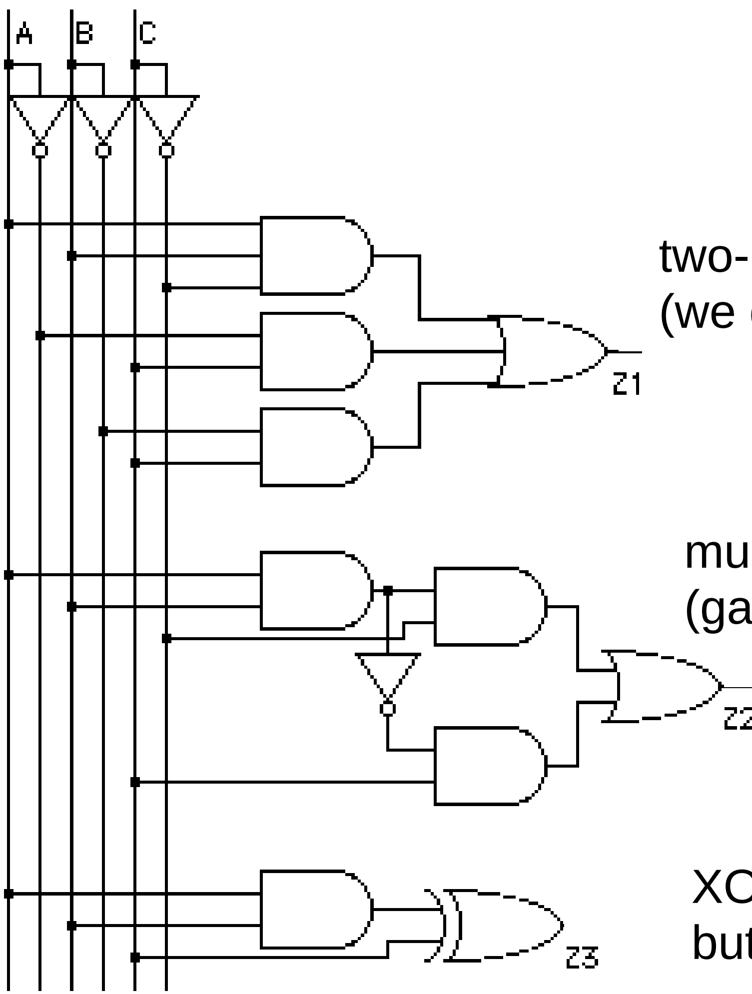
Choosing different realizations of a function

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



two-level realization
(we don't count NOT gates)

multi-level realization
(gates with fewer inputs)

XOR gate (easier to draw
but costlier to build)

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Which realization is best?

- Reduce number of inputs
  - literal: input variable (complemented or not)
    - can approximate cost of logic gate as 2 transitors per literal
    - why not count inverters?
  - fewer literals means less transistors
    - smaller circuits
  - fewer inputs implies faster gates
    - gates are smaller and thus also faster
  - fan-ins (# of gate inputs) are limited in some technologies
- Reduce number of gates
  - fewer gates (and the packages they come in) means smaller circuits
    - directly influences manufacturing costs

© Copyright 2004, Gaetano Borriello and Randy H. Katz

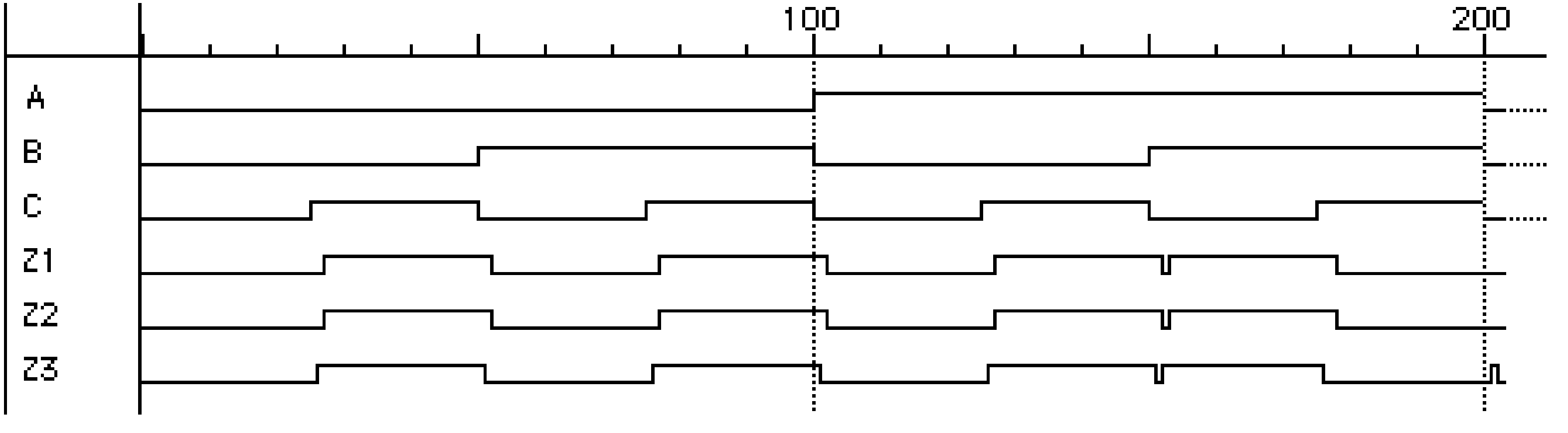
Which is the best realization? (cont'd)

