0001
Introduction to Digital Logic

ENGR 3410 – Computer Architecture
Fall 2010

Acknowledgements

- Patterson & Hennessy: Book & Lecture Notes
- Patterson's 1997 course notes (U.C. Berkeley CS 152, 1997)
- Tom Fountain 2000 course notes (Stanford EE182)
- Michael Wahl 2000 lecture notes (U. of Siegen CS 3339)
- Ben Dugan 2001 lecture notes (UW-CSE 378)
- Professor Scott Hauck lecture notes (UW EE 471)
- Mark L. Chang lecture notes for Digital Logic (NWU B01)
- Mark Chang's notes for ENGR 3410 (which these are, with some modifications by Mark Sheldon and Alex Morrow)

Example: Car Electronics

- Door ajar light (driver door, passenger door):

- High-beam indicator (lights, high beam selected):

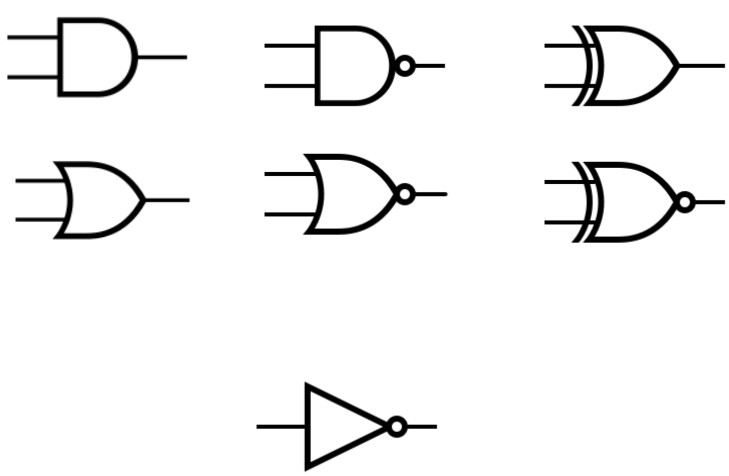
Example: Car Electronics (cont.)

- Seat Belt Light (driver belt in):

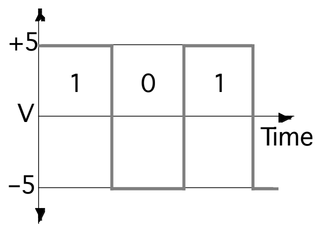
- Seat Belt Light (driver belt in, passenger belt in, passenger present):

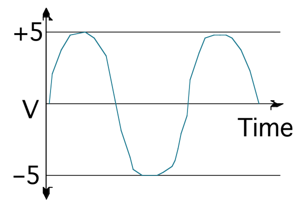
Basic Logic Gates

Logic gates



Digital vs. Analog





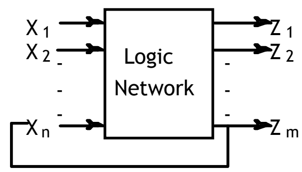
Digital:
only assumes discrete values

Analog:
values vary over a broad range continuously

Advantages of Digital Circuits

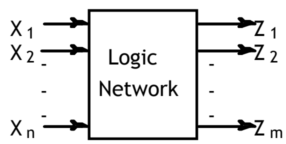
Combinational vs. Sequential Logic

Sequential logic



Network implemented from logic gates. The presence of memory distinguishes *sequential* and *combinational* networks.

Combinational logic



No feedback among inputs and outputs.
Outputs are a function of the inputs only.

Truth Tables

Algebra: variables, values, operations

In Boolean algebra, the values are the symbols 0 and 1
If a logic statement is false, it has value 0
If a logic statement is true, it has value 1

Operations: AND, OR, NOT

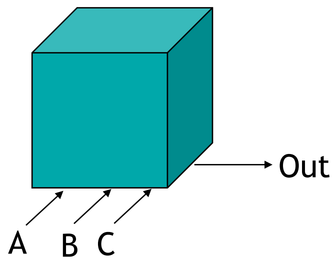
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	NOT X
0	1
1	0

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Black Box (Majority)

- Given a design problem, first determine the function
- Consider the unknown combination circuit a “black box”



Truth Table

“Black Box” Design & Truth Tables

- Given an idea of a desired circuit, implement it
  - Example: Majority with inputs: A, B, C, output: Out

Boolean Formulae

Boolean Algebra

values: 0, 1
variables: A, B, C, . . ., X, Y, Z
operations: NOT, AND, OR, . . .

