
Digital Logic Design

ECEN 3233

Module 2: Combinational Logic

Louis G. Johnson
School of Electrical and Computer Engineering
Oklahoma State University

Fall 2007

Where do we begin?

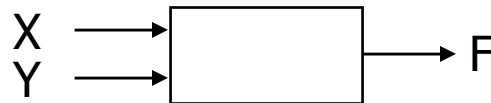
- How can we deal with logical expressions, and then put them in a form suitable for implementation in hardware?
 - *I will go to the store if I am out of eggs and I am out of milk*

Combinational logic

- Basic logic
 - Boolean algebra, proofs by re-writing, proofs by perfect induction
 - logic functions, truth tables, and switches
 - NOT, AND, OR, NAND, NOR, XOR, . . . , minimal set
- Logic realization
 - two-level logic and canonical forms
 - incompletely specified functions
- Simplification
 - uniting theorem
 - grouping of terms in Boolean functions
- Alternate representations of Boolean functions
 - cubes
 - Karnaugh maps

Possible logic functions of two variables

- There are 16 possible functions of 2 input variables:
 - in general, there are 2^{2^n} functions of n inputs



- Which ones are useful?

X	Y	16 possible functions (F0–F15)																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1

0
 X and Y
 X
 Y
 X xor Y
 X or Y
 X nor Y
 not (X or Y)
 X = Y
 not Y
 not X
 X nand Y
 not (X and Y)

An algebraic structure

- An algebraic structure consists of
 - a set of elements B
 - binary operations $\{ + , \cdot \}$
 - and a unary operation $\{ ' \}$
 - such that the following axioms hold:

1. the set B contains at least two elements: a, b

2. closure: $a + b$ is in B

$a \cdot b$ is in B

3. commutativity: $a + b = b + a$

$a \cdot b = b \cdot a$

4. associativity: $a + (b + c) = (a + b) + c$

$a \cdot (b \cdot c) = (a \cdot b) \cdot c$

5. identity: $a + 0 = a$

$a \cdot 1 = a$

6. distributivity: $a + (b \cdot c) = (a + b) \cdot (a + c)$

$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

7. complementarity: $a + a' = 1$

$a \cdot a' = 0$

Boolean algebra

- Boolean algebra
 - $B = \{0, 1\}$
 - variables
 - $+$ is logical OR, \cdot is logical AND
 - $'$ is logical NOT
- All algebraic axioms hold

Logic functions and Boolean algebra

- Any logic function that can be expressed as a truth table can be written as an expression in Boolean algebra using the operators: ', +, and •

X	Y	X • Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X'	X' • Y
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

X	Y	X'	Y'	X • Y	X' • Y'	(X • Y) + (X' • Y')
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

$$(X \bullet Y) + (X' \bullet Y') \equiv X = Y$$

Boolean expression that is true when the variables X and Y have the same value and false, otherwise

X, Y are Boolean algebra variables

Axioms and theorems of Boolean algebra

- identity

1. $X + 0 = X$

1D. $X \cdot 1 = X$

- null

2. $X + 1 = 1$

2D. $X \cdot 0 = 0$

- idempotency:

3. $X + X = X$

3D. $X \cdot X = X$

- involution:

4. $(X')' = X$

- complementarity:

5. $X + X' = 1$

5D. $X \cdot X' = 0$

- commutativity:

6. $X + Y = Y + X$

6D. $X \cdot Y = Y \cdot X$

- associativity:

7. $(X + Y) + Z = X + (Y + Z)$

7D. $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$

Axioms and theorems of Boolean algebra (cont'd)

- distributivity:

$$8. X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$$

$$8D. X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$$

- uniting:

$$9. X \cdot Y + X \cdot Y' = X$$

$$9D. (X + Y) \cdot (X + Y') = X$$

- absorption:

$$10. X + X \cdot Y = X$$

$$10D. X \cdot (X + Y) = X$$

$$11. (X + Y') \cdot Y = X \cdot Y$$

$$11D. (X \cdot Y') + Y = X + Y$$

- factoring:

$$12. (X + Y) \cdot (X' + Z) = \\ X \cdot Z + X' \cdot Y$$

$$12D. X \cdot Y + X' \cdot Z = \\ (X + Z) \cdot (X' + Y)$$

- consensus:

$$13. (X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) = \\ X \cdot Y + X' \cdot Z$$

$$13D. (X + Y) \cdot (Y + Z) \cdot (X' + Z) = \\ (X + Y) \cdot (X' + Z)$$

Axioms and theorems of Boolean algebra (cont'd)