- Reduce number of levels of gates
  - fewer level of gates implies reduced signal propagation delays
  - minimum delay configuration typically requires more gates
    - wider, less deep circuits
- How do we explore tradeoffs between increased circuit delay and size?
  - automated tools to generate different solutions
  - logic minimization: reduce number of gates and complexity
  - logic optimization: reduction while trading off against delay

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Are all realizations equivalent?

- Under the same input stimuli, the three alternative implementations have
  almost the same waveform behavior
  - delays are different
  - glitches (hazards) may arise – these could be bad, it depends
  - variations due to differences in number of gate levels and structure
- The three implementations are functionally equivalent



© Copyright 2004, Gaetano Borriello and Randy H. Katz

Implementing Boolean functions

- Technology independent
  - canonical forms
  - two-level forms
  - multi-level forms
- Technology choices
  - packages of a few gates
  - regular logic
  - two-level programmable logic
  - multi-level programmable logic

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Canonical forms

- Truth table is the unique signature of a Boolean function
- The same truth table can have many gate realizations
- Canonical forms
  - standard forms for a Boolean expression
  - provides a unique algebraic signature

 © Copyright 2004, Gaetano Borriello and Randy H. Katz

Sum-of-products canonical forms

- Also known as disjunctive normal form
- Also known as minterm expansion

F = *001* *011* *101* *110* *111*

F = A'B'C + A'BC + AB'C + ABC' + ABC

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

F' = A'B'C' + A'BC' + AB'C'

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Sum-of-products canonical form (cont'd)

- Product term (or minterm)
  - ANDed product of literals – input combination for which output is true
  - each variable appears exactly once, true or inverted (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

short-hand notation for minterms of 3 variables

F in canonical form:

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

canonical form ≠ minimal form

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'
= C + ABC'
= ABC' + C
= AB + C

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Product-of-sums canonical form

- Also known as conjunctive normal form
- Also known as maxterm expansion

F = *000* *010* *100*

F = (A + B + C) (A + B' + C) (A' + B + C)

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

F' = (A + B + C') (A + B' + C') (A' + B + C') (A' + B' + C) (A' + B' + C')

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Product-of-sums canonical form (cont'd)

- Sum term (or maxterm)
  - ORed sum of literals – input combination for which output is false
  - each variable appears exactly once, true or inverted (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

short-hand notation for maxterms of 3 variables

F in canonical form:

F(A, B, C) = ΠM(0,2,4)
= M0 • M2 • M4
= (A + B + C) (A + B' + C) (A' + B + C)

canonical form ≠ minimal form

F(A, B, C) = (A + B + C) (A + B' + C) (A' + B + C)
= (A + B + C) (A + B' + C)
(A + B + C) (A' + B + C)
= (A + C) (B + C)

