

EECS150

Section 1

Introduction to Combinational Logic

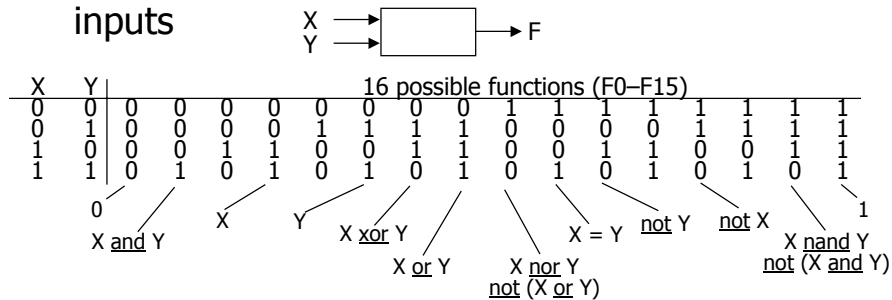
Fall 2001



2-Variable Logic Functions

- There are 16 possible functions of 2 input variables:

- in general, there are $2^{(2^n)}$ functions of n inputs



Boolean algebra

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

An algebraic structure

- An algebraic structure consists of
 - a set of elements B
 - binary operations $\{ +, \bullet \}$
 - and a unary operation $\{ ' \}$
 - such that the following axioms hold:
 1. set B contains at least two elements, a, b , such that $a \neq b$
 2. closure: $a + b$ is in B $a \bullet b$ is in B
 3. commutativity: $a + b = b + a$ $a \bullet b = b \bullet a$
 4. associativity: $a + (b + c) = (a + b) + c$ $a \bullet (b \bullet c) = (a \bullet b) \bullet c$
 5. identity: $a + 0 = a$ $a \bullet 1 = a$
 6. distributivity: $a + (b \bullet c) = (a + b) \bullet (a + c)$ $a \bullet (b + c) = (a \bullet b) + (a \bullet c)$
 7. complementarity: $a + a' = 1$ $a \bullet a' = 0$

Axioms and Theorems

- 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

- 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. $X \cdot Y + X' \cdot Z = (X + Z) \cdot (X' + Y)$
- Concensus:
 - 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

- 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 $+$

A	B	F	\bar{A}	\bar{B}	$\bar{A} + \bar{B}$	\bar{F}	$\overline{\bar{A} + \bar{B}}$
0	0	0					
0	1	0					
1	0	0					
1	1	1					

Axioms and theorems

- Duality
 - Dual of a Boolean expression is derived by replacing \cdot by $+$, $+$ by \cdot , 0 by 1, and 1 by 0, and leaving variables unchanged
 - Any theorem that can be proven is thus also proven for its dual!
 - 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

Proving theorems (rewriting)

- Using the axioms of Boolean algebra:

- e.g., prove the theorem: $X \cdot Y + X \cdot Y' = X$

distributivity (8)	$X \cdot Y + X \cdot Y'$	=	$X \cdot (Y + Y')$
complementarity (5)	$X \cdot (Y + Y')$	=	$X \cdot (1)$
identity (1D)	$X \cdot (1)$	=	$X \checkmark$

- e.g., prove the theorem: $X + X \cdot Y = X$

identity (1D)	$X + X \cdot Y$	=	$X \cdot 1 + X \cdot Y$
distributivity (8)	$X \cdot 1 + X \cdot Y$	=	$X \cdot (1 + Y)$
identity (2)	$X \cdot (1 + Y)$	=	$X \cdot (1)$
identity (1D)	$X \cdot (1)$	=	$X \checkmark$

Proving theorems (perfect induction)

- Using perfect induction (complete truth table):

- e.g., de Morgan's:

$(X + Y)' = X' \cdot Y'$	X	Y	X'	Y'	$(X + Y)'$	$X' \cdot Y'$
NOR is equivalent to AND	0	0	1	1	1	1
with inputs complemented	0	1	1	0	0	0
	1	0	0	1	0	0
	1	1	0	0	0	0

$(X \cdot Y)' = X' + Y'$	X	Y	X'	Y'	$(X \cdot Y)'$	$X' + Y'$
NAND is equivalent to OR	0	0	1	1	1	1
with inputs complemented	0	1	1	0	1	1
	1	0	0	1	1	1
	1	1	0	0	0	0

