01
Introduction to Digital Logic

ENGR 3410 – Computer Architecture

Mark L. Chang

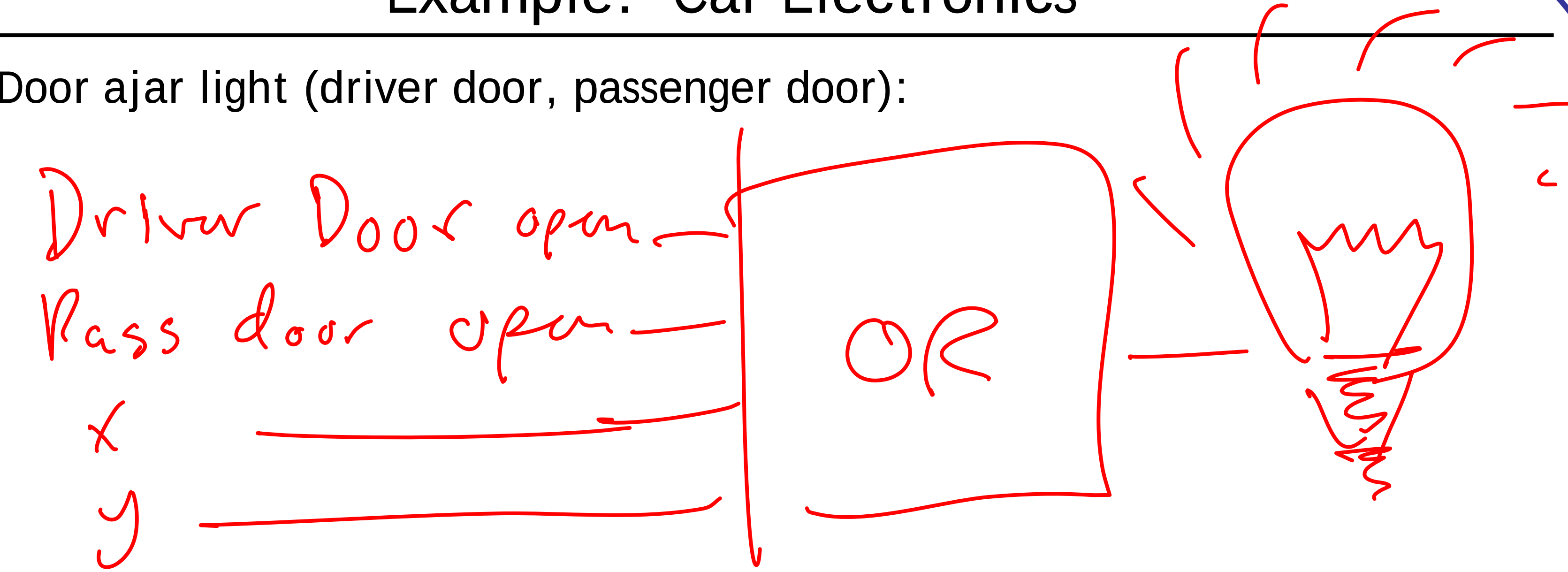
Fall 2006

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)

Example: Car Electronics

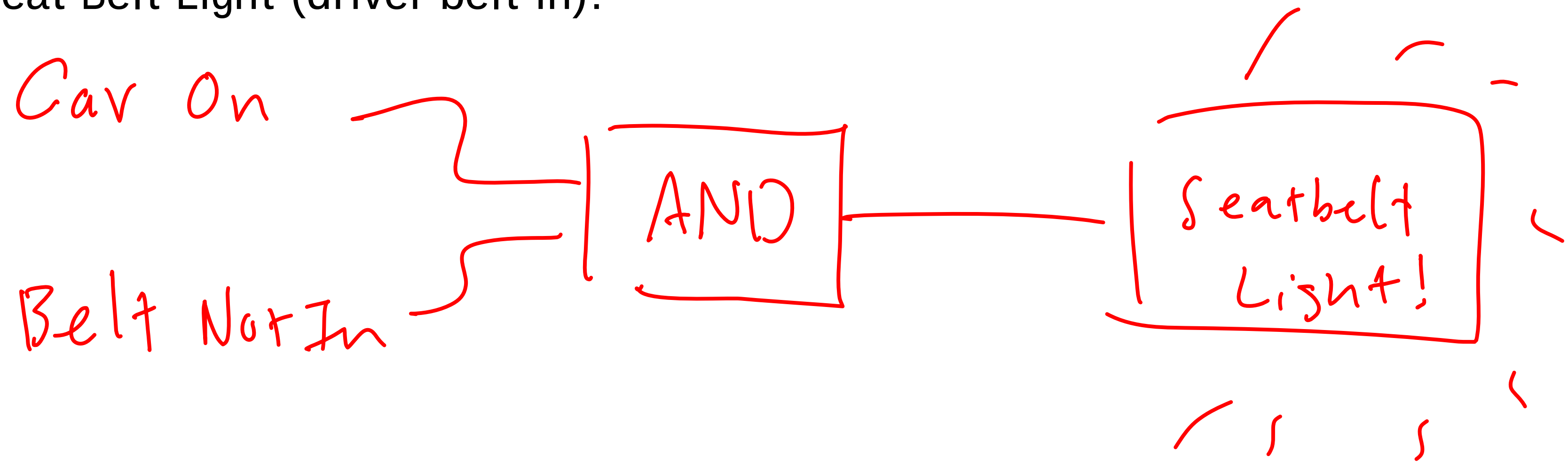
- Door ajar light (driver door, passenger door):



