05-Boolean Algebra Part 3

Text: Unit 3, 7

ECEGR/ISSC 201
Digital Operations and Computations
Winter 2011

Overview

- DeMorgan's Laws
- Consensus Theorem
- Algebraic Simplifications
- Proving Validity
- Exclusive-OR and Equivalence
- Functionally Complete
- NAND-NOR gates

DeMorgan's Law

- DeMorgan's Laws are helpful when complements appear in expressions:
  - (X+Y)' = X'Y'
  - (XY)' = X' + Y'

Example

- Use a truth table to show
  - (X+Y)' = X'Y'

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

Example

- Use a truth table to show
  - (X+Y)' = X'Y'

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

DeMorgan's Law

- Is X'Y' = (XY)'?

DeMorgan's Law

- Is X'Y' = (XY)'?
  - NO!

X	Y	X Y	(XY)'	X'Y'
0	0	0	1	1
0	1	0	1	0
1	0	0	1	0
1	1	1	0	0

DeMorgan's Laws

- Can be generalized to n variables:
  - $(X_1 + \ldots X_n)' = X'_1 \ldots X'_n$
  - $(X_1 \ldots X_n)' = X'_1 + \ldots X'_n$
- The complement of the sum is the product of the complements
- The complement of the product is the sum of the complements

DeMorgan's Laws

- Find the complement of (A'+B)C'

DeMorgan's Laws

- Find the complement of (A'+B)C'
  - (X+Y)' = X'Y'
  - (XY)' = X' + Y'
  - ((A'+B)C')' = (A'+B)' + C = AB' + C
    - First set (A'+B) = X; C' = Y
    - Then set A' = X; B = Y

Consensus Theorem

- Complete the truth table for: XY + X'Z +YZ

X	Y	Z	XY	X'Z	YZ	XY + X'Z	XY + X'Z +YZ
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Consensus Theorem

- Consider: XY + X'Z +YZ

X	Y	Z	XY	X'Z	YZ	XY + X'Z	XY + X'Z +YZ
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

Identical columns

Consensus Theorem

- In XY + X'Z +YZ, the YZ is redundant
- Given two pairs (in this case XY and X'Z) for which one variable appears in one term and it's complement in the other (X in this case), then the consensus term is found by multiplying the two original terms together
- See proof (page 63)

Consensus Theorem

- Formally
  - XY +X'Z + YZ = XY +X'Z
- Example: identify X, Y and Z in
  BDA' + ABC + BCBD

Consensus Theorem

- Formally
  - XY +X'Z + YZ = XY +X'Z
- Example: identify X, Y and Z in
  BDA' + ABC + BCBD
  - X = A
  - Y = BD
  - Z = BC

  BDA' + ABC + BCBD = BDA' + ABC
- Note: usually you will not see BCBD (instead: BCD), but you can still use the theorem

Algebraic Simplifications

- Options are:

1. Combine Terms
   - XY + XY' = X
2. Eliminate Terms
   - X + XY = X
3. Eliminate Literals
   - X + X'Y = X +Y
4. Add Redundant Terms
   - Add XX'
   - Multiply by (X + X')
   - Add YZ to XY + X'Z (consensus)

Example

- Simplify:

A'B'C'D + A'BC'D + A'BD + A'BC'D + ABCD+ACD' + B'CD

Example

- Simplify:

A'B'C'D + A'BC'D + A'BD + A'BC'D + ABCD+ACD' + B'CD

combining terms: A'B'C'D + A'BC'D = A'C'D

A'C'D + A'BD + ~~A'BC'D~~ + ABCD+ACD' + B'CD

eliminating terms: A'BD + A'BC'D = A'BD

Example

- Simplify:

A'C'D + A'BD + ABCD+ACD' + B'CD

A'C'D + BD(A'+AC) + ACD' + B'CD

eliminating literals: A' + AC = A' + C

A'C'D + BD(A'+ C) + ACD' + B'CD

A'C'D + BDA'+ BCD + ACD' + B'CD

adding redundant terms: BCD + ACD' = BCD + ACD' + ABCC

A'C'D + BDA'+ BCD + ACD' + B'CD + ABC

Example

- Simplify:

A'C'D + BDA'+ ~~BCD~~ + ACD' + B'CD' + ABC

consensus: BDA' + ABC + BCBD = BDA' + ABC

A'C'D + BDA' + ~~ACD'~~ + B'CD' + ABC

consensus: B'CD' + ABC + ACCD' = B'CD' + ABC

A'C'D + BDA' + B'CD' + ABC

Proving Validity

- There are a number of ways to prove that
  - A'C'D + BDA' + B'CD' + ABC and

A'B'C'D + A'BC'D + A'BD + A'BC'D + ABCD+ACD' + B'CD

are equivalent expressions

Proving Validity

- In ordinary algebra we can show equivalence
  - x + y = x + z is the same as y = z by subtracting x from both sides:
  - x + y − x = x + z −x => y = z
- cancellation does not work in Boolean algebra
- X + Y = X + Z does imply that Y = Z
- Let X = 1, Y = 0, Z =1
- 1 + 0 = 1 + 1 (but Y does not equal Z)

Proving Validity

- Subtraction is not defined in Boolean algebra
- Division is not defined either:
  - xy = xz implies y = z (in ordinary algebra, so long as x does not equal zero)
  - XY = XZ does not imply Y = Z in Boolean algebra
  - X will be equal to 0 about half the time

Proving Validity

1. Construct a truth table and check for identical results to all input combinations
2. Manipulate one side using various theorems until it is identical to the other side
3. Reduce both sides independently to the same expression
4. Perform reversible operations to both sides until equivalent

Exclusive-OR

- Exclusive OR (⊕), XOR,
  - 0 XOR 0 = 0
  - 0 XOR 1 = 1
  - 1 XOR 0 = 1
  - 1 XOR 1 = 0
- How can we construct the XOR with ANDs, ORs and NOTs?

Exclusive-OR

- Exclusive OR (⊕), XOR,
  - 0 XOR 0 = 0
  - 0 XOR 1 = 1
  - 1 XOR 0 = 1
  - 1 XOR 1 = 0
- How can we construct the XOR with ANDs and ORs?
  - X⊕Y = X'Y + XY'

Exclusive OR Theorems

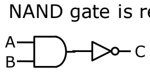
- X⊕0 = X
- X⊕1 = X'
- X⊕X = 0
- X⊕X' = 1
- X⊕Y = Y⊕X
- (X⊕Y)X = X⊕(X⊕Y)
- X(Y⊕Z) = XY⊕XZ
- (X⊕Y)' = X⊕Y' = X'⊕Y =XY + X'Y'

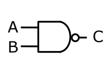
Equivalence Operation

- Equivalence operation: ≡,
- Compares values, if equal, then 1, else 0
- (0≡0) = 1
- (0≡1) = 0
- (1≡0) = 0
- (1≡1) = 1
- Equivalence is the opposite of XOR
- Equivalence is commutative and associative

NAND and NOR Gates

- NAND gate is really an AND-NOT gate



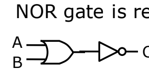


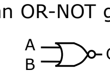
- Truth table is the complement of an AND gate

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

NAND and NOR Gates

- NOR gate is really an OR-NOT gate



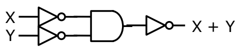


- Truth table is the complement of an OR gate

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

Functionally Complete

- A set of logic operations (e.g. AND, OR, NOT) are functionally complete if and Boolean function can be expressed in terms of this set of operations
- The set AND, OR and NOT is functionally complete
- The set of AND and NOT is functionally complete
  - OR gate can be constructed from them

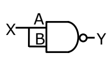


NAND and NOR Gates

- NAND gates alone are functionally complete
  - Derive NOT using NAND gate(s)

NAND and NOR Gates

- NAND gates alone are functionally complete
  - Derive NOT using NAND gate(s)
  - Y = (AB)′ = A′ + B′; X = A = B
  - Y = X′ + X′ = X′



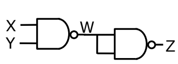
X	A	B	Y
0	0	0	1
1	1	1	0

NAND and NOR Gates

- NAND gates alone are functionally complete
  - Derive AND using NAND gate(s)

NAND and NOR Gates

- NAND gates alone are functionally complete
  - Derive AND using NAND gate(s)
  - Z = (W)′; W = (XY)′
  - Z = ((XY)′)′ = (X′ + Y′)′ = XY



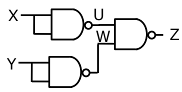
X	Y	W	Z
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

NAND and NOR Gates

- NAND gates alone are functionally complete
  - Derive OR using NAND gate(s)

NAND and NOR Gates

- NAND gates alone are functionally complete
  - Derive OR using NAND gate(s)
  - Z = (UW)’ = (U’W’)’ = U + W



X	Y	U	W	Z
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

NAND and NOR Gates

- We can show that NOR is also functionally complete