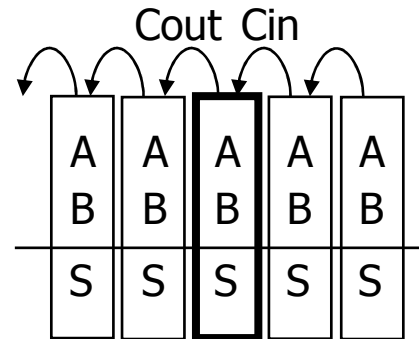
- de Morgan's:
14. $(X + Y + \dots)' = X' \cdot Y' \cdot \dots$ 14D. $(X \cdot Y \cdot \dots)' = X' + Y' + \dots$
- generalized de Morgan's:
15. $f'(X_1, X_2, \dots, X_n, 0, 1, +, \cdot) = f(X_1', X_2', \dots, X_n', 1, 0, \cdot, +)$
- establishes relationship between \cdot and $+$

Axioms and theorems of Boolean algebra (cont'd)

- Duality
 - a dual of a Boolean expression is derived by replacing
 - by +, + by •, 0 by 1, and 1 by 0, and leaving variables unchanged
 - any theorem that can be proven is thus also proven for its dual!
 - a meta-theorem (a theorem about theorems)
- duality:
 - 16. $X + Y + \dots \Leftrightarrow X \cdot Y \cdot \dots$
- generalized duality:
 - 17. $f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot) \Leftrightarrow f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$
- Different than deMorgan's Law
 - this is a statement about theorems
 - this is not a way to manipulate (re-write) expressions

A simple example: 1-bit binary full adder

- Inputs: A, B, Carry-in
- Outputs: Sum, Carry-out



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



$$S = A' B' Cin + A' B Cin' + A B' Cin' + A B Cin$$

$$Cout = A' B Cin + A B' Cin + A B Cin' + A B Cin$$

Apply the theorems to simplify expressions

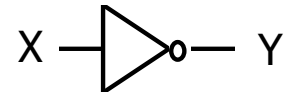
- The theorems of Boolean algebra can simplify Boolean expressions
 - e.g., full adder's carry-out function (same rules apply to any function)

$$\begin{aligned} \text{Cout} &= A' B \text{Cin} + A B' \text{Cin} + A B \text{Cin}' + A B \text{Cin} \\ &= A' B \text{Cin} + A B' \text{Cin} + A B \text{Cin}' + \boxed{A B \text{Cin} + A B \text{Cin}} \\ &= A' B \text{Cin} + A B \text{Cin} + A B' \text{Cin} + A B \text{Cin}' + A B \text{Cin} \\ &= (A' + A) B \text{Cin} + A B' \text{Cin} + A B \text{Cin}' + A B \text{Cin} \\ &= (1) B \text{Cin} + A B' \text{Cin} + A B \text{Cin}' + A B \text{Cin} \\ &= B \text{Cin} + A B' \text{Cin} + A B \text{Cin}' + \boxed{A B \text{Cin} + A B \text{Cin}} \\ &= B \text{Cin} + A B' \text{Cin} + A B \text{Cin} + A B \text{Cin}' + A B \text{Cin} \\ &= B \text{Cin} + A (B' + B) \text{Cin} + A B \text{Cin}' + A B \text{Cin} \\ &= B \text{Cin} + A (1) \text{Cin} + A B \text{Cin}' + A B \text{Cin} \\ &= B \text{Cin} + A \text{Cin} + A B (\text{Cin}' + \text{Cin}) \\ &= B \text{Cin} + A \text{Cin} + A B (1) \\ &= B \text{Cin} + A \text{Cin} + A B \end{aligned}$$

adding extra terms
creates new factoring
opportunities

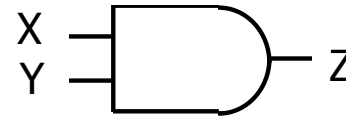
From Boolean expressions to logic gates

■ NOT: X' \bar{X} $\sim X$



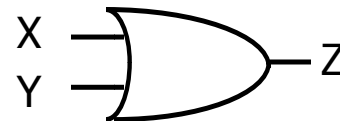
X	Y
0	1
1	0

■ AND: $X \cdot Y$ XY $X \wedge Y$



X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

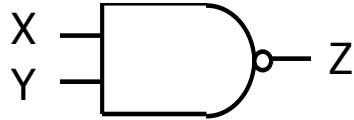
■ OR: $X + Y$ $X \vee Y$



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

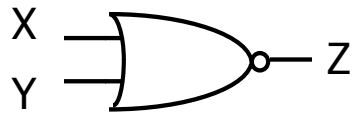
From Boolean expressions to logic gates (cont'd)