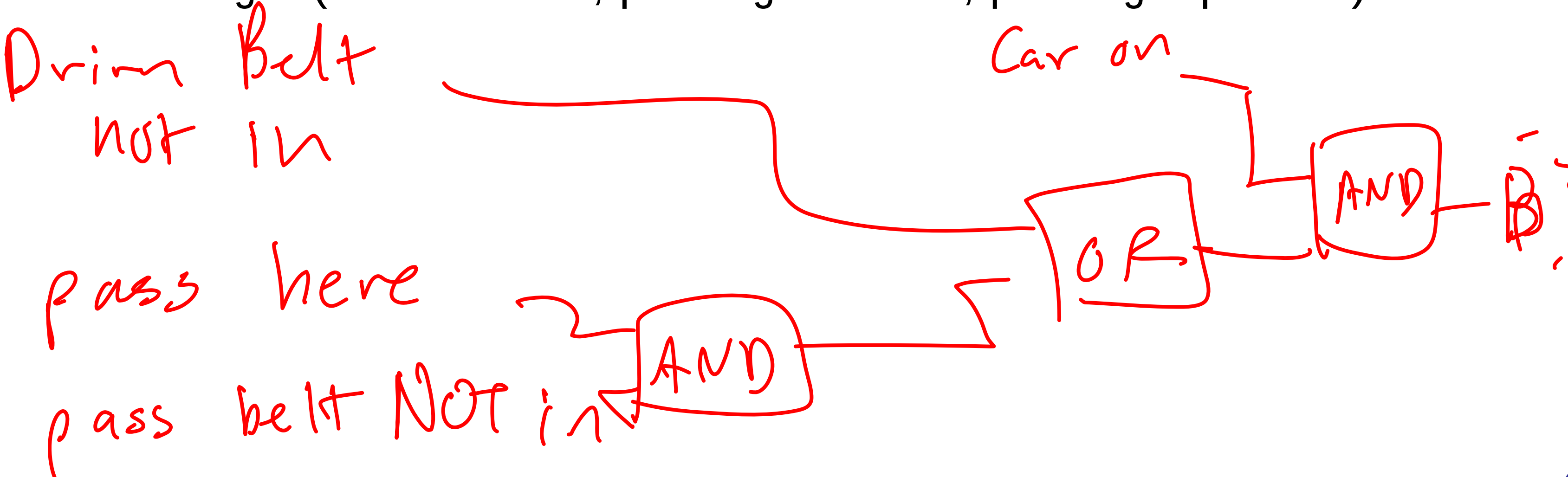
- High-beam indicator (lights, high beam selected):

Example: Car Electronics (cont.)

- Seat Belt Light (driver belt in):

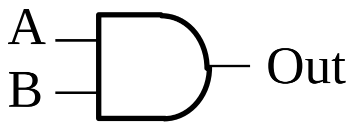


- Seat Belt Light (driver belt in, passenger belt in, passenger present):

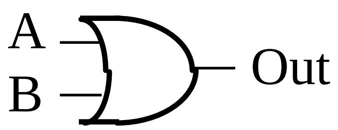


Basic Logic Gates

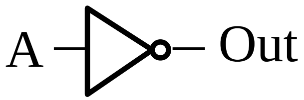
- AND: If A and B are True, then Out is True



