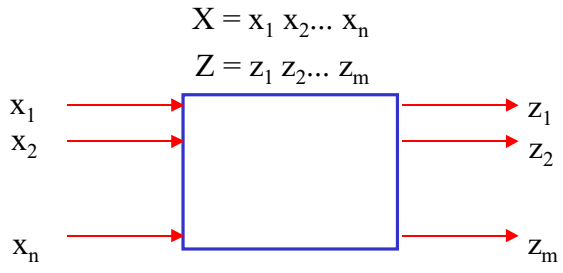
CPE/EE 422/522
Spring 2004
Chapter 1 - Review of Logic Design Fundamentals

Dr. Rhonda Kay Gaede

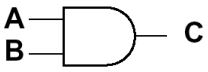
UAH

1.1 Combinational Logic

- Combinational Logic has no control inputs. When the inputs to a combinational network change, the output changes _____.

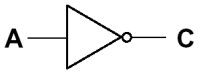


1.1 Combinational Logic - Basic Logic Gates



AND: C = A B

OR: C = A + B

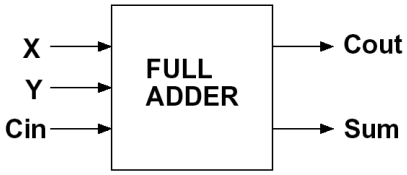


NOT: C = A'

EXCLUSIVE OR: C = A ⊕ B

1.1 Combinational Logic - Full Adder (Minterm Form)

Module



Truth table

X	Y	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
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	1

m-form

Sum =

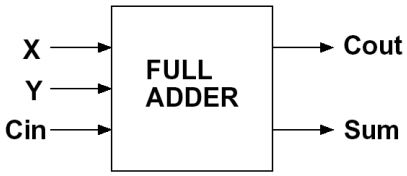
Cout =

Sum =

Cout =

1.1 Combinational Logic - Full Adder (Maxterm Form)

Module



Truth table

X	Y	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
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	1

$\overline{Sum}$ =

Sum =

$\overline{Cout}$ =

Cout =

m-form

Sum =

Cout =

1.2 Boolean Algebra and Algebraic Simplification

Operations with 0 and 1:

$X + 0 = X$ (1-5) $X \cdot 1 = X$ (1-5D)

$X + 1 = 1$ (1-6) $X \cdot 0 = 0$ (1-6D)

Idempotent laws:

$X + X = X$ (1-7) $X \cdot X = X$ (1-7D)

Involution law:

$(X')' = X$ (1-8)

Laws of complementarity

$X + X' = 1$ (1-9) $X \cdot X' = 0$ (1-9D)

Commutative laws:

$X + Y = Y + X$ (1-10) $XY = YX$ (1-10D)

Associative laws:

$(X + Y) + Z = X + (Y + Z) = X + Y + Z$ (1-11) $(XY)Z = X(YZ) = XYZ$ (1-11D)

Distributive laws:

$X(Y + Z) = XY + XZ$ (1-12) $X + YZ = (X + Y)(X + Z)$ (1-12D)

1.3 More Boolean Algebra and Algebraic Simplification

Simplification theorems:

$XY + XY' = X$	(1-13)	$(X + Y)(X + Y') = X$	(1-13D)
$X + XY = X$	(1-14)	$X(X + Y) = X$	(1-14D)
$(X + Y')Y = XY$	(1-15)	$XY' + Y = X + Y$	(1-15D)

DeMorgan's laws:

$(X + Y + Z + \cdots)' = X'Y'Z'\cdots$	(1-16)	$(XYZ\cdots)' = X' + Y' + Z' + \cdots$	(1-16D)
$[f(X_1, X_2, \ldots, X_n, 0, 1, +, \cdot)]' = f(X_1', X_2', \ldots, X_n', 1, 0, \cdot, +)$			(1-17)

Duality:

$(X + Y + Z + \cdots)^D = XYZ\cdots$	(1-18)	$(XYZ\cdots)^D = X + Y + Z + \cdots$	(1-18D)
$[f(X_1, X_2, \ldots, X_n, 0, 1, +, \cdot)]^D = f(X_1, X_2, \ldots, X_n, 1, 0, \cdot, +)$			(1-19)

Theorem for multiplying out and factoring:

$(X + Y)(X' + Z) = XZ + X'Y$	(1-20)	$XY + X'Z = (X + Z)(X' + Y)$	(1-20D)

Consensus theorem:

$XY + YZ + X'Z = XY + X'Z$	(1-21)	$(X + Y)(Y + Z)(X' + Z) = (X + Y)(X' + Z)$	(1-21D)

1.2 Boolean Algebra with Exclusive OR

$X \oplus 1 = X'$

$X \oplus 0 = X$

$X \oplus X = 0$

$X \oplus X' = 1$

$X \oplus Y = Y \oplus X$ (Commutative law)

$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$

$X(Y \oplus Z) = XY \oplus XZ$ (Distributive law)

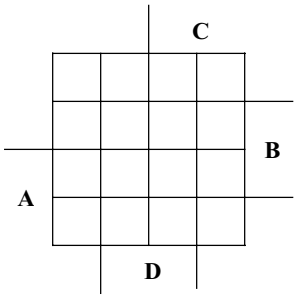
$(X \oplus Y)' = X \oplus Y' = X' \oplus Y = XY + X'Y'$