Logic Functions

- 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	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X'	$X' \cdot Y$
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

X	Y	X'	Y'	$X \cdot Y$	$X' \cdot Y'$	$(X \cdot Y) + (X' \cdot Y')$
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

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

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

X, Y are Boolean algebra variables

Cost of different logic functions

- Different functions are easier or harder to implement
 - Each has a cost associated with the number of switches needed
 - 0 (F0) and 1 (F15): require 0 switches, directly connect output to low/high
 - X (F3) and Y (F5): require 0 switches, output is one of inputs
 - X' (F12) and Y' (F10): require 2 switches for "inverter" or NOT-gate
 - X nor Y (F4) and X nand Y (F14): require 4 switches
 - X or Y (F7) and X and Y (F1): require 6 switches
 - $X = Y$ (F9) and $X \oplus Y$ (F6): require 16 switches
- Because NOT, NOR, and NAND are the cheapest they are the functions we implement the most in practice

A simple example

- 1-bit binary adder
 - inputs: A, B, Carry-in
 - outputs: Sum, Carry-out



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

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

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

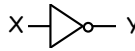
Apply theorems to simplify

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

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

From 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 expressions to logic gates

- NAND $X \cdot Y$



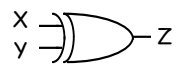
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

- NOR $X + Y$



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

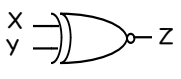
- XOR $X \oplus Y$



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

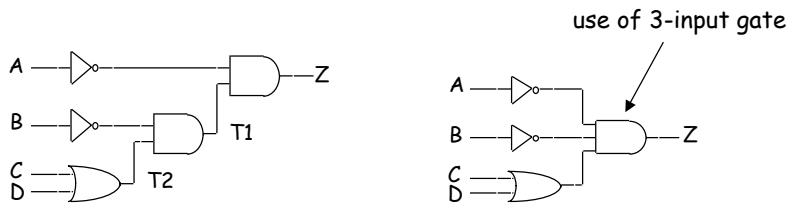
$X \text{ xnor } Y = X Y + X' Y'$
 X and Y are the same
 ("equality", "coincidence")

From expressions to logic gates

- More than one way to map expressions to gates

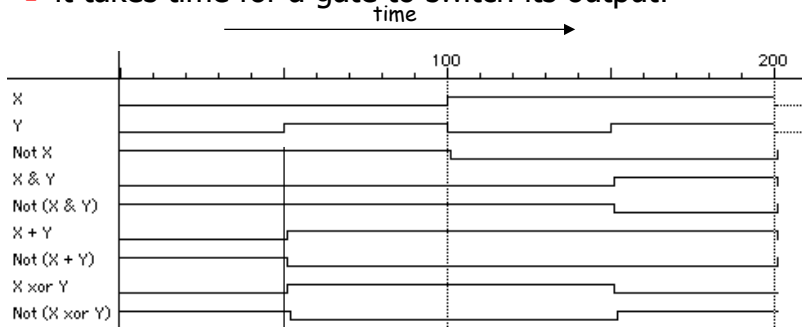
■ e.g., $Z = A' \cdot B' \cdot (C + D) = (A' \cdot (B' \cdot (C + D)))$

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



Waveform view of logic functions

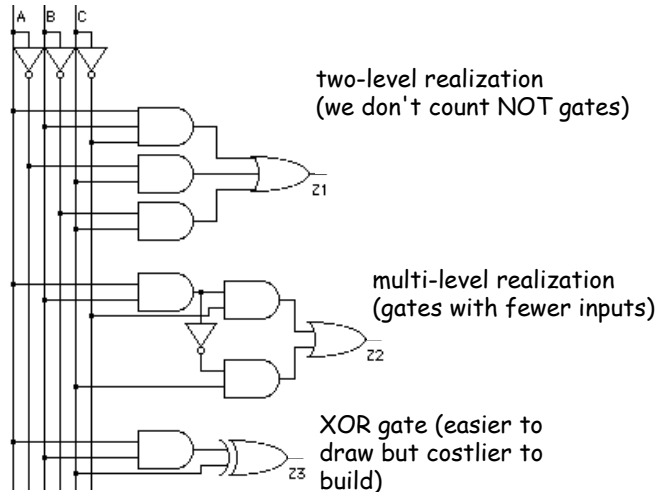
- Just a sideways truth table
 - but note how edges don't line up exactly
 - it takes time for a gate to switch its output!