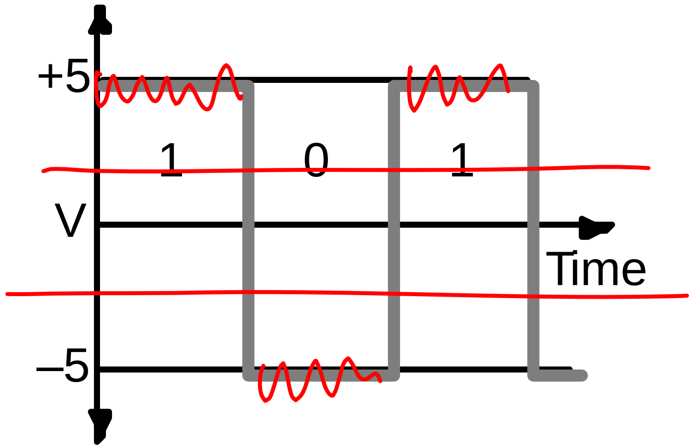
- OR: If A or B is True, or both, then Out is True

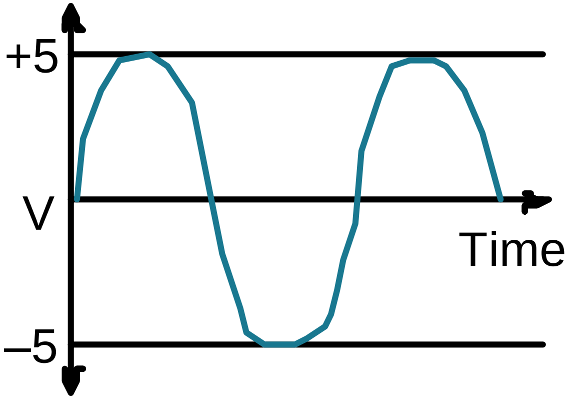


- Inverter (NOT): If A is False, then Out is True



Digital vs. Analog





Digital:
only assumes discrete values

Analog:
values vary over a broad range continuously

Advantages of Digital Circuits

- Analog systems:
  slight error in input yields large error in output

- Digital systems:
  more accurate and reliable
  readily available as self-contained, easy to cascade building blocks

- Computers use digital circuits internally
- Interface circuits (i.e., sensors & actuators) often analog

Binary/Boolean Logic

- *Two discrete values:*
  yes, on, 5 volts, TRUE, "1"
  no, off, 0 volts, FALSE, "0"

- *Advantage of binary systems:*
  rigorous mathematical foundation based on logic

IF the garage door is open
AND the car is running
THEN the car can be backed out of the garage

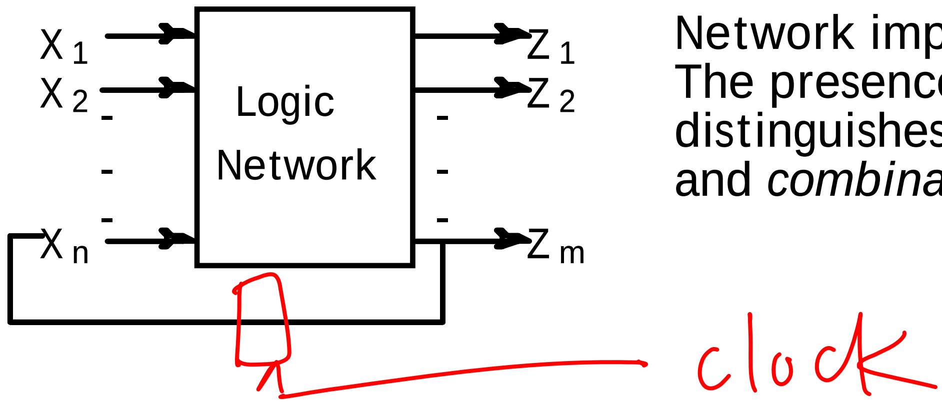
both the door must be open and the car running before I can back out

IF passenger is in the car
AND passenger belt is in
AND driver belt is in
THEN we can turn off the fasten seat belt light

