Boolean algebra

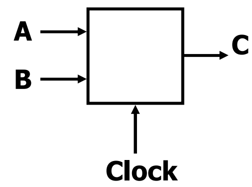
- Last lecture
  - Binary numbers
  - Base conversion
  - Number systems
    - Twos-complement
  - A/D and D/A conversion
- Today's lecture
  - Boolean algebra
    - Axioms
    - Useful laws and theorems
    - Simplifying Boolean expressions

Major topic: Combinational logic

- Axioms and theorems of Boolean algebra
- Logic functions and truth tables
  - AND, OR, Buffer, NAND, NOR, NOT, XOR, XNOR
- Gate logic
  - Networks of Boolean functions
- Canonical forms
  - Sum of products and product of sums
- Simplification
  - Boolean cubes and Karnaugh maps
  - Two-level simplification

Combinational versus sequential

- Combinational: Memoryless
  - Apply fixed inputs A, B
  - Wait for clock edge
  - Observe C
  - Wait for another clock edge
  - Observe C again: C will stay the same



- Sequential: With Memory
  - Apply fixed inputs A, B
  - Wait for clock edge
  - Observe C
  - Wait for another clock edge
  - Observe C again: C may be different

Boolean algebra

- A Boolean algebra comprises...
  - A set of elements B
  - Binary operators {+ , •}
  - A unary operation { ' }
- ...and the following axioms
  - 1. The set B contains at least two elements {a b} with a ≠ b

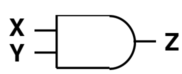
2. Closure:	a+b is in B	a•b is in B
3. Commutative:	a+b = b+a	a•b = b•a
4. Associative:	a+(b+c) = (a+b)+c	a•(b•c) = (a•b)•c
5. Identity:	a+0 = a	a•1 = a
6. Distributive:	a+(b•c)=(a+b)•(a+c)	a•(b+c)=(a•b)+(a•c)
7. Complementarity:	a+a' = 1	a•a' = 0

Digital (binary) logic is a Boolean algebra

- Substitute
  - {0, 1} for B
  - AND for •   Boolean Product
  - OR for +   Boolean Sum
  - NOT for '
- All the axioms hold for binary logic
- Definitions
  - Boolean function
    - Maps inputs from the set {0,1} to the set {0,1}
  - Boolean expression
    - An algebraic statement of Boolean variables and operators

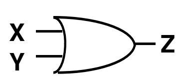
AND, OR, Not

- AND   X•Y   XY



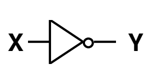
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

- OR   X+Y



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

- NOT $\overline{X}$ X'



X	Y
0	1
1	0

Logic functions and Boolean algebra

- *Any* logic function that is expressible as a truth table can be written in Boolean algebra using +, •, and '

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

Z=X•Y

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

Z=X'•Y

X	Y	X'	Y'	X • Y	X' •Y'	Z
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

Z=(X•Y)+(X' •Y')

Two key concepts

- Duality (a meta-theorem— *a theorem about theorems*)
  - All Boolean expressions have logical duals
  - Any theorem that can be proved is also proved for its dual
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Leave the variables unchanged
- de Morgan's Theorem
  - Procedure for complementing Boolean functions
  - Replace: • with +, + with •, 0 with 1, and 1 with 0
  - Replace all variables with their complements

Useful laws and theorems

Identity:	X + 0 = X	Dual: X • 1 = X
Null:	X + 1 = 1	Dual: X • 0 = 0
Idempotent:	X + X = X	Dual: X • X = X
Involution:	(X')' = X	
Complementarity:	X + X' = 1	Dual: X • X' = 0
Commutative:	X + Y = Y + X	Dual: X • Y = Y • X
Associative:	(X+Y)+Z=X+(Y+Z)	Dual: (X•Y)•Z=X•(Y•Z)
Distributive:	X•(Y+Z)=(X•Y)+(X•Z)	Dual: X+(Y•Z)=(X+Y)•(X+Z)
Uniting:	X•Y+X•Y'=X	Dual: (X+Y)•(X+Y')=X

Useful laws and theorems (con't)

Absorption:	X+X•Y=X	Dual: X•(X+Y)=X
Absorption (#2):	(X+Y')•Y=X•Y	Dual: (X•Y')+Y=X+Y
de Morgan's:	(X+Y+...)'=X'•Y'•...	Dual: (X•Y•...)'=X'+Y'+...
Duality:	$(X+Y+...)^D$=X•Y•...	Dual: $(X•Y•...)^D$=X+Y+...

