Chapter overview

- Logic functions and switches
  - NOT, AND, OR, NAND, NOR, XOR, . . .
- Gate Logic
  - laws and theorems of Boolean algebra
  - two-level canonical forms
  - incompletely specified functions
- Two-level simplification
  - Boolean cubes
  - Karnaugh maps
  - Quine-McCluskey method

© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: Boolean algebra

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

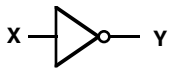
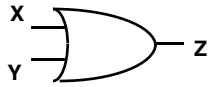
1. B contains at least two elements, a, b, such that $a \neq b$		
2. Closure:	$a + b$ is in B	$a \cdot b$ is in B
3. Idempotency:	$a + a = a$	$a \cdot a = a$
4. Identity elements:	$a + 0 = a$	$a \cdot 1 = a$
5. Commutative law:	$a + b = b + a$	$a \cdot b = b \cdot a$
6. Distributive law:	$a + (b \cdot c) = (a + b) \cdot (a + c)$	
	$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$	
7. Associative law:	$a + (b + c) = (a + b) + c$	
	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$	
8. Absorption:	$a + a \cdot b = a$	$a \cdot (a + b) = a$
9. Complement:	$a + a' = 1$	$a \cdot a' = 0$

- Switching algebra: B = {0, 1}, + = OR, · = AND, ' = NOT
  - verify the above laws (e.g., check that commutative law holds)

© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: Boolean algebra

- Theorem: any function that can be expressed as a truth table can be written as an expression in the {0,1}–Boolean algebra using the operators ', +, and •.

Operator	Gate	Truth table
Complement		
AND		
OR		

X	Y
0	1
1	0

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

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

X, Y, and Z are Boolean algebra variables

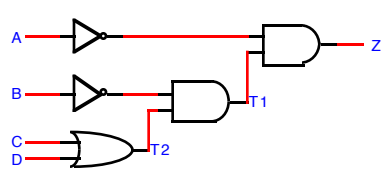
© 1992 Gaetano Borriello, 1994 Steven Burns

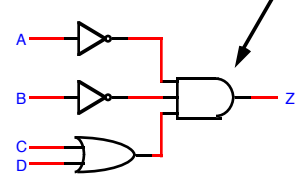
Logic functions: from expressions to gates

- More than one way to map expressions to gates

e.g., $Z = A' \cdot B' \cdot (C + D) = (A' \cdot (B' \cdot (C + D)))$ T2 T1

use of 3-input gate





- Literal: the appearance of a variable or its complement in an expression
- The number of literals is proportional to the number of switches needed
  - e.g., Z = A B' C + A' B + A' B C' + B' C 3 variables and 10 literals

© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: NAND, NOR, XOR, XNOR

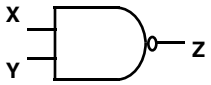
- 16 functions of two variables:

X	Y							F0–F15								
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
		"0"	"X · Y"		"X"		"Y"		"X + Y"		" Y' "		" X' "			"1"

- X, X', Y, Y', X · Y, X + Y, 0, 1 are only half of the possible functions

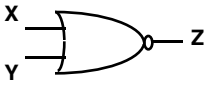
NAND

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



© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: NAND, NOR, minimal set

- NAND and NOR gates far outnumber AND and OR in typical designs
  - easier to construct and faster in underlying transistor technologies
- Any Boolean expression can be implemented by NAND, NOR, NOT gates
  - in fact, NOT is superfluous
  - (NOT = NAND or NOR with both inputs tied together)

X	Y	X NOR Y
0	0	1
1	1	0

X	Y	X NAND Y
0	0	1
1	1	0

- We'll see later that we need only one of NAND or NOR gates to implement any Boolean function whatsoever

© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: XOR, XNOR

- **XOR: X or Y but not both ("inequality", "difference")**
- **XNOR: X is the same as Y ("equality", "coincidence")**