the three preconditions must be true to imply the conclusion

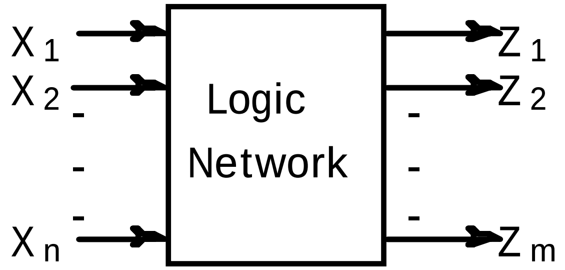
Combinational vs. Sequential Logic

Sequential logic



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

Combinational logic

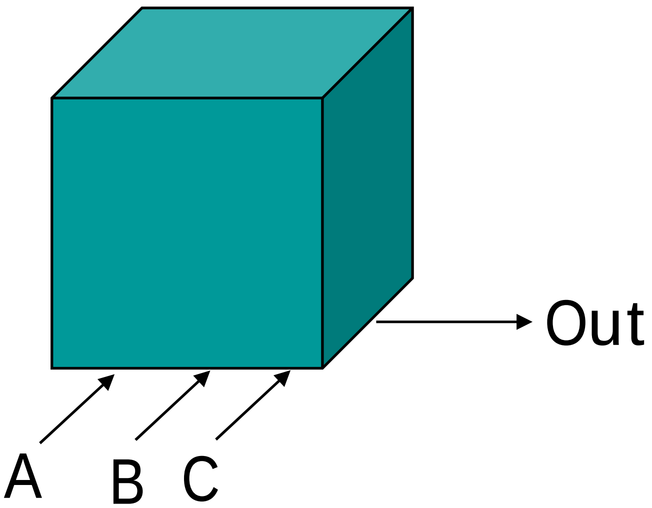


No feedback among inputs and outputs. Outputs are a function of the inputs only.

Black Box (Majority)

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

If ≥ 2/3 are true: false.

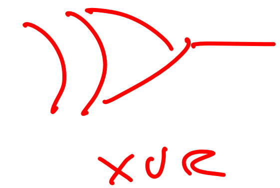


Truth Table M

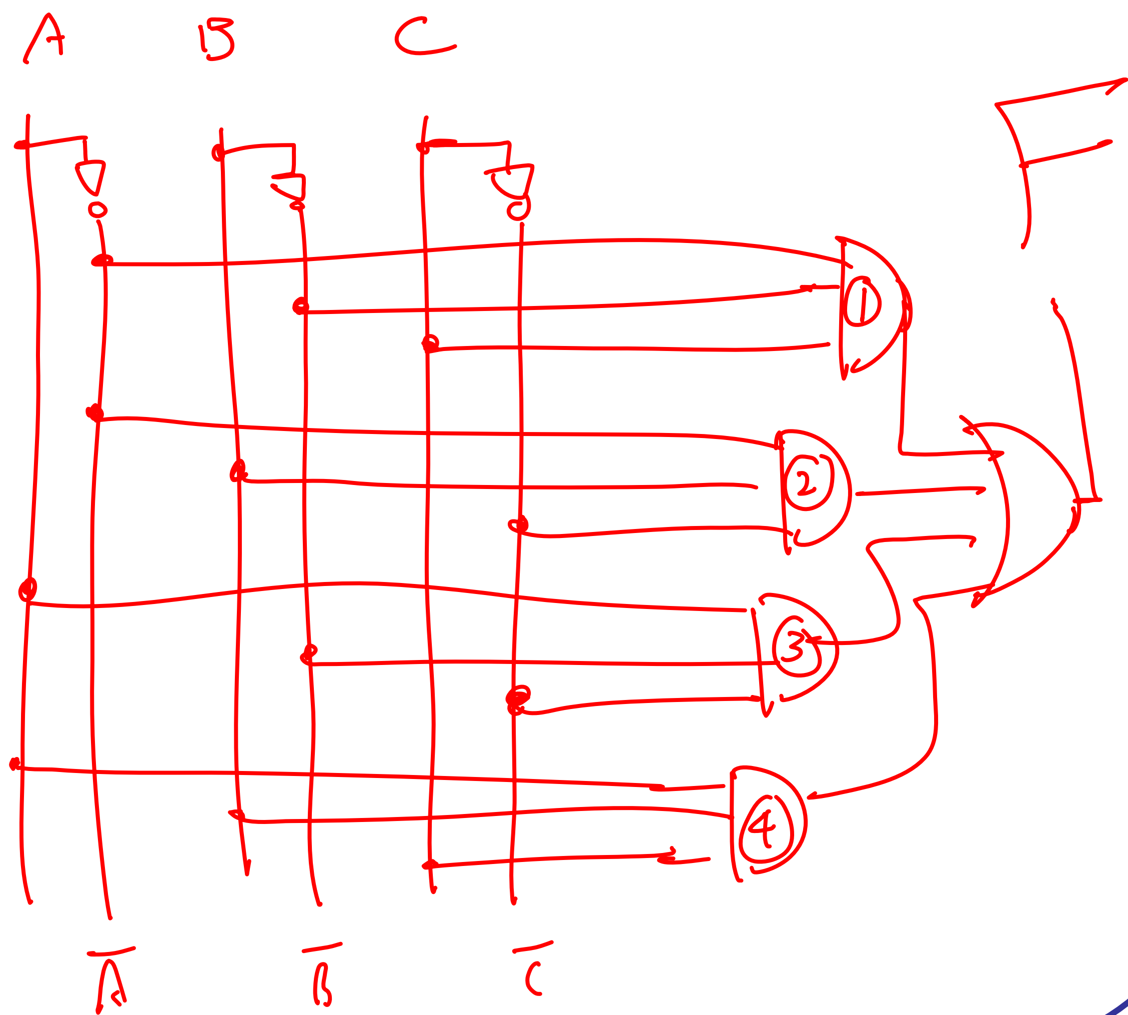
A	B	C	Out	m
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