© Copyright 2004, Gaetano Borriello and Randy H. Katz

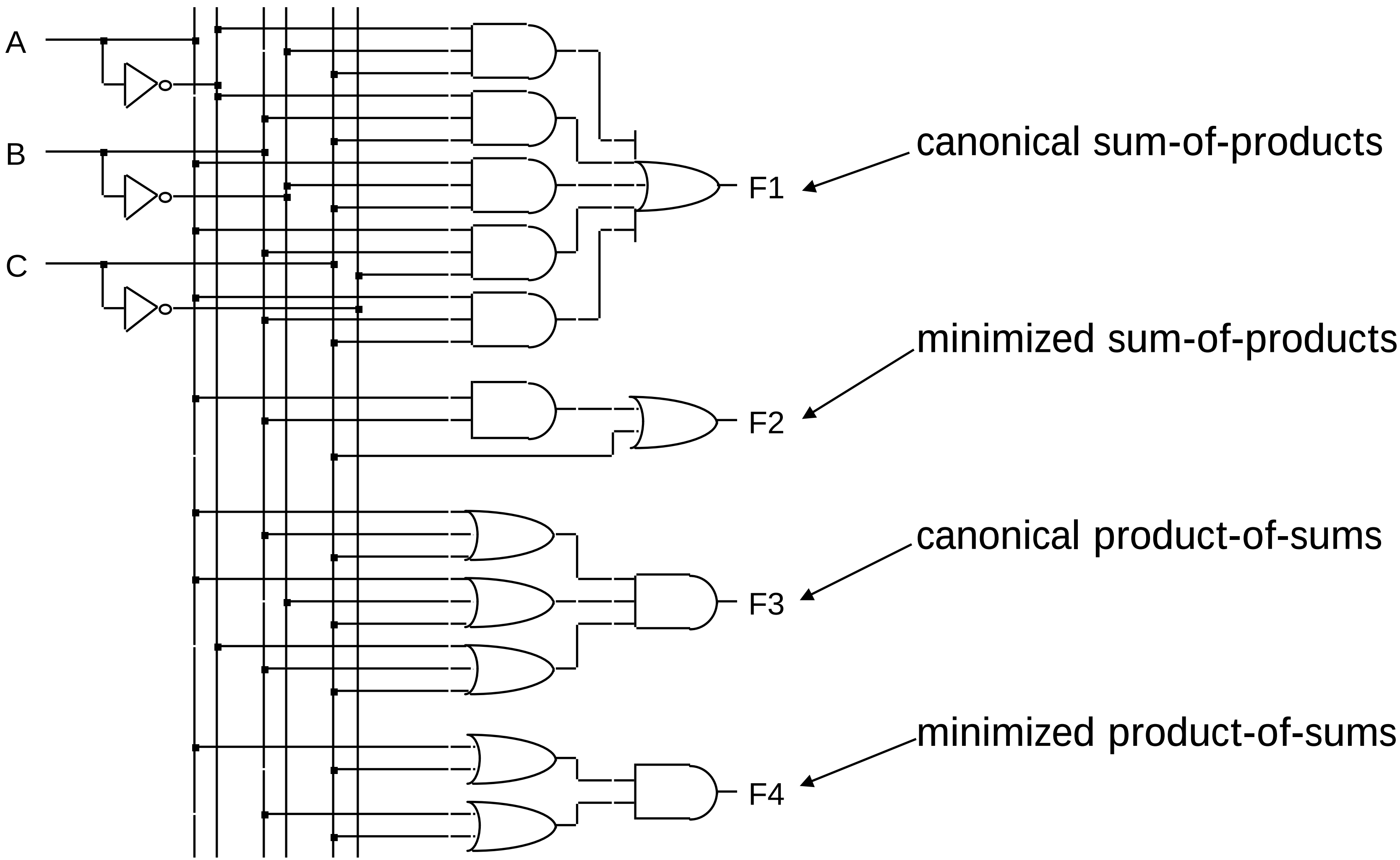
S-o-P, P-o-S, and de Morgan's theorem

- Sum-of-products
  - F' = A'B'C' + A'BC' + AB'C'
- Apply de Morgan's
  - (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
  - (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