- NAND



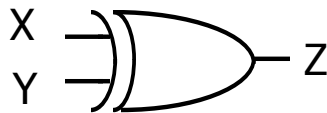
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

- NOR



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

- XOR
 $X \oplus Y$



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

$X \text{ xor } Y = X Y' + X' Y$
 X or Y but not both
 ("inequality", "difference")

- XNOR
 $X = Y$



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

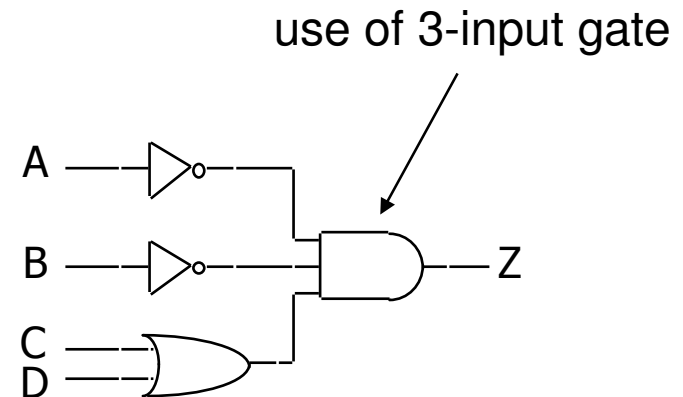
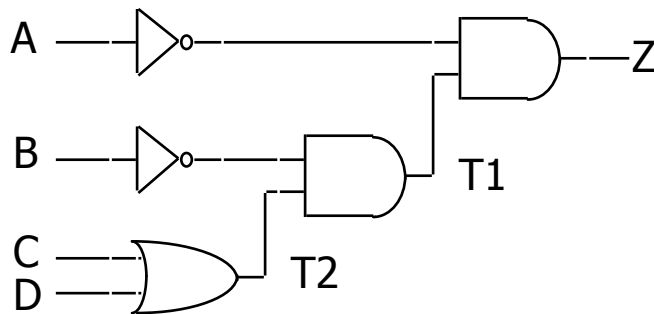
$X \text{ xnor } Y = X Y + X' Y'$
 X and Y are the same
 ("equality", "coincidence")

From Boolean expressions to logic gates (cont'd)