"Black Box" Design & Truth Tables

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



A	B	C	F	
0	0	0	0	
0	0	1	1	(1)
0	1	0	1	(2)
0	1	1	0	
1	0	0	1	(3)
1	0	1	0	
1	1	0	0	
1	1	1	1	(4)



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

Boolean Equations

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, or sometimes 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

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

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

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

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

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:

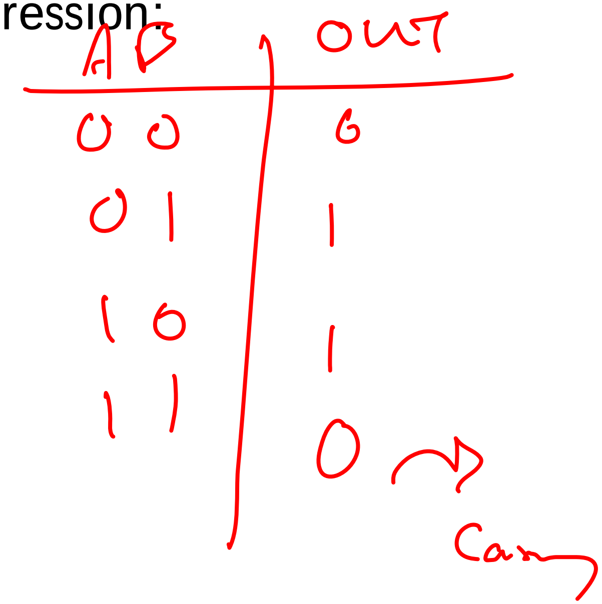
Cout = A Cin + B Cin + A B

B Cin

A Cin

A B

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



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

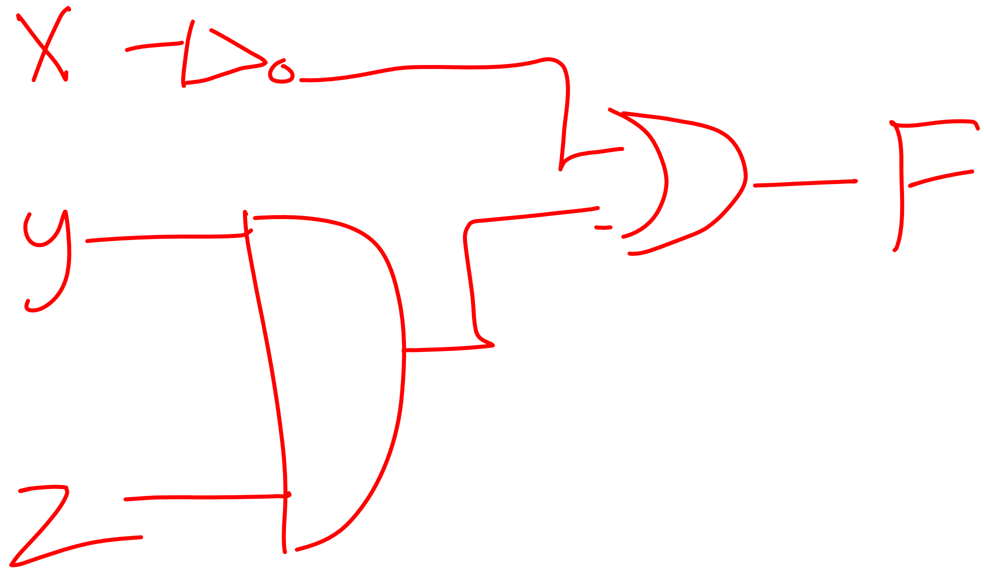
- Boolean Function: $F = \overline{X} + YZ$