© Copyright 2004, Gaetano Borriello and Randy H. Katz

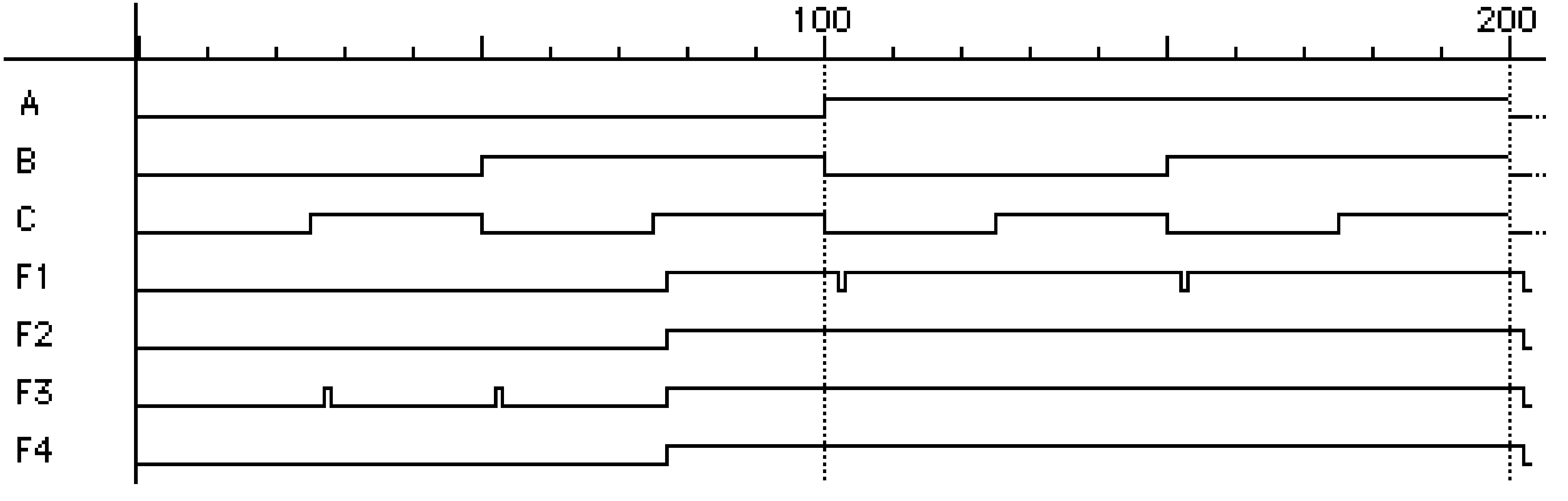
Four alternative two-level implementations of F = AB + C



 © Copyright 2004, Gaetano Borriello and Randy H. Katz

Waveforms for the 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)



© Copyright 2004, Gaetano Borriello and Randy H. Katz

Mapping between canonical forms

- Minterm to maxterm conversion
  - use maxterms whose indices do not appear in minterm expansion
  - e.g., $F(A,B,C) = \Sigma m(1,3,5,6,7) = \Pi M(0,2,4)$
- Maxterm to minterm conversion
  - use minterms whose indices do not appear in maxterm expansion
  - e.g., $F(A,B,C) = \Pi M(0,2,4) = \Sigma m(1,3,5,6,7)$
- Minterm expansion of F to minterm expansion of F'
  - use minterms whose indices do not appear
  - e.g., $F(A,B,C) = \Sigma m(1,3,5,6,7)$ $F'(A,B,C) = \Sigma m(0,2,4)$
- Maxterm expansion of F to maxterm expansion of F'
  - use maxterms whose indices do not appear
  - e.g., $F(A,B,C) = \Pi M(0,2,4)$ $F'(A,B,C) = \Pi M(1,3,5,6,7)$

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Incompleteley 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	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	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

off-set of W

on-set of W

don’t care (DC) set of W

these inputs patterns should never be encountered in practice – **"don’t care"** about associated output values, can be exploited in minimization

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Notation for incompletely specified functions

- 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:
  - $Z = m0 + m2 + m4 + m6 + m8 + d10 + d11 + d12 + d13 + d14 + d15$
  - $Z = \Sigma\ [\ m(0,2,4,6,8) + d(10,11,12,13,14,15)\ ]$
  - $Z = M1 \bullet M3 \bullet M5 \bullet M7 \bullet M9 \bullet D10 \bullet D11 \bullet D12 \bullet D13 \bullet D14 \bullet D15$
  - $Z = \Pi\ [\ M(1,3,5,7,9) \bullet D(10,11,12,13,14,15)\ ]$

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Simplification of two-level combinational logic

- Finding 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)

© Copyright 2004, Gaetano Borriello and Randy H. Katz

The 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

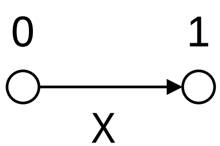
A has a different value in the two rows – A is eliminated

