Introduction to Computer Engineering – EECS 203

http://ziyang.eecs.northwestern.edu/~dickrp/eecs203/

Instructor:	Robert Dick	TA:	Neal Oza
Office:	L477 Tech	Office:	Tech. Inst. L375
Email:	dickrp@northwestern.edu	Phone:	847-467-0033
Phone:	847–467–2298	Email:	nealoza@u.northwestern.edu
		TT:	David Bild
		Office:	Tech. Inst. L470
		Phone:	847-491-2083
		Email:	d-bild@northwestern.edu

NORTHWESTERN UNIVERSITY

Quiz (ungraded)

- What is the relationship between a truth table and a Boolean formula? Is it one/many-to-one/many?
- What is the relationship between a Boolean formula and a combinational circuit? Is it one/many-to-one/many?
- What quirks does the CMOS implementation technology have?
- What is signal restoration or voltage regeneration?
- What practices are unsafe in switch-based design? Why?

Boolean algebra

- Set of elements, B
- Binary operators, { [AND, $\wedge$, *, $\cdot$], [OR, $\vee$, $+$] }
  - We'll prefer $\cdot$ and $+$
  - $\cdot$ frequently omitted
- Unary operator, [NOT, ', $\overline{o}$]

Axioms of Boolean algebra

$$\exists x, y \in B \text{ s.t. } x \neq y$$

closure	$\forall x, y \in B$	$xy \in B$
		$x + y \in B$
commutative laws	$\forall x, y \in B$	$xy = yx$
		$x + y = y + x$
identities	$0, 1 \in B, \forall x \in B$	$x1 = x$
		$x + 0 = x$

Axioms of Boolean algebra

$$\exists x, y \in B \text{ s.t. } x \neq y$$

distributive laws	$\forall x, y, z \in B$	$x + (yz) = (x + y)(x + z)$
		$x(y + z) = xy + xz$
complement	$x \in B$	$x\overline{x} = 0$
		$x + \overline{x} = 1$

De Morgan's laws

$$\overline{(a + b)} = \overline{a}\,\overline{b}$$

$$\overline{ab} = \overline{a} + \overline{b}$$

$$\overline{f(x_1, x_2, \ldots, x_n, \cdot, +)} = f(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}, +, \cdot)$$

- Those xs could be functions
- Apply in stages
  - Top-down

De Morgan's laws example

$$\overline{a + bc}$$

$$\overline{a} \cdot \overline{(bc)}$$

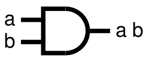
$$\overline{a} \cdot (\overline{b} + \overline{c})$$

Representations of Boolean functions

- Truth table
- Expression using only $\cdot$, $+$, and '
- Symbolic
- Karnaugh map
  - More useful as visualization and optimization tool

Review: *AND*

a	b	a b
0	0	0
0	1	0
1	0	0
1	1	1

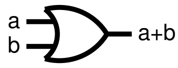


a *AND* b = a b

Will show Karnaugh map later

Review: *OR*

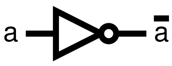
a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1



a *OR* b = a + b

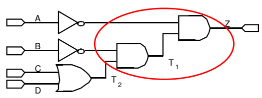
Review: *NOT*

a	$\overline{a}$
0	1
1	0

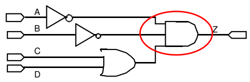


NOT a = $\overline{a}$

Different representations possible



$Z = ((C + D)\overline{B})\overline{A}$



$Z = (C + D)\overline{A}\,\overline{B}$

Simplifying logic functions

- Minimize literal count (related to gate count, delay)
- Minimize gate count
- Minimize levels (delay)
- Trade off delay for area
  - Sometimes no real cost

Proving theorems = simplification

Prove $XY + X\overline{Y} = X$

$XY + X\overline{Y} = X(Y + \overline{Y})$	distributive law
$X(Y + \overline{Y}) = X(1)$	complementary law
$X(1) = X$	identity law

Proving theorems = simplification

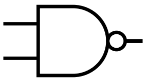
Prove $X + XY = X$

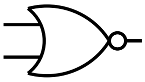
$X + XY = X1 + XY$	identity law
$X1 + XY = X(1 + Y)$	distributive law
$X(1 + Y) = X1$	identity law
$X1 = X$	identity law

Literals

- Each appearance of a variable (complement) in expression
- Fewer literals usually implies simpler to implement
- E.g., $Z = A\overline{B}C + \overline{A}B + \overline{A}B\overline{C} + \overline{B}C$
  - Three variables, ten literals

NANDs and NORs





- Can be implemented in CMOS
  - More on this later
- X NAND $Y = \overline{XY}$
- X NOR $Y = \overline{X + Y}$
- Do we need inverters?

Summary

- Administrative details
- Finished CMOS and basic gates
- Boolean algebra

Reading assignment

- M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, fourth edition, 2008
- Sections 2.4 and 2.5

Next lectures

- Karnaugh maps
- Visual minimization
- We'll also learn the optimal Quine-McCluskey method
  - Optimal two-level minimization is fun!