$F = \overline{X} + YZ$

Truth Table:

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

Page 57

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

$X * 1 =$ X

$X * 0 =$ 0

$X * X =$ X

$X * \overline{X} =$ 0

Basic Laws

- Commutative Law:

  $X + Y = Y + X$ $\quad\quad$ $XY = YX$

- Associative Law:

  $X+(Y+Z) = (X+Y)+Z$ $\quad\quad$ $X(YZ)=(XY)Z$

- Distributive Law:

  $X(Y+Z) = XY + XZ$ $\quad\quad$ $X+YZ = (X+Y)(X+Z)$

Boolean Manipulations

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

Truth Table:

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

$\overline{x}y$ (rows 010, 011); $xy\overline{z}$ (row 110); xyz (row 111)

Reduce Function:

$$F = xyz + xy\overline{z} + \overline{x}y$$
$$= xy(z+\overline{z}) + \overline{x}y$$
$$= xy + \overline{x}y$$
$$= y(x+\overline{x})$$
$$= y$$

Advanced Laws

- $X+XY =$ x
- $XY + X\overline{Y} =$ x
- $X+\overline{X}Y =$ $x + y$
- $X(X+Y) =$ $xx + xy = x + xy = x$
- $(X+Y)(X+\overline{Y}) =$ $x(x+\overline{y}) + y(x+\overline{y}) = xx + x\overline{y} + xy + y\overline{y}$

  $x\overline{y} + xy = x(y+\overline{y})$

  $= x + x + 0 = x$
- $X(\overline{X}+Y) =$

  $x\overline{x} + xy = 0 + xy$

  $= xy$

Boolean Manipulations (cont.)

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

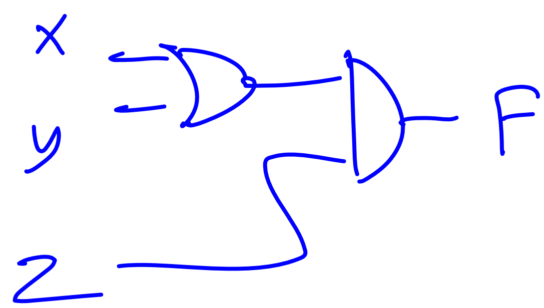
Truth Table:

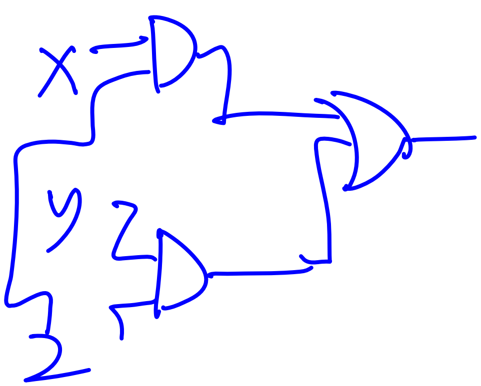
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

XZ

Reduce Function:

$$F = \bar{x}yz + xz$$
$$= z(\bar{x}y + x)$$
$$= z(x + y) = xz + yz$$





Boolean Manipulations (cont.)

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

$\bar{y}(1+x)+x = x+\bar{y}$

Truth Table:

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

Reduce Function:

$F = (x+\bar{y})(xy+\bar{x}z+yz)$

$= xxy + x\bar{x}z + xyz + xy\bar{y} + \bar{x}\bar{y}z + y\bar{y}z$

$= xy + xyz + \bar{x}\bar{y}z$

$= xy(1+z) + \bar{x}\bar{y}z$

$= \boxed{xy + \bar{x}\bar{y}z}$

$\overline{XY} \neq \bar{X}\bar{Y}$

DeMorgan's Law

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

NOR

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

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

NAND

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

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

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

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

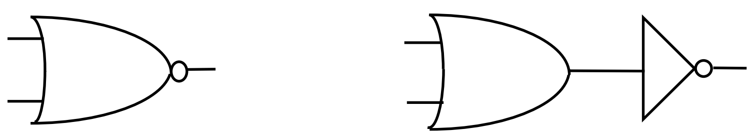
$$= (\overline{X}+\overline{Y})(\overline{Z}) + Y(X+\overline{Z}) + (\overline{X}+Y)Z$$