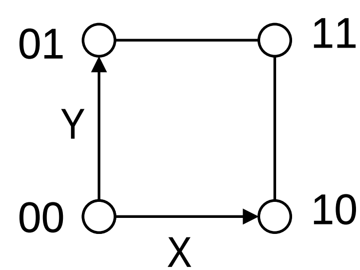
© Copyright 2004, Gaetano Borriello and Randy H. Katz

Boolean cubes

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

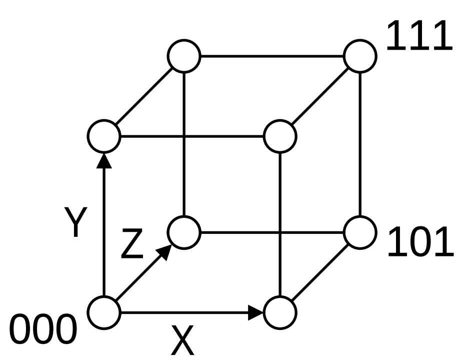
1-cube

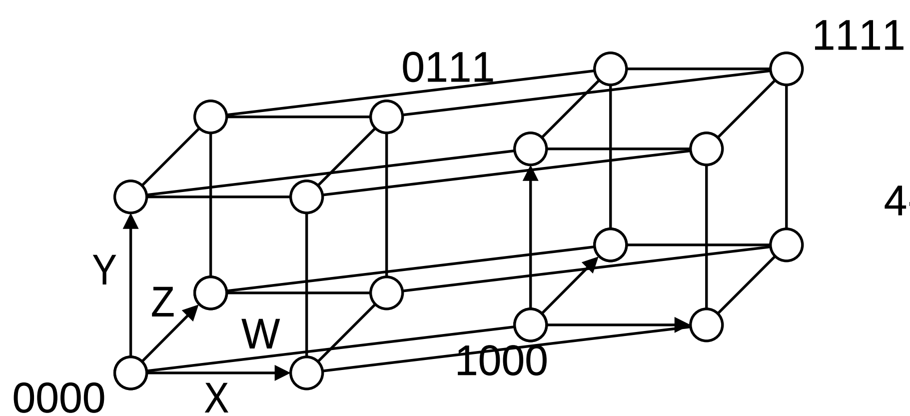




2-cube

3-cube





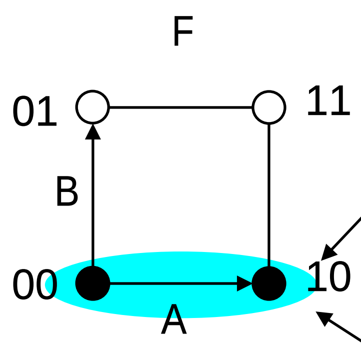
4-cube

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Mapping truth tables onto Boolean cubes

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

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



two faces of size 0 (nodes) combine into a face of size 1(line)

A varies within face, B does not this face represents the literal B'

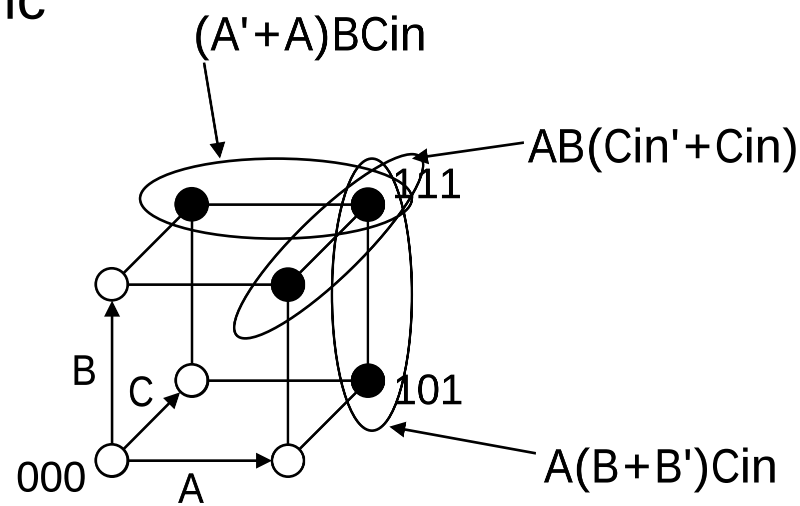
ON-set = solid nodes
OFF-set = empty nodes
DC-set = $\times$'d nodes

