

Suppose a light has to be lit by switching on simultaneously n switches. We may push any one of the n buttons at any time but we don't know if they are on or off. What is the smallest number of steps necessary to guarantee that we turn the light on starting from any initial configuration of the switches?