Multiplying & factoring: (X+Y)•(X'+Z)=X•Z+X'•Y
Dual: X•Y+X'•Z=(X+Z)•(X'+Y)

Consensus: (X•Y)+(Y•Z)+(X'•Z)= X•Y+X'•Z
Dual: (X+Y)•(Y+Z)•(X'+Z)=(X+Y)•(X'+Z)

Proving theorems

- Example 1: Prove the uniting theorem-- X•Y+X•Y'=X

Distributive	X•Y+X•Y' = X•(Y+Y')
Complementarity	= X•(1)
Identity	= X

- Example 2: Prove the absorption theorem-- X+X•Y=X

Identity	X+X•Y = (X•1)+(X•Y)
Distributive	= X•(1+Y)
Null	= X•(1)
Identity	= X

Proving theorems

- Example 3: Prove the consensus theorem-- (XY)+(YZ)+(X'Z)= XY+X'Z

Complementarity	XY+YZ+X'Z = XY+(X+X')YZ + X'Z
Distributive	= XYZ+XY+X'YZ+X'Z

Use absorption {AB+A=A} with A=XY and B=Z

	= XY+X'YZ+X'Z
Rearrange terms	= XY+X'ZY+X'Z

Use absorption {AB+A=A} with A=X'Z and B=Y

XY+YZ+X'Z = XY+X'Z

de Morgan's Theorem

- Use de Morgan's Theorem to find complements
- Example: F=(A+B)•(A′+C), so F′=(A′•B′)+(A•C′)

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

A	B	C	F′
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Logic simplification

- Use the axioms to simplify logical expressions
  - Why? To use less hardware
- Example: A two-level logic expression

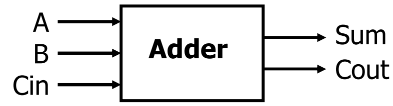
Z	= A'BC + AB'C' + AB'C + ABC' + ABC	
	= AB'C + AB'C' + A'BC + ABC' + ABC	rearrange
	= AB'(C + C') + A'BC + AB(C' + C)	distributive
	= AB' + A'BC + AB	comp.
	= AB' + AB + A'BC	rearrange
	= A(B' + B) + A'BC	distributive
	= A + A'BC	comp.

Use absorption #2D {(X•Y')+Y=X+Y} with X=BC and Y=A

Z = A + BC

Example: A full adder

- 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'BCin' + AB'Cin' + ABCin

Cout = A'BCin + AB'Cin + ABCin' + ABCin

Simplifying the carry-out function

Cout = A'BCin + AB'Cin + ABCin' + ABCin
= A'BCin + AB'Cin + ABCin' + ABCin + ABCin
= A'BCin + ABCin + AB'Cin + ABCin' + ABCin
= (A'+A)BCin + AB'Cin + ABCin' + ABCin
= (1)BCin + AB'Cin + ABCin' + ABCin
= BCin + AB'Cin + ABCin' + ABCin + ABCin
= BCin + AB'Cin + ABCin + ABCin' + ABCin
= BCin + A(B'+B)Cin + ABCin' + ABCin
= BCin + A(1)Cin + ABCin' + ABCin
= BCin + ACin + AB(Cin'+Cin)
= BCin + ACin + AB(1)
= BCin + ACin + AB

associative

idempotent

Some notation

- Priorities: $\overline{A} \bullet B + C = ((\overline{A}) \bullet B) + C$
- Variables are sometimes called literals