© Copyright 2004, Gaetano Borriello and Randy H. Katz

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



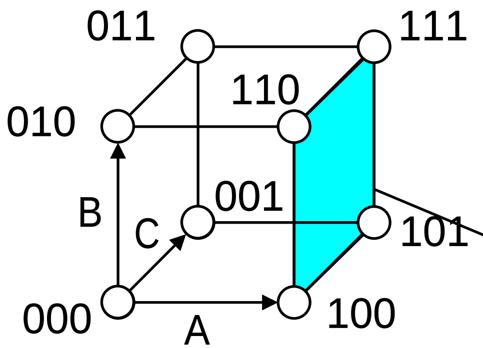
the on-set is completely covered by the combination (OR) of the subcubes of lower dimensionality - note that “111” is covered three times

Cout = BCin+AB+ACin

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Higher dimensional cubes

- Sub-cubes of higher dimension than 2



F(A,B,C) = Σm(4,5,6,7)

on-set forms a square
i.e., a cube of dimension 2

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

A is asserted (true) and unchanged
B and C vary

This subcube represents the literal A

© Copyright 2004, Gaetano Borriello and Randy H. Katz

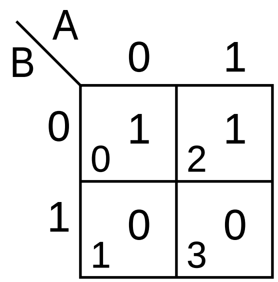
m-dimensional cubes in a n-dimensional Boolean space

- 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

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Karnaugh maps

- Flat map of Boolean cube
  - wrap–around at edges
  - hard to draw and visualize for more than 4 dimensions
  - virtually impossible for more than 6 dimensions
- Alternative to truth-tables to help visualize adjacencies
  - guide to applying the uniting theorem
  - on-set elements with only one variable changing value are adjacent unlike the situation in a linear truth-table

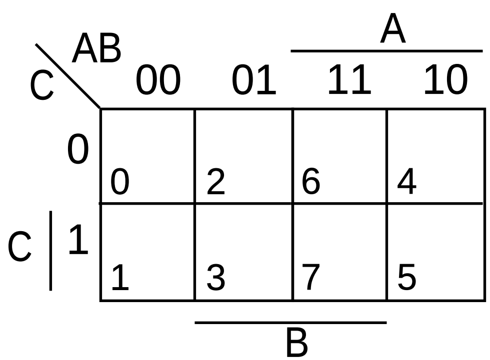


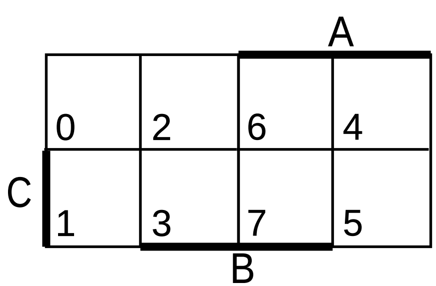
A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

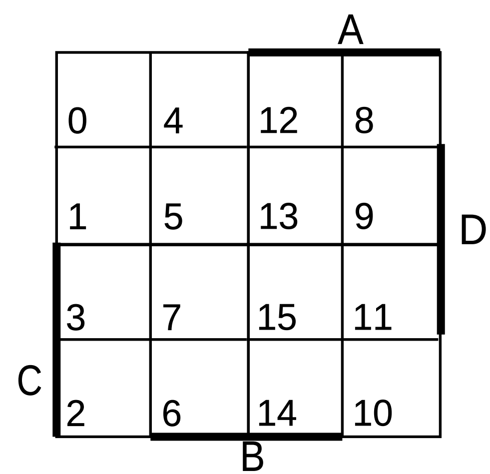
© Copyright 2004, Gaetano Borriello and Randy H. Katz

Karnaugh maps (cont'd)

- Numbering scheme based on Gray–code
  - e.g., 00, 01, 11, 10
  - only a single bit changes in code for adjacent map cells





