Introduction to Digital Logic

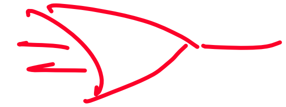
Motivation

Electronics an increasing part of our lives

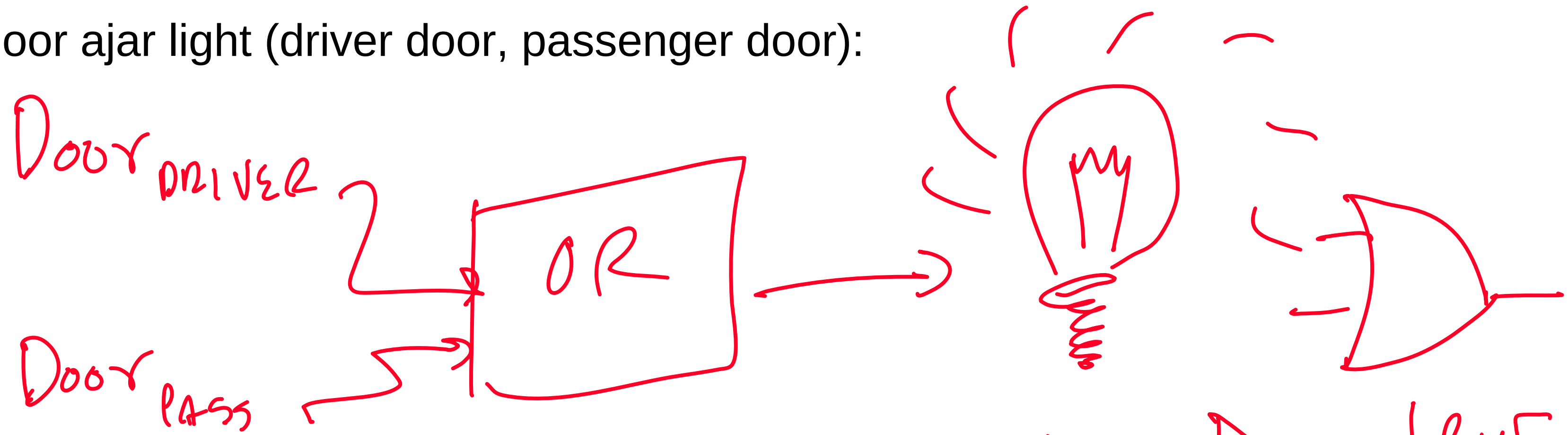
- Computers & the Internet
- Car electronics
- Robots
- Electrical Appliances
- Telephones

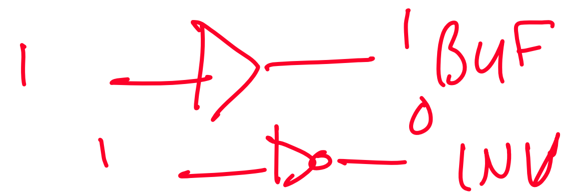
Class is an exercise in digital logic design & implementation