NOT X is written as $\overline{X}$
X AND Y is written as X & Y, X•Y, X Y
X OR Y is written as X + Y

Deriving Boolean equations from truth tables:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$\text{Sum} = \overline{A}\,B + A\,\overline{B}$

OR'd together *product* terms for each truth table row where the function is 1

if input variable is 0, it appears in complemented form; if 1, it appears uncomplemented

$\text{Carry} = A\,B$

Boolean Algebra

Another example:

A	B	Cin	Sum	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

$\text{Sum} = \overline{A}\,\overline{B}\,\text{Cin} + \overline{A}\,B\,\overline{\text{Cin}} + A\,\overline{B}\,\overline{\text{Cin}} + A\,B\,\text{Cin}$

$\text{Cout} = \overline{A}\,B\,\text{Cin} + A\,\overline{B}\,\text{Cin} + A\,B\,\overline{\text{Cin}} + A\,B\,\text{Cin}$

Two-argument Boolean functions

In

L R
0 0
0 1
1 0
1 1

Two-argument Boolean functions

All possible two input logic gates

In		Out															
L	R	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Two-argument Boolean functions

In Out

L	R	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0 □ □ L < R ≠ □

Two-argument Boolean functions

In Out

L	R	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0 □ □ L < R ≠ □ □ ≠ R < L > □ 0

Two-argument Boolean functions

In		Out															
			and					xor	or	nor	xnor					nand	
			*						+								
L	R	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0 [illegible] [illegible] L < R ≠ [illegible] [illegible] ≠ R < L > [illegible] 0

0 [illegible] [illegible] L < R ≠ [illegible] [illegible] = R ≥ L ≤ [illegible] 1

Boolean Algebra

Reducing the complexity of Boolean equations

Laws of Boolean algebra can be applied to full adder's carry out function to derive the following simplified expression:

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

B Cin

A Cin

A B

Cout = A Cin + B Cin + A B

Verify equivalence with the original Carry Out truth table:

place a 1 in each truth table row where the product term is true

each product term in the above equation covers exactly two rows in the truth table; several rows are "covered" by more than one term

Representations of Boolean Functions

- Boolean Function: $F = \overline{X} + YZ$

Truth Table:

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

Circuit Diagram:

Why Boolean Algebra/Logic Minimization?

Logic Minimization: reduce complexity of the gate level implementation

- reduce number of literals (gate inputs)
- reduce number of gates
- reduce number of levels of gates

fewer inputs implies faster gates in some technologies

fan-ins (number of gate inputs) are limited in some technologies

fewer levels of gates implies reduced signal propagation delays

number of gates (or gate packages) influences manufacturing costs

Basic Boolean Identities:

- $X + 0 =$ $\qquad$ $X * 1 =$
- $X + 1 =$ $\qquad$ $X * 0 =$
- $X + X =$ $\qquad$ $X * X =$
- $X + \overline{X} =$ $\qquad$ $X * \overline{X} =$
- $\overline{\overline{X}} =$

Basic Laws

- Commutative Law:
  $X + Y = Y + X$ $\qquad$ $XY = YX$
- Associative Law:
  $X+(Y+Z) = (X+Y)+Z$ $\qquad$ $X(YZ)=(XY)Z$
- Distributive Law:
  $X(Y+Z) = XY + XZ$ $\qquad$ $X+YZ = (X+Y)(X+Z)$

Boolean Manipulations

- Boolean Function: $F = XYZ + \overline{X}Y + XY\overline{Z}$

Truth Table:

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

Reduce Function:

Advanced Laws

$X+XY =$

$XY + X\overline{Y} =$

$X+\overline{X}Y =$

$X(X+Y) =$

$(X+Y)(X+\overline{Y}) =$

$X(\overline{X}+Y) =$

Boolean Manipulations (cont.)

- Boolean Function: $F = \overline{X}YZ + XZ$

Truth Table:

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

Reduce Function:

Boolean Manipulations (cont.)

- Boolean Function: $F = (X+\overline{Y}+X\overline{Y})(XY+\overline{X}Z+YZ)$

Truth Table:

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

Reduce Function:

DeMorgan's Law

$\overline{(X + Y)} = \overline{X} * \overline{Y}$

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X+Y}$	$\overline{X}\cdot\overline{Y}$
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

$\overline{(X * Y)} = \overline{X} + \overline{Y}$

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X\cdot Y}$	$\overline{X}+\overline{Y}$
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

DeMorgan's Law can be used to convert AND/OR expressions to OR/AND expressions

Example:

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

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

DeMorgan's Law example

If $F = (XY+Z)(\overline{Y}+\overline{X}Z)(X\overline{Y}+\overline{Z})$,

$\overline{F} =$

NAND and NOR Gates

- NAND Gate: NOT(AND(A, B))

X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

- NOR Gate: NOT(OR(A, B))

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

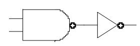
NAND and NOR Gates

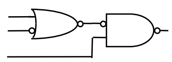
- NAND and NOR gates are universal
  - can implement all the basic gates (AND, OR, NOT)

	NAND	NOR
NOT		
AND		
OR		

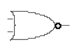
Bubble Manipulation

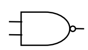
- Bubble Matching





- DeMorgan’s Law





XOR and XNOR Gates

- XOR Gate: Z=1 if X is different from Y

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

- XNOR Gate: Z=1 if X is the same as Y

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

Boolean Equations to Circuit Diagrams

$$F = XYZ + \overline{X}Y + XY\overline{Z}$$

$$F = XY + X(WZ + W\overline{Z})$$