- More than one way to map expressions to gates

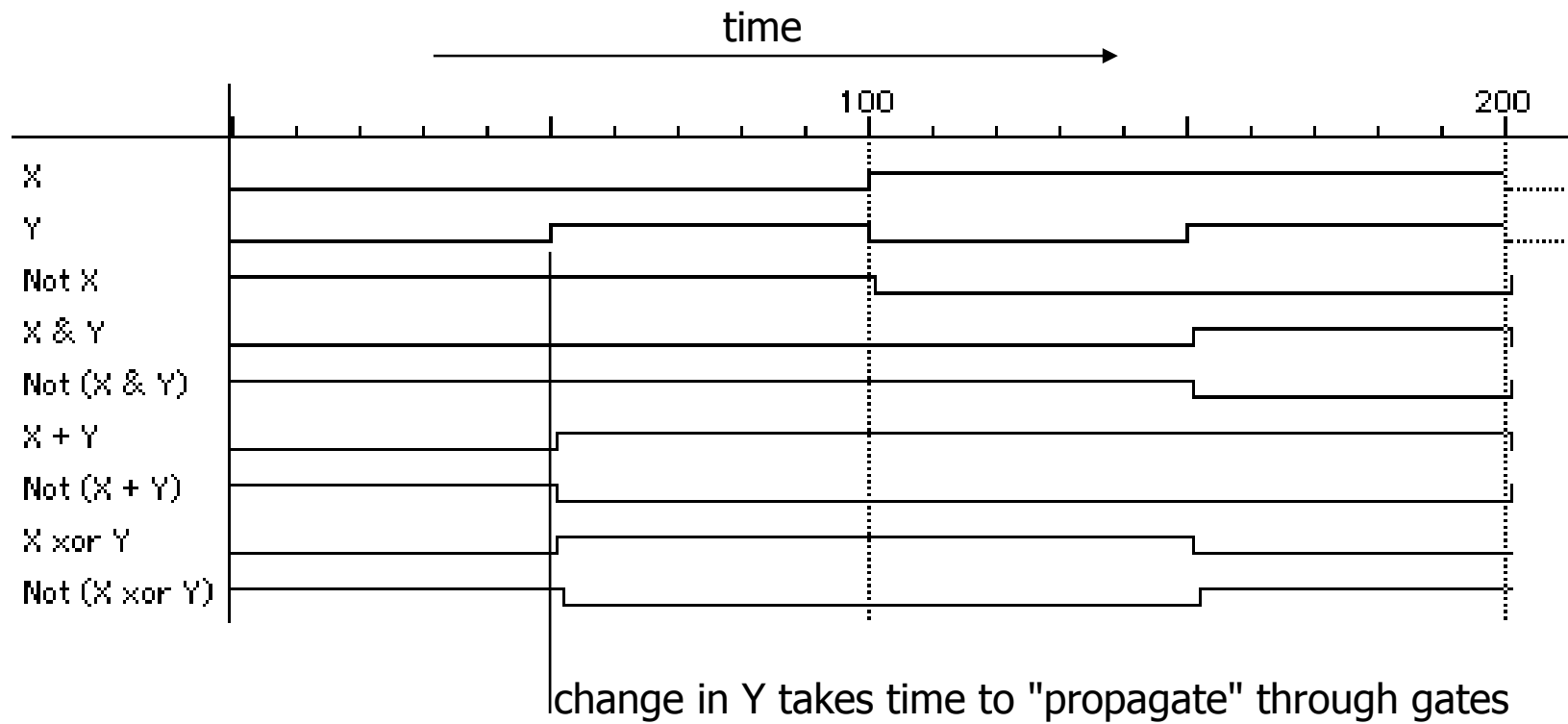
- e.g., $Z = A' \cdot B' \cdot (C + D) = (A' \cdot (B' \cdot (C + D)))$