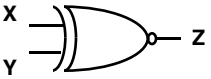
XOR

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

X xor Y = X Y' + X' Y

XNOR

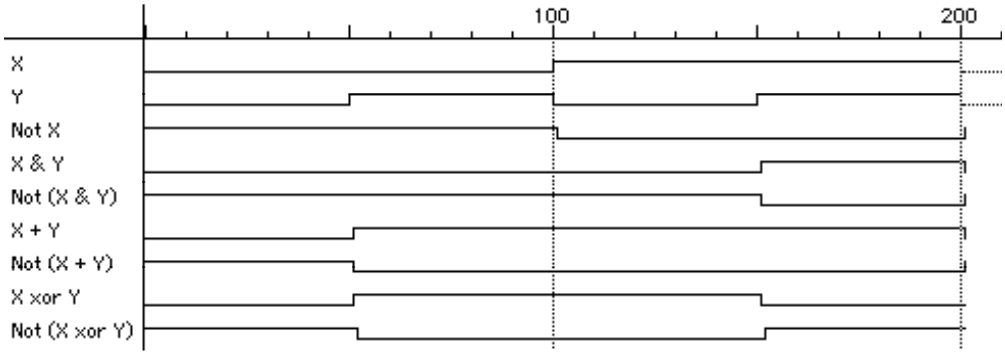
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1



X xnor Y = X Y + X' Y'

© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: waveform view



© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: rationale for simplification

- Logic minimization: reduce complexity of the gate level implementation

- 1. Reduce number of literals
  - fewer literals means less transistors (less space)
  - fewer inputs implies faster gates (less switches in series)
  - fan-ins (# of gate inputs) are limited in some technologies

- 2. Reduce number of gates
  - number of gates (or gate packages) influences manufacturing costs

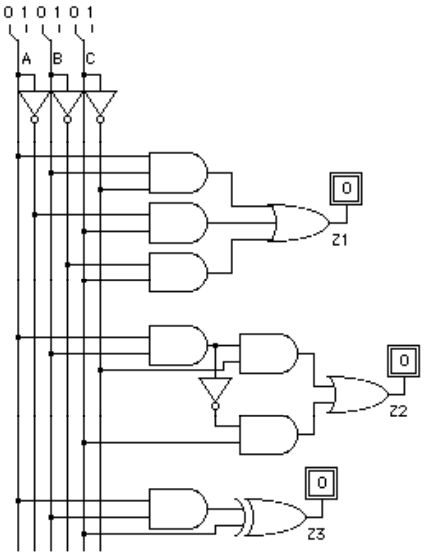
- 3. Reduce number of levels of gates
  - fewer level of gates implies reduced signal propagation delays
  - minimum delay configuration typically requires more gates
  - (wider less deep circuits)

- Explore tradeoffs between increased circuit delay and reduced gate count
  - automated tools to optimize logic and explore possibilities

© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: alternative gate realizations

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
(inverters don't count)

multi-level realization
(advantage: reduced
gate fan-ins)

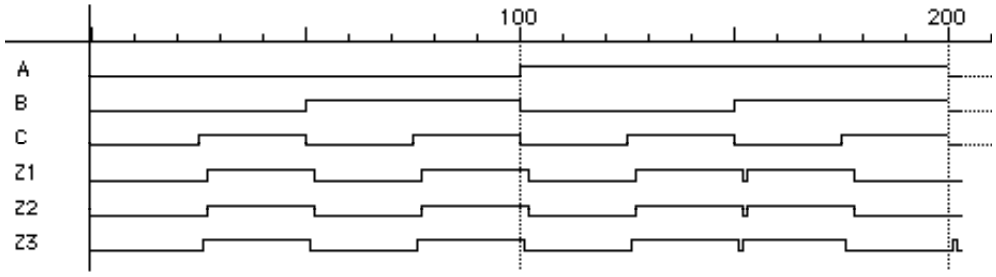
complex gate: XOR
(advantage: fewest gates)

TTL package counts (only one gate type per package):
- Z1 - 3 packages (hex inv, triple 3-input AND, triple 3-input NOR)
- Z2 - 3 packages (hex inv, quad 2-input AND, quad 2-input OR)
- Z3 - 2 packages (quad 2-input AND, quad 2-input XOR)

© 1992 Gaetano Borriello, 1994 Steven Burns

Logic functions: waveform verification

- Under the same input stimuli, the three alternative implementations have essentially the same waveform behavior
- Slight variations are due to differences in number of gate levels and details of realization
- The three implementations are functionally equivalent



© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- Duality: a dual of a Boolean expression is derived by replacing AND operations by ORs, OR operations by ANDs, constant 0s by 1s, and 1s by 0s (literals are left unchanged)

  <u>Any statement that is true for an expression is also true for its dual!</u>

- Useful laws/theorems of Boolean algebra:
  - operations with 0 and 1:
    - 1. $X + 0 = X$ — 1D. $X \cdot 1 = X$
    - 2. $X + 1 = 1$ — 2D. $X \cdot 0 = 0$
  - idempotent laws:
    - 3. $X + X = X$ — 3D. $X \cdot X = X$
  - involution law:
    - 4. $(X')' = X$
  - laws of complementarity:
    - 5. $X + X' = 1$ — 5D. $X \cdot X' = 0$

