Lecture 5

- Logistics
  - HW2 posted on Wed, due 10/8
  - Lab1 done
  - Final exam schedule confirmed: 12/8 8:30am here Gowen 201
- Last lecture
  - Logic gates and truth tables
  - Implementing logic functions
- Today's lecture
  - Example using de Morgan's theorem
  - Canonical forms
  - Converting to use NAND and NOR

The "WHY" slide

- More de Morgan's theorem?
  - De Morgan's theorem is extremely useful and is used all the time for design implementation. But it is initially not an easy thing to do, so practicing it over and over is critical.
- Canonical forms
  - There are many forms to expression one Boolean function. It is good to have one standard way. Canonical form is the standard form for Boolean expressions. It has a nice property that allows you to go back and forth between truth table/expressions/gates easily.
- Converting to use NAND and NOR
  - NAND and NOR are more efficient gates than AND or OR (and therefore more common). Your computer is built almost exclusively on NAND and NOR gates. It is good to knowhow to convert any logic circuits to a NAND/NOR circuit.

de Morgan's theorem

- Replace
  - • with +, + with •, 0 with 1, and 1 with 0
  - All variables with their complements
- Example 1: Z = A'B' + A'C'

  Z' = (A'B' + A'C')'

  = (A+B) • (A+C)
- Example 2: Z = A'B'C + A'BC + AB'C + ABC'

  Z' = (A'B'C + A'BC + AB'C + ABC')'

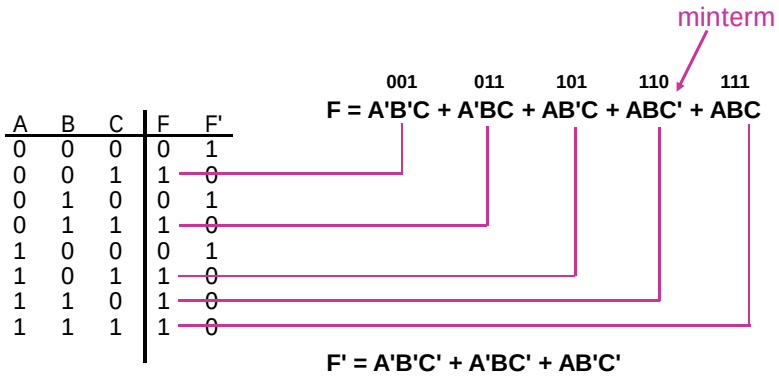
  = (A+B+C') • (A+B'+C') • (A'+B+C') • (A'+B'+C)

Canonical forms

- Canonical forms
  - Standard forms for Boolean expressions
  - Generally not the simplest forms
    - Can be minimized
  - Derived from truth table
- Two canonical forms
  - Sum-of-products (minterms)
  - Product-of-sum (maxterms)

Sum-of-products canonical form (SOP)

- Also called disjunctive normal form (DNF)
  - Commonly called a **minterm expansion**



Minterms

- Variables appears exactly once in each minterm
  - 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

short-hand notation

F in canonical form:

$F(A,B,C) = \Sigma m(1,3,5,6,7)$

$= m1 + m3 + m5 + m6 + m7$

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

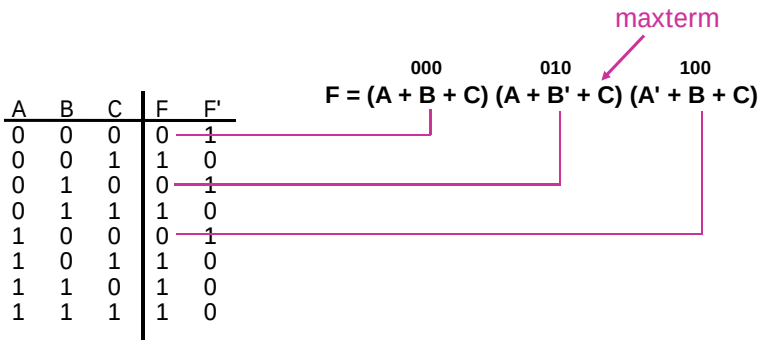
canonical form $\rightarrow$ minimal form

$F(A,B,C) = A'B'C+A'BC+AB'C+ABC+ABC'$

$= AB + C$

Product-of-sums canonical form (POS)

- Also called conjunctive normal form (CNF)
  - Commonly called a **maxterm expansion**



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')

Maxterms

- Variables appears exactly once in each maxterm
  - 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

short-hand notation

F in canonical form:

$F(A,B,C) = \Pi M(0,2,4)$

$= M0 \bullet M2 \bullet M4$

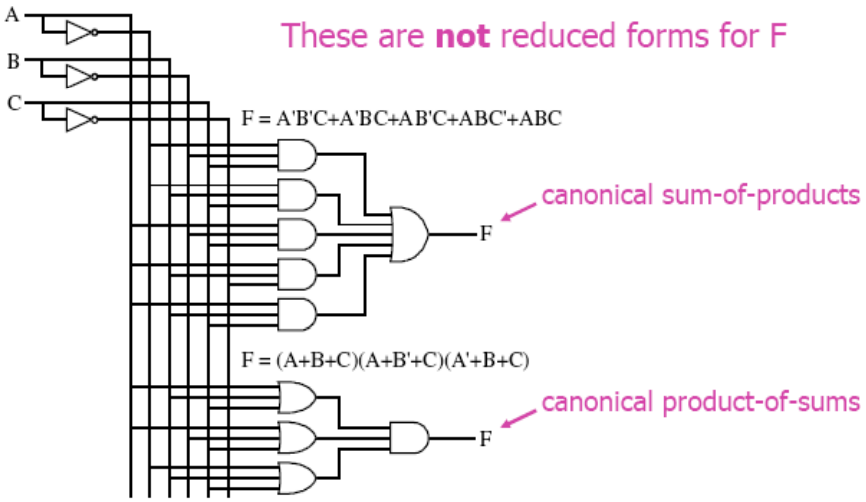
$= (A+B+C)(A+B'+C)(A'+B+C)$

canonical form $\rightarrow$ minimal form

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

$= AB + C$

Canonical implementations of F = AB + C



Conversion between canonical forms

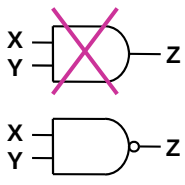
- Minterm to maxterm
  - Use maxterms that aren't in minterm expansion
  - F(A,B,C) = $\sum$m(1,3,5,6,7) = $\prod$M(0,2,4)
- Maxterm to minterm
  - Use minterms that aren't in maxterm expansion
  - F(A,B,C) = $\prod$M(0,2,4) = $\sum$m(1,3,5,6,7)
- Minterm of F to minterm of F'
  - Use minterms that don't appear
  - F(A,B,C) = $\sum$m(1,3,5,6,7) F'(A,B,C) = $\sum$m(0,2,4)
- Maxterm of F to maxterm of F'
  - Use maxterms that don't appear
  - F(A,B,C) = $\prod$M(0,2,4) F'(A,B,C) = $\prod$M(1,3,5,6,7)

SOP, POS, and de Morgan's theorem

- Sum-of-products
  - F' = A'B'C' + A'BC' + AB'C'
- Apply de Morgan's to get POS
  - (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 to get SOP
  - (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

NAND/NOR more common/efficient

- CMOS logic gates are more common and efficient in the inverted forms
  - NAND, NOR, NOT
  - Even though Canonical forms discussed so far used AND/OR, NAND/NOR preferred for real hardware implementation



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

NAND and NOR (truth table)

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

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

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

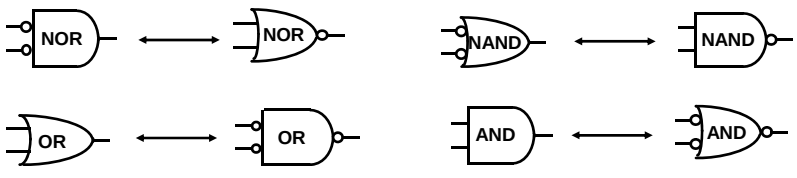
NAND is equivalent to OR with inputs complemented

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

NAND and NOR (logic gates)

- de Morgan's
  - Standard form: A'B' = (A + B)' A' + B' = (AB)'
  - Inverted: A + B = (A'B')' (AB) = (A' + B')'
  - AND with complemented inputs ≡ NOR
  - OR with complemented inputs ≡ NAND
  - OR ≡ NAND with complemented inputs
  - AND ≡ NOR with complemented inputs

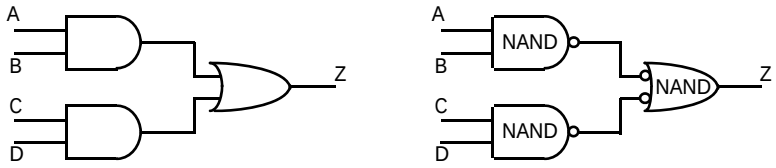
pushing the bubble



Converting to use NAND/NOR

- Introduce inversions ("bubbles")
  - Introduce bubbles in pairs
    - Conserve inversions
    - Do not alter logic function
- Example
  - AND/OR to NAND/NAND

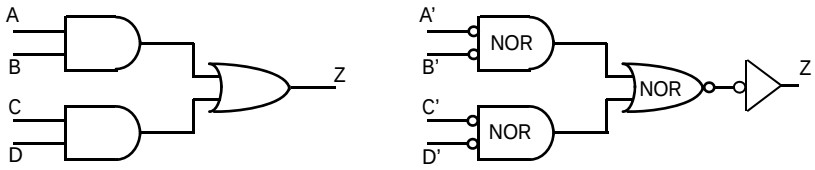
Z = AB + CD
= (A'+B')'+(C'+D')'
= [(A'+B')(C'+D')]'
= [(AB)'(CD)']'



Converting to use NAND/NOR (con't)

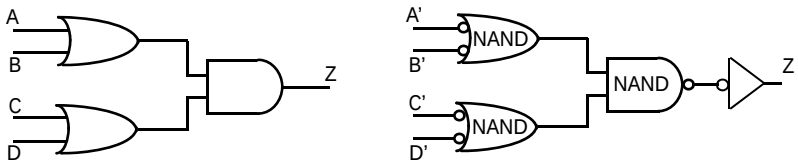
- Example: AND/OR network to NOR/NOR

Z = AB+CD
= (A'+B')'+(C'+D')'
= [(A'+B')'+(C'+D')']''
= {[(A'+B')'+(C'+D')']'}'



Converting to use NAND/NOR (con't)

- Example: OR/AND to NAND/NAND



Converting to use NAND/NOR(con't)

- Example: OR/AND to NOR/NOR