$$\frac{\overline{T2}}{T1}$$



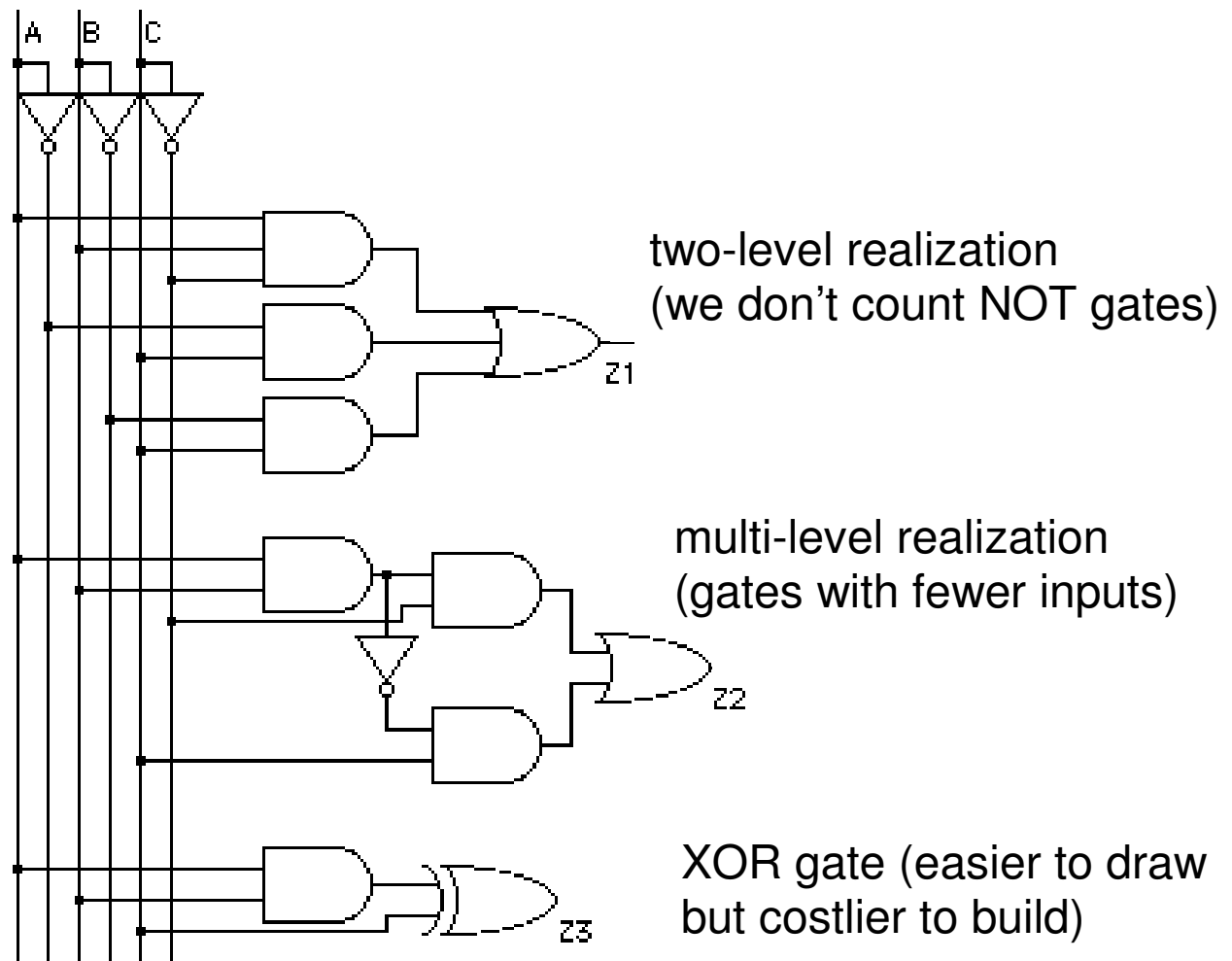
Waveform view of logic functions

- Just a sideways truth table
 - but note how edges don't line up exactly
 - it takes time for a gate to switch its output! (nothing happens instantaneously in real life)



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



Which realization is best?

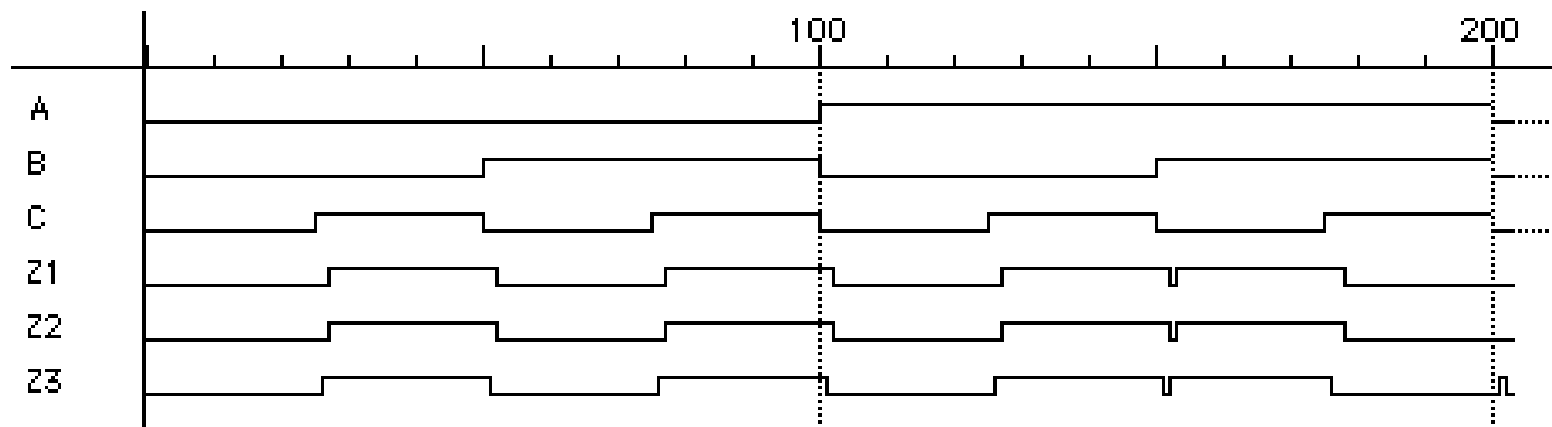
- Reduce number of inputs
 - literal: input variable (complemented or not)
 - can approximate cost of logic gate as 2 transistors per literal
 - why not count inverters?
 - fewer literals means fewer transistors
 - smaller circuits
 - fewer inputs implies faster gates
 - gates are smaller and thus also 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