Example: Car Electronics

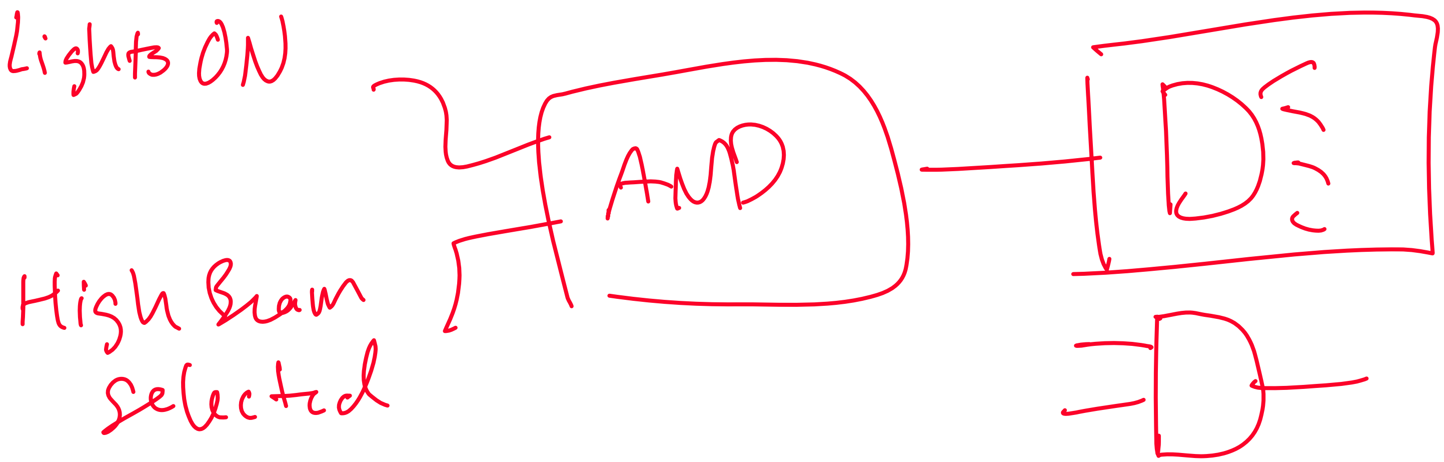


Door ajar light (driver door, passenger door):



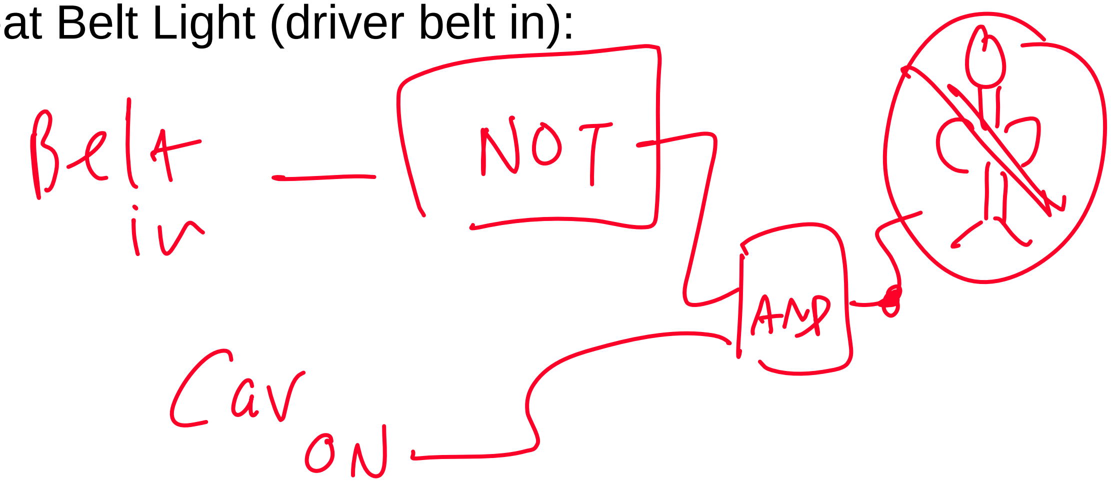


High-beam indicator (lights, high beam selected):