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

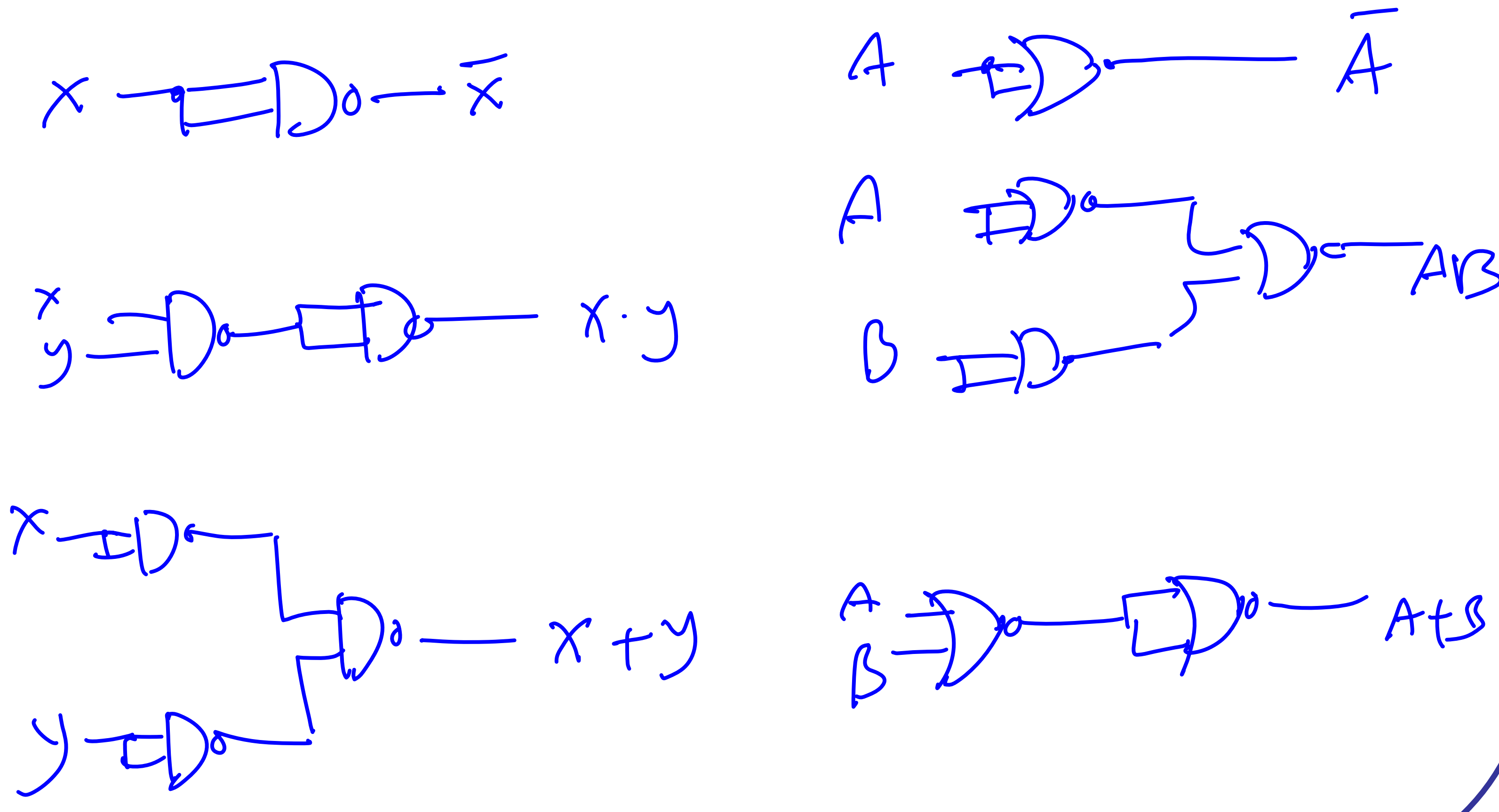
- NAND and NOR gates are universal
  - can implement all the basic gates (AND, OR, NOT)