change in Y takes time to "propagate" through gates

Different realizations of a function

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



EECS150 - Fall 2001

1-19

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 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

EECS150 - Fall 2001

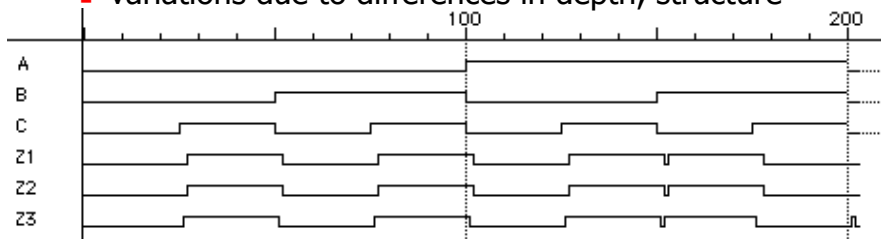
1-20

Which is the best realization?

- 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

Are all realizations equivalent?

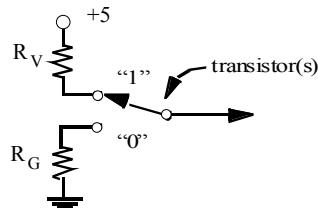
- Under the same input stimuli, the three alternative implementations have almost the same waveform behavior
 - delays are different
 - glitches (hazards) may arise
 - variations due to differences in depth, structure



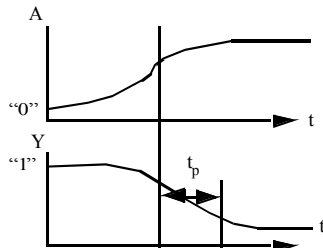
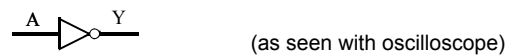
Timing

- Two key concepts: what {0,1} and when (timing)
 - How fast can I move or process information?
 - Fundamental limitations:
 - speed of light 3×10^8 m/sec vacuum
 $\sim 2 \times 10^8$ m/sec conduction
 - switching time: capacitance & inductance

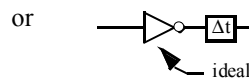
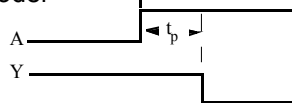
Zeroth order model



Voltage model (more physical)



4.3 Delay model



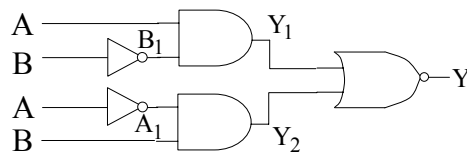
Timing

- Why big deal?
- Dynamic output \neq static output ("glitches")

Example: XOR function implemented from AND, OR, INV

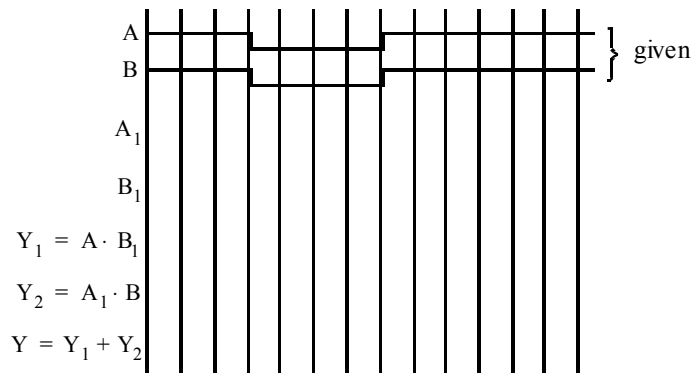
idealization: unit delay in each symbol

$$Y = A \cdot \bar{B} + \bar{A} \cdot B$$



Timing Diagram

given $\begin{matrix} A & 1 & \rightarrow & 0 & \rightarrow & 1 \\ B & 1 & \rightarrow & 0 & \rightarrow & 1 \end{matrix}$