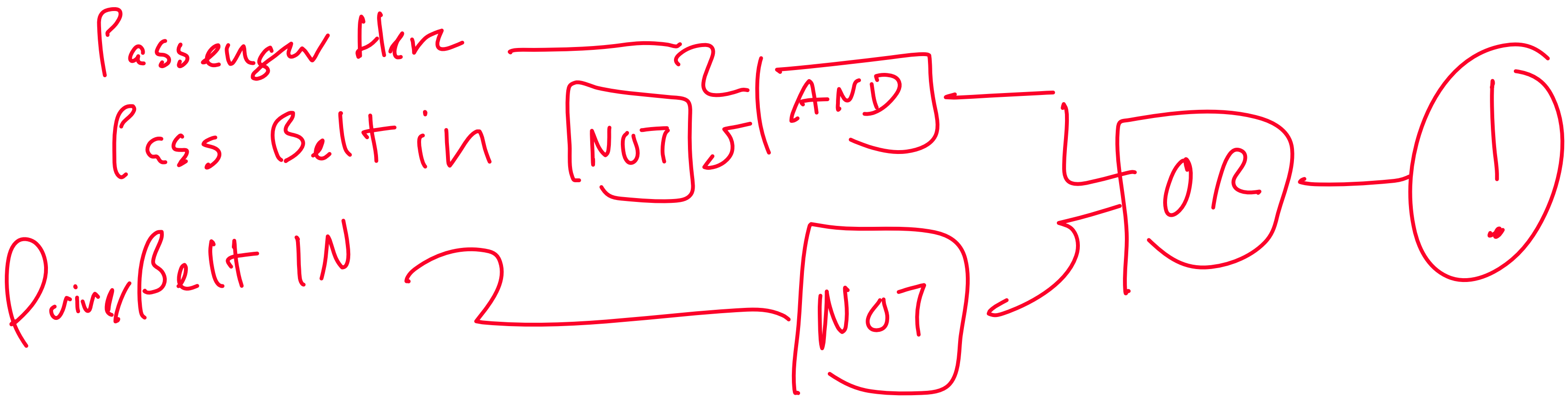


Example: Car Electronics (cont.)

Seat Belt Light (driver belt in):

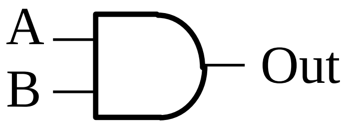


Seat Belt Light (driver belt in, passenger belt in, passenger present):