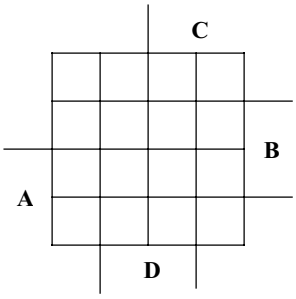
1.3 Karnaugh Maps

- Convenient way to simplify logic functions of 3, 4, 5, (6) variables
- Four-variable K-map
  - Each square
    - 1
    - 0
    - d
  - adjacent cells

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

1.3 Karnaugh Maps - Examples





1.3 Karnaugh Maps - Sum of Products

- Function consists of a sum of _____
- Prime implicant

  _____
  __

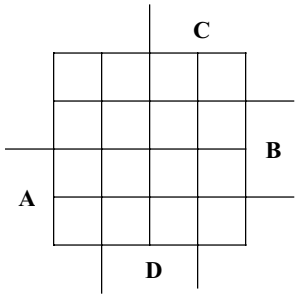
  _____
  __
- Prime implicant is _____ if it contains a 1 that is not contained in any other prime implicant

1.3 Karnaugh Maps - Prime Implicant Selection

- Two minimum forms

  f =

  f =

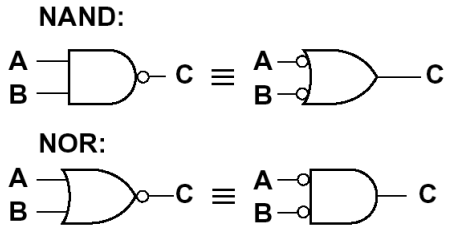


1.3 Karnaugh Maps - Selection Procedure

1. Choose a 1 (minterm) that has not been covered yet
2. Find all 1s and ds adjacent to that minterm. (Check the n adjacent squares on an n-variable map.)
3. If a single term covers the minterm and all the adjacent 1s and ds, then that term is an essential prime implicant, so select that term.
4. Repeat steps 1, 2, 3 until all essential prime implicants have been chosen
5. Find a minimum set of prime implicants that cover the remaining 1s on the map. If there is more than one such set, choose a set with a minimum number of literals

1.4 Designing with NAND and NOR Gates

¥ Implementation of NAND and NOR gates is easier than that of AND and OR gates (e.g., CMOS)



1.4 Designing with NAND and NOR Gates (continued)

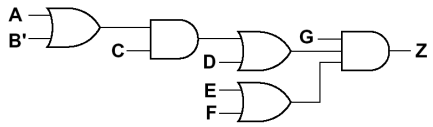
- Any logic function can be realized using only NAND or NOR gates -

  _____

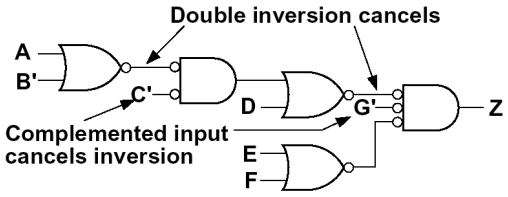
  - 1:
  - 0:
  - a :
  - ab:
  - a+b:

1.4 Designing with NAND and NOR Gates - Conversion to NOR Gates

- Start with POS (Product Of Sums)
  - circle 0s in K-maps
- Find network of OR and AND gates



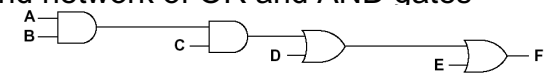
(a) AND-OR network



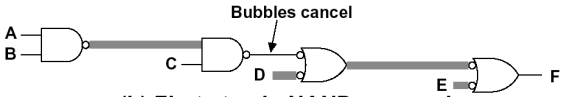
(b) Equivalent NOR-gate network

1.4 Designing with NANDand NOR Gates - Conversion to NAND Gates

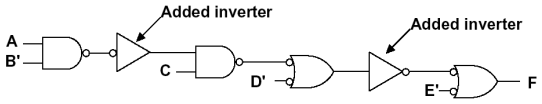
- Start with SOP (Sum of Products)
  - circle 1s in K-maps
- Find network of OR and AND gates



(a) AND_OR network



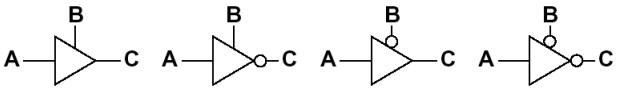
(b) First step in NAND conversion



(c) Completed conversion

1.13 Tristate Logic and Busses

- Four kinds of tristate buffers
  - B is a control input used to enable and disable the output



B	A	C
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

(a)

B	A	C
0	0	Hi-Z
0	1	Hi-Z
1	0	1
1	1	0

(b)

B	A	C
0	0	0
0	1	1
1	0	Hi-Z
1	1	Hi-Z

(c)

B	A	C
0	0	1
0	1	0
1	0	Hi-Z
1	1	Hi-Z

(d)

1.13 Tristate Logic and Busses - Data Transfer