Static behavior fine $1 \oplus 1 = 0, 0 \oplus 0 = 0 \checkmark$

Dynamic behavior **glitch** \rightarrow fundamental feature of switching

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
- Many alternative gate realizations may have the same truth table
- 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 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

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

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

canonical form \neq minimal form

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

short-hand notation for minterms of 3 variables

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

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

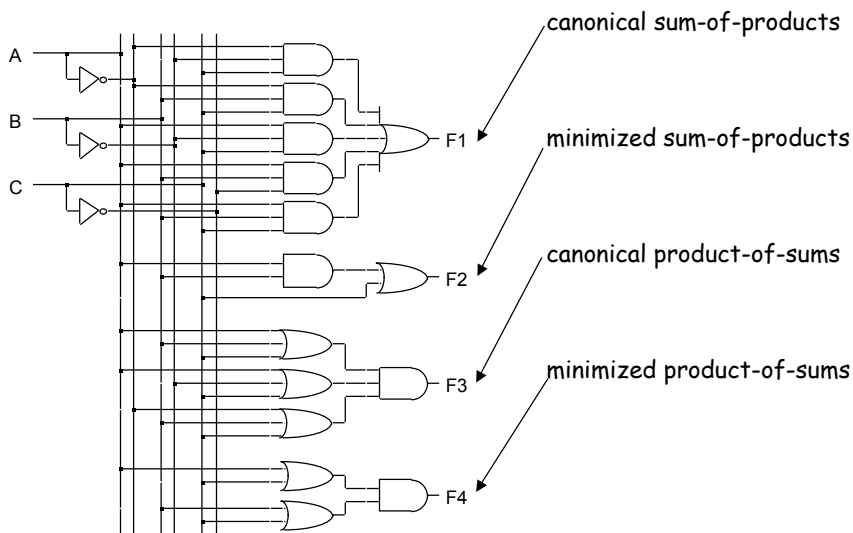
$$\begin{aligned} F(A, B, C) &= (A + B + C) (A + B' + C) (A' + B + C) \\ &= (A + B + C) (A + B' + C) \\ &\quad (A + B + C) (A' + B + C) \\ &= (A + C) (B + C) \end{aligned}$$

short-hand notation for maxterms of 3 variables

Relating S-o-P & P-o-S

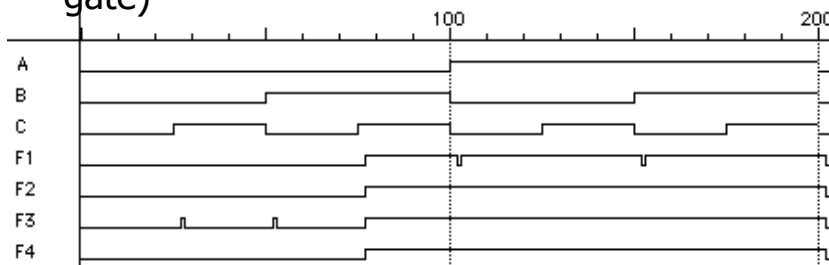
- Sum-of-products
 - $F' = A'B'C' + A'BC' + AB'C'$
- Apply de Morgan's
 - $(F')' = (A'B'C' + A'BC' + AB'C')'$
 - $F = (A + B + C)(A + B' + C)(A' + B + C)$
- Product-of-sums
 - $F = (A + B + C')(A + B' + C')(A' + B + C')(A' + B' + C)(A' + B' + C')$
- Apply de Morgan's
 - $(F')' = ((A + B + C')(A + B'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

Example: $F = AB + C$



Waveforms

- 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

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

Incompletely specified functions

- Example: binary coded decimal increment by 1
 - BCD digits encode decimal digits 0 – 9 in 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	1	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

- 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:
 - $Z = m_0 + m_2 + m_4 + m_6 + m_8 + d_{10} + d_{11} + d_{12} + d_{13} + d_{14} + d_{15}$
 - $Z = \Sigma [m(0,2,4,6,8) + d(10,11,12,13,14,15)]$
 - $Z = M_1 \cdot M_3 \cdot M_5 \cdot M_7 \cdot M_9 \cdot D_{10} \cdot D_{11} \cdot D_{12} \cdot D_{13} \cdot D_{14} \cdot D_{15}$
 - $Z = \Pi [M(1,3,5,7,9) \cdot D(10,11,12,13,14,15)]$