Which is the best realization? (cont'd)

- Reduce number of levels of gates
 - fewer levels 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

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



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

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

Sum-of-products canonical forms

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


					$F = 001 \quad 011 \quad 101 \quad 110 \quad 111$				
					$F = A'B'C + A'BC + AB'C + ABC' + ABC$				
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					
					$F' = A'B'C' + A'BC' + AB'C'$				

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:

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

canonical form \neq minimal form

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

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			

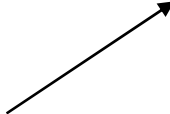
$$F' = (A + B + C') (A + B' + C') (A' + B + C') (A' + B' + C) (A' + B' + C')$$

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:

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

canonical form \neq minimal form

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

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

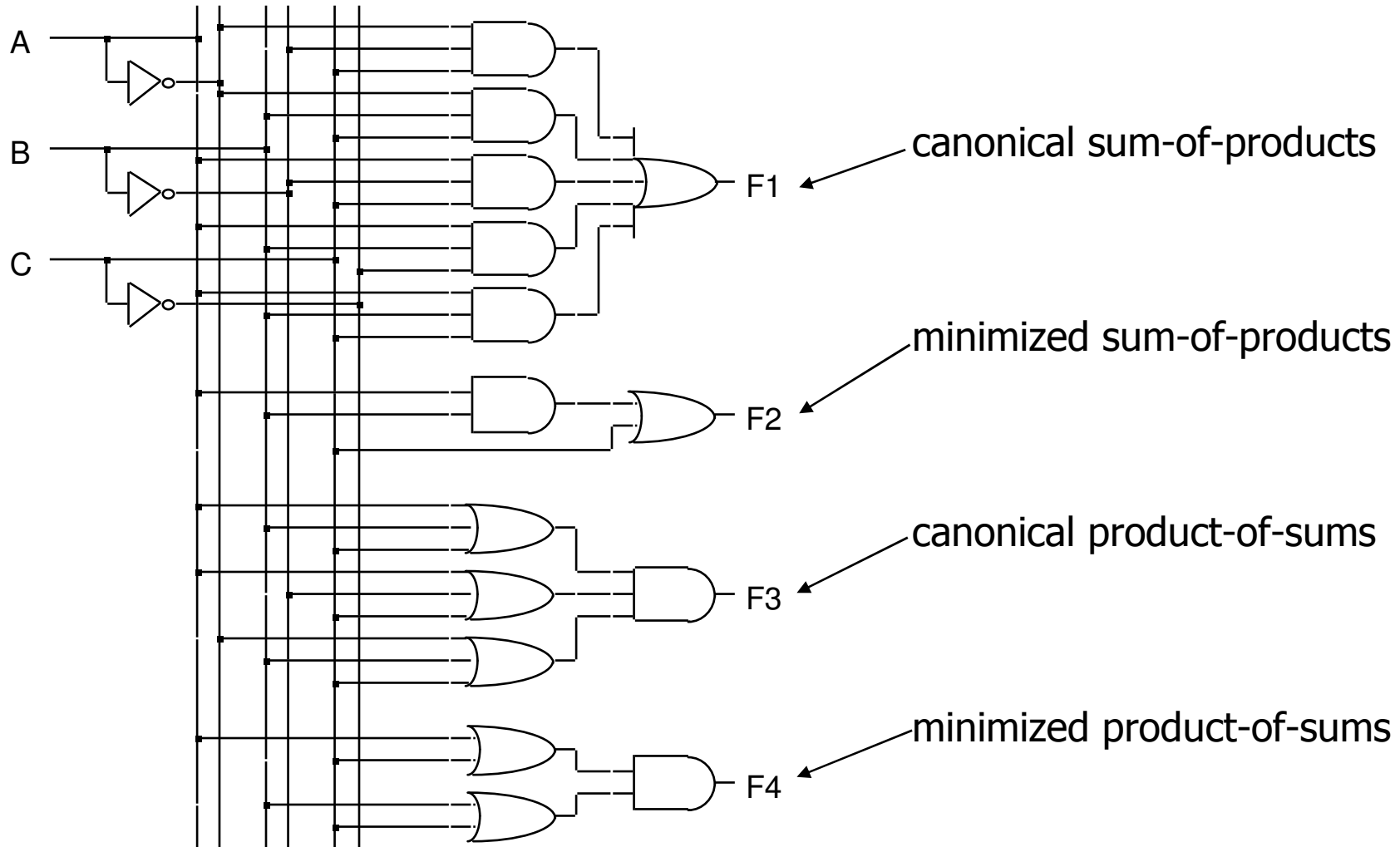
- $F' = (A + B + C') (A + B' + C') (A' + B + C') (A' + B' + C) (A' + B' + C')$