NAND | NOR

NOT

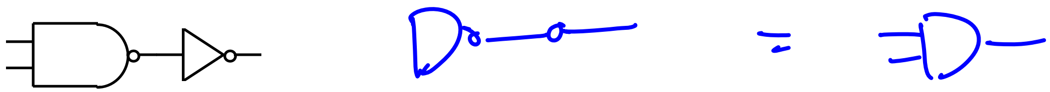
AND

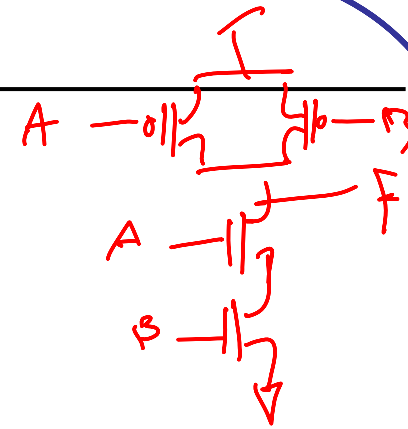
OR

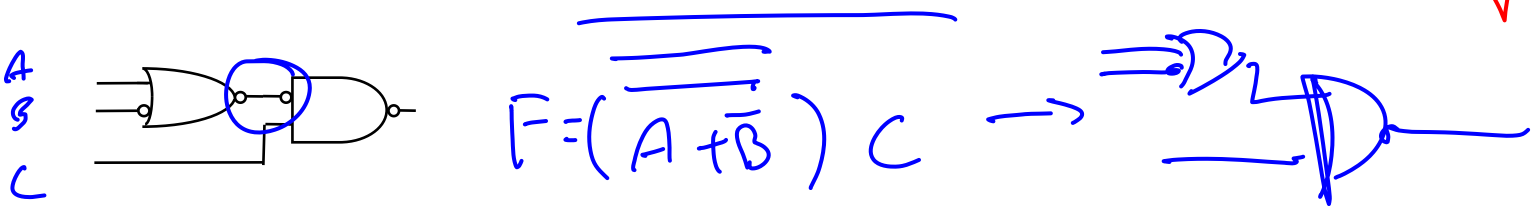


Bubble Manipulation

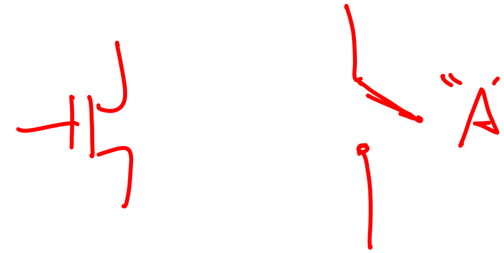
- Bubble Matching

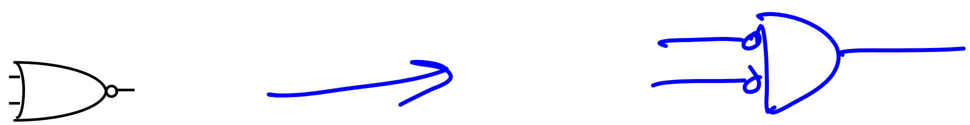


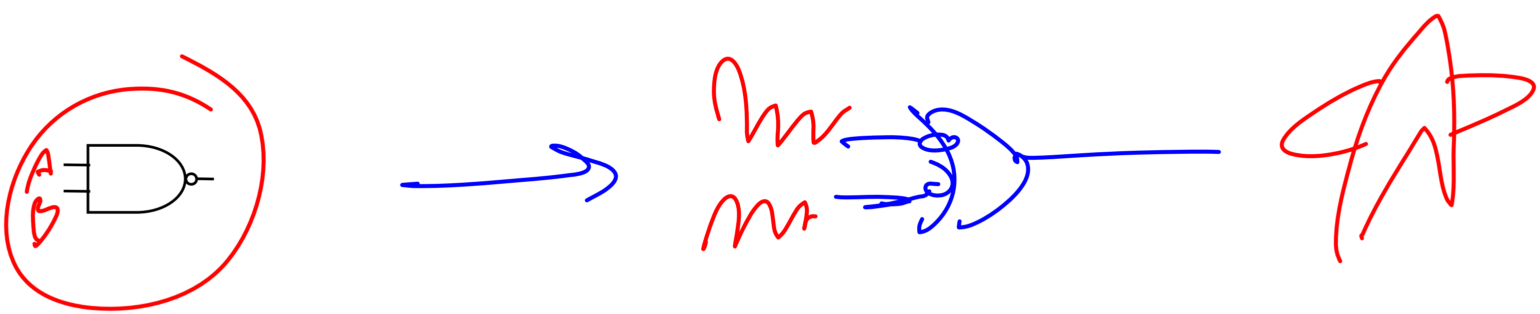




- DeMorgan’s Law

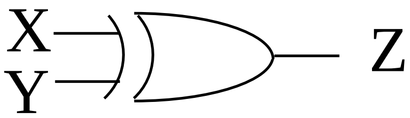






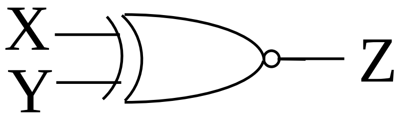
XOR and XNOR Gates

- 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

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

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