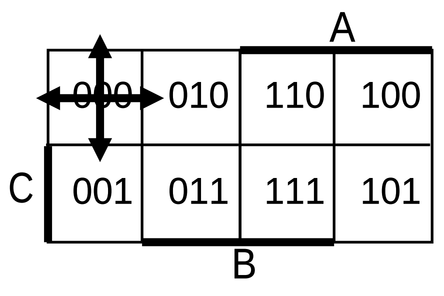


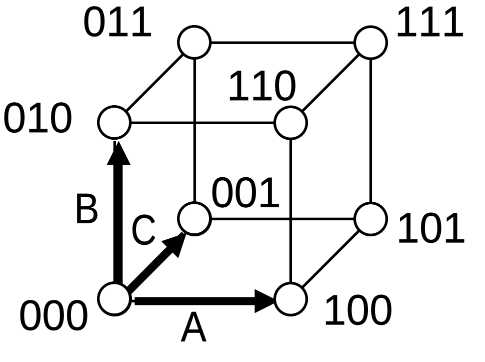
13 = 1101= ABC'D

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Adjacencies in Karnaugh maps

- Wrap from first to last column
- Wrap top row to bottom row

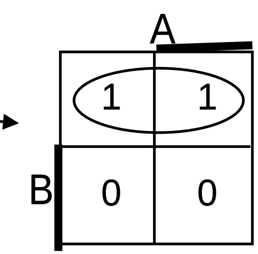




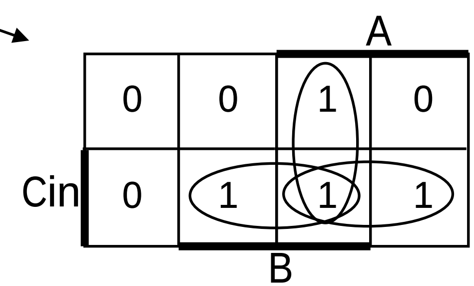
© Copyright 2004, Gaetano Borriello and Randy H. Katz

Karnaugh map examples

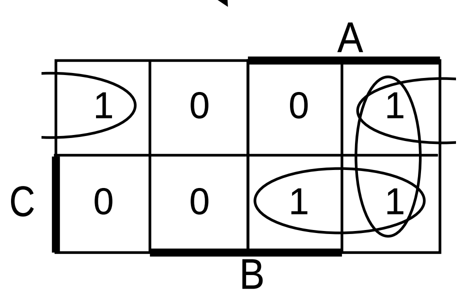
- F =
- Cout =
- f(A,B,C) = Σm(0,4,5,7)



B'



AB + ACin + BCin

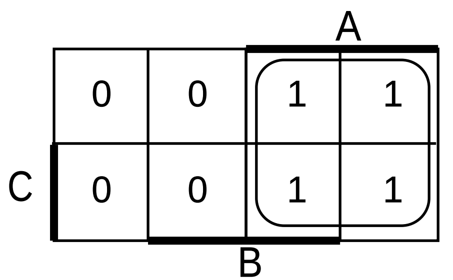


AC + B'C' + $\cancel{AB'}$

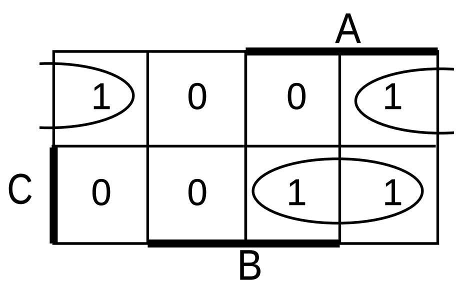
obtain the complement of the function by covering 0s with subcubes

© Copyright 2004, Gaetano Borriello and Randy H. Katz

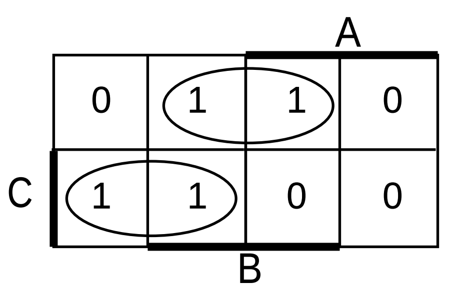
More Karnaugh map examples



G(A,B,C) = A



F(A,B,C) = Σm(0,4,5,7) = AC + B'C'