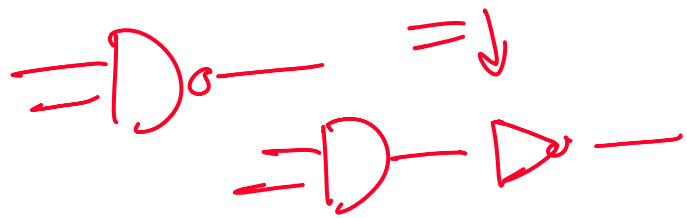
Basic Logic Gates

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

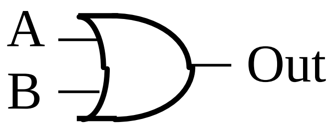


A B	OUT
0 0	0
0 1	0
1 0	0
1 1	1

NAND

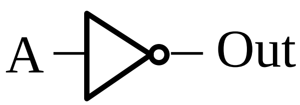


OR: If A or B is True, or both, then Out is True



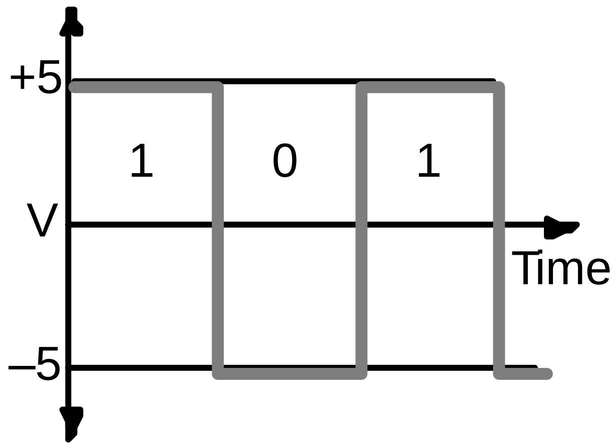
A B	OUT
0 0	0
0 1	1
1 0	1
1 1	1

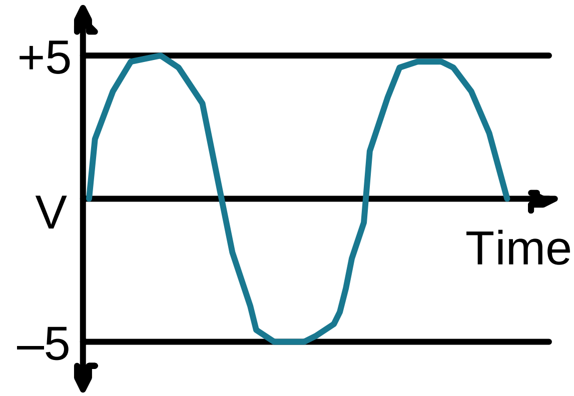
Inverter (NOT): If A is False, then Out is True



A	OUT
0	1
1	0

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

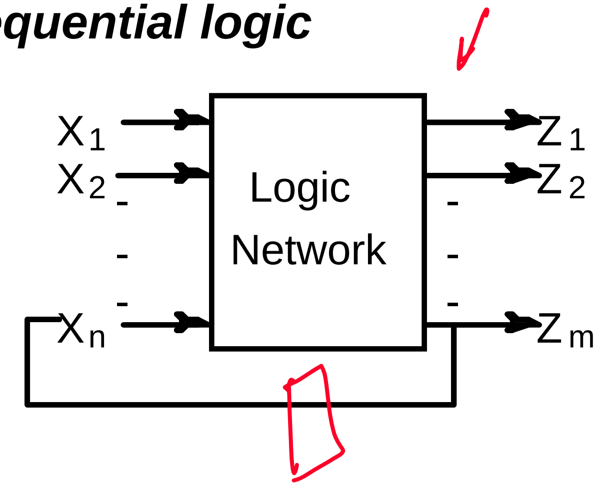
both the door must be open and the car running before I can back out

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

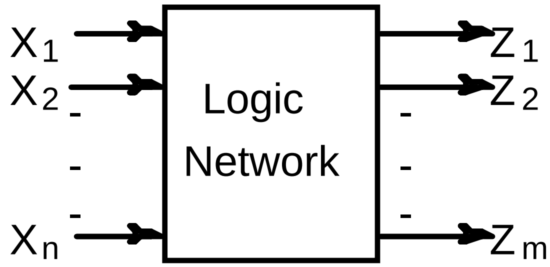
Combinational vs. Sequential Logic

Sequential logic



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

Combinational logic

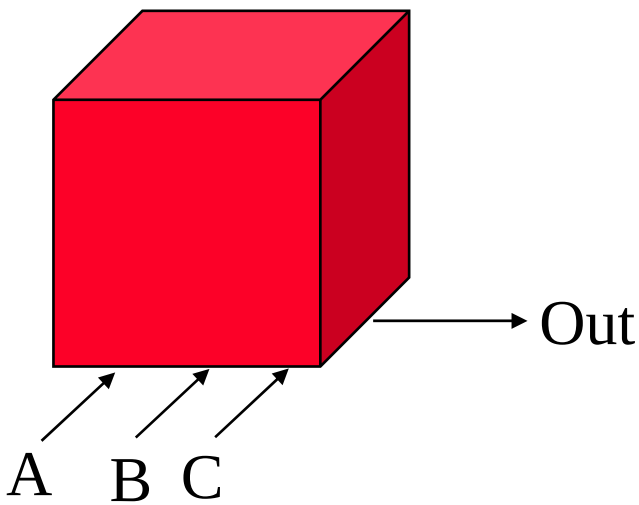


No feedback among inputs and outputs. Outputs are a function of the inputs only.

Black Box (Majority)

Given a design problem, first determine the function

Consider the unknown combination circuit a “black box”



Truth Table

A	B	C	Out
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

“Black Box” Design & Truth Tables

0 = even

NOT

$A \cdot B$ = AND

$A + B$ = OR