- Apply de Morgan's

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

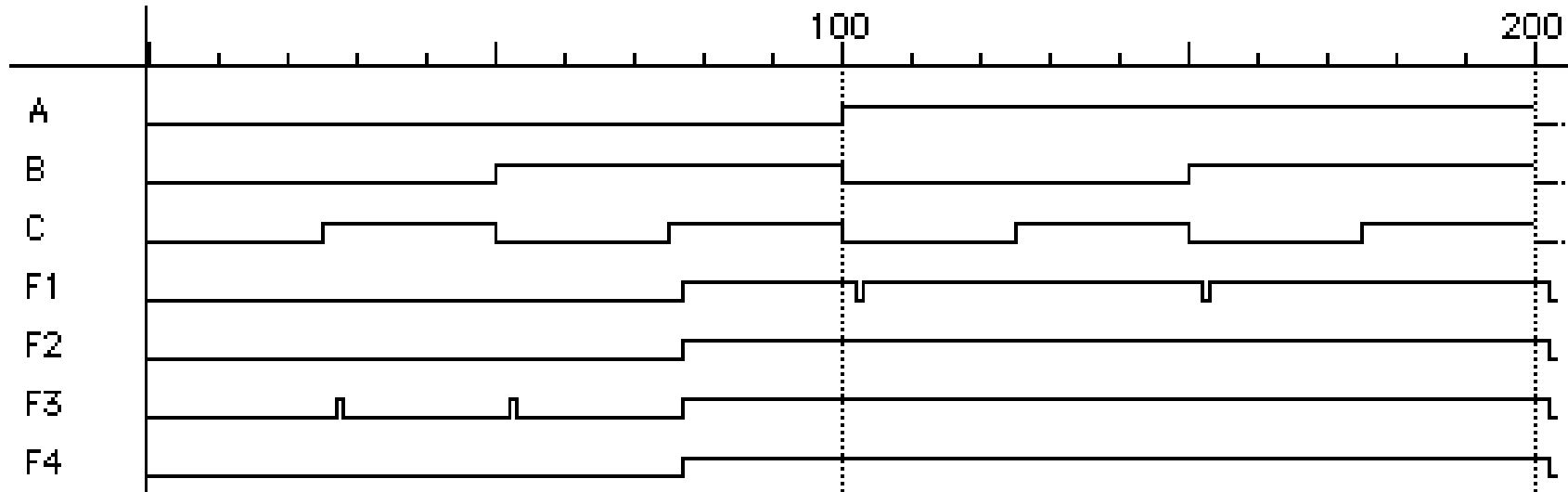
- $F = A'B'C + A'BC + AB'C + ABC' + ABC$

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



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)



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., $F(A,B,C) = \Pi M(0,2,4)$ $F'(A,B,C) = \Pi M(1,3,5,6,7)$

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	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

Notation for incompletely specified functions

- Don't cares and canonical forms
 - so far, only represented on-set
 - can 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:
 - $Z = m_0 + m_2 + m_4 + m_6 + m_8 + d_{10} + d_{11} + d_{12} + d_{13} + d_{14} + d_{15}$
 - $Z = \Sigma [m(0,2,4,6,8) + d(10,11,12,13,14,15)]$
 - $Z = M_1 \cdot M_3 \cdot M_5 \cdot M_7 \cdot M_9 \cdot D_{10} \cdot D_{11} \cdot D_{12} \cdot D_{13} \cdot D_{14} \cdot D_{15}$
 - $Z = \Pi [M(1,3,5,7,9) \cdot D(10,11,12,13,14,15)]$

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