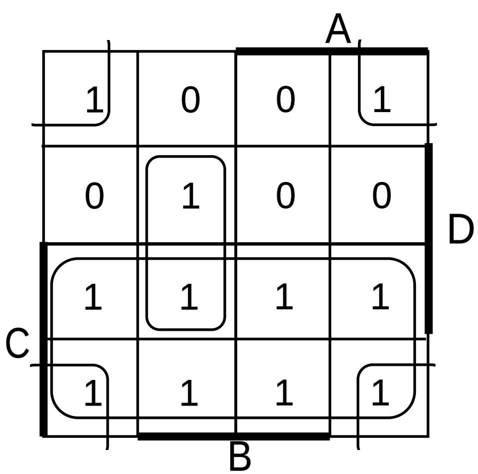
F' simply replace 1's with 0's and vice versa
F'(A,B,C) = Σ m(1,2,3,6)= BC' + A'C

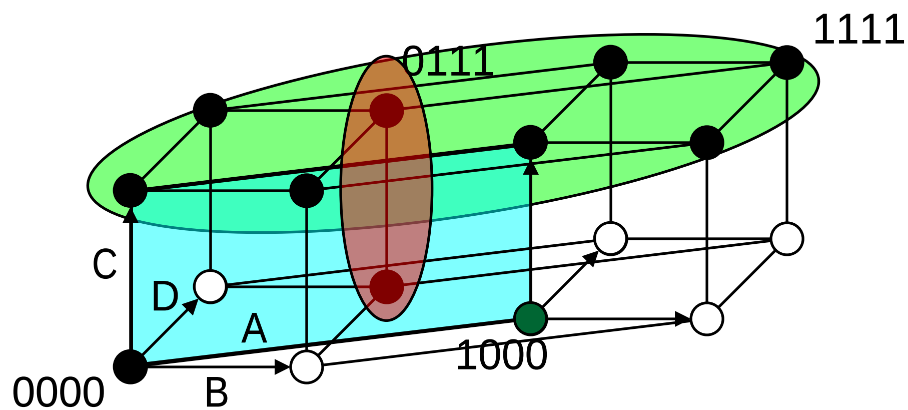
 © Copyright 2004, Gaetano Borriello and Randy H. Katz

Karnaugh map: 4-variable example

- F(A,B,C,D) = Σm(0,2,3,5,6,7,8,10,11,14,15)

F = C + A'BD + B'D'



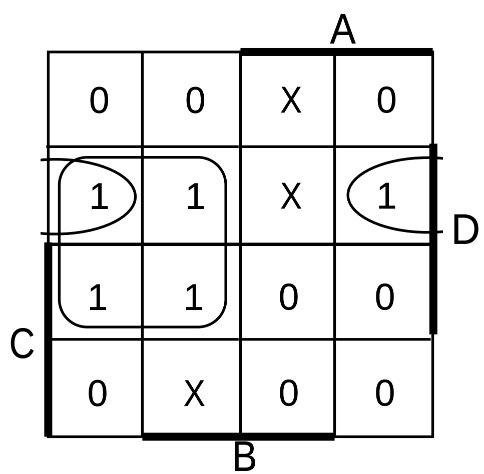


find the smallest number of the largest possible subcubes to cover the ON-set
(fewer terms with fewer inputs per term)

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Karnaugh maps: don't cares

- $f(A,B,C,D) = \Sigma\ m(1,3,5,7,9) + d(6,12,13)$
  - without don't cares
    - f = A'D + B'C'D



 © Copyright 2004, Gaetano Borriello and Randy H. Katz

Karnaugh maps: don’t cares (cont’d)

- f(A,B,C,D) = Σ m(1,3,5,7,9) + d(6,12,13)
  - f = A'D + B'C'D — without don't cares
  - f = A'D + C'D — with don't cares

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

by using don't care as a "1"
a 2-cube can be formed
rather than a 1-cube to cover
this node

<u>don't cares</u> can be treated as
1s or 0s
depending on which is more
advantageous

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Activity

- Minimize the function $F = \Sigma\, m(0, 2, 7, 8, 14, 15) + d(3, 6, 9, 12, 13)$

© Copyright 2004, Gaetano Borriello and Randy H. Katz

Combinational logic summary

- Logic functions, truth tables, and switches
  - NOT, AND, OR, NAND, NOR, XOR, . . ., minimal set
- Axioms and theorems of Boolean algebra
  - proofs by re-writing and perfect induction
- Gate logic
  - networks of Boolean functions and their time behavior
- Canonical forms
  - two-level and incompletely specified functions
- Simplification
  - a start at understanding two-level simplification
- Later
  - automation of simplification
  - multi-level logic
  - time behavior
  - hardware description languages
  - design case studies

© Copyright 2004, Gaetano Borriello and Randy H. Katz