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- **Useful laws/theorems of Boolean algebra (continued):**

commutative laws:

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

associative laws:

7. (X + Y) + Z = X + (Y + Z)	**7D. (X · Y) · Z = X · (Y · Z)**

distributive laws:

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

simplification theorems:

9. X · Y + X · Y' = X	**9D. (X + Y) · (X + Y') = X**
10. X + X · Y = X	**10D. X · (X + Y) = X**
11. (X + Y') · Y = X · Y	**11D. (X · Y') + Y = X + Y**

concensus theorems:

12. (X · Y) + (Y · Z) + (X' · Z) =	**12D. (X + Y) · (Y + Z) · (X' + Z) =**
X · Y + X' · Z	**(X + Y) · (X' + Z)**

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- **Useful laws/theorems of Boolean algebra (continued):**

de Morgan's laws:

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

14. $f'(X1,X2,...,Xn,0,1,+,\cdot) = f(X1',X2',...,Xn',1,0,\cdot,+)$

duality:

15. $(X + Y + Z + ...)^D = X \cdot Y \cdot Z \cdot ...$ **15D. $(X \cdot Y \cdot Z \cdot ...)^D = X + Y + Z + ...$**

16. $f^D(X1,X2,...,Xn,0,1,+,\cdot) = f(X1,X2,...,Xn,1,0,\cdot,+)$

theorems for multiplying and factoring:

17. (X + Y) · (X' + Z) = X · Z + X' · Y **17D. X · Y + X' · Z = (X + Z) · (X' + Y)**

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- Proving theorems using the axioms of Boolean algebra:

  e.g., prove the theorem: X · Y + X · Y' = X

  e.g., prove the theorem: X + X · Y = X

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- Proving theorems using the axioms of Boolean algebra:

  e.g., prove the theorem: X · Y + X · Y' = X

  distributive law (8) X · Y + X · Y' = X · (Y + Y')
  complementarity law (5) X · (Y + Y') = X · (1)
  identity (1D) X · (1) = X √

  e.g., prove the theorem: X + X · Y = X

  identity (1D) X · 1 + X · Y = X · 1 + X · Y
  distributive law (8) X · 1 + X · Y = X · (1 + Y)
  identity (2) X · (1 + Y) = X · (1)
  identity (1D) X · (1) = X √

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- de Morgan's law

(X + Y)' = X' • Y'
NOR is equivalent to AND with inputs complemented

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

(X • Y)' = X' + Y'
NAND is equivalent to OR with inputs complemented

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

- It can be used to convert AND/OR expressions to OR/AND expressions

e.g., Z = A' B' C + A' B C + A B' C + A B C'
Z' = (A + B + C') • (A + B' + C') • (A' + B + C') • (A' + B' + C)

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- Apply the laws and theorems to simplify Boolean expressions

e.g., full adder's carry-out function

A' B C + A B' C + A B C' + A B C

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: laws of Boolean algebra

- Apply the laws and theorems to simplify Boolean expressions

  e.g., full adder's carry-out function

□ □ □ 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 B C
□ □ □ = A' B C + A B C + A B' C + A B C' + A B C
□ □ □ = (A' + A) B C + A B' C + A B C' + A B C
□ □ □ = (1) B C + A B' C + A B C' + A B C
□ □ □ = B C + A B' C + A B C' + A B C + A B C
□ □ □ = B C + A B' C + A B C + A B C' + A B C
□ □ □ = B C + A (B' + B) C + A B C' + A B C
□ □ □ = B C + A (1) C + A B C' + A B C
□ □ □ = B C + A C + A B (C' + C)
□ □ □ = B C + A C + A B (1)
□ □ □ = B C + A C + A B

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: two-level canonical forms

- Truth table is a unique signature of a Boolean function
- Many alternative expression (and gate realizations) may have the same truth table
- Canonical form: standard form for a Boolean expression provides a unique algebraic signature
- <u>Sum-of-products form</u> – also known as disjunctive normal form
  □ □ □ □ □ or the minterm expansion

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

011 100 101 110 111

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

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

© 1992 Gaetano Borriello, 1994 Steven Burns

Gate logic: two-level canonical forms

- **Product term (or minterm)**
  - **ANDed product of literals in which each variable appears exactly once,**
  - **in true or complemented 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

short-hand notation for minterms of 3 variables

F in canonical form:

- **F(A, B, C)=**

© 1992 Gaetano Borriello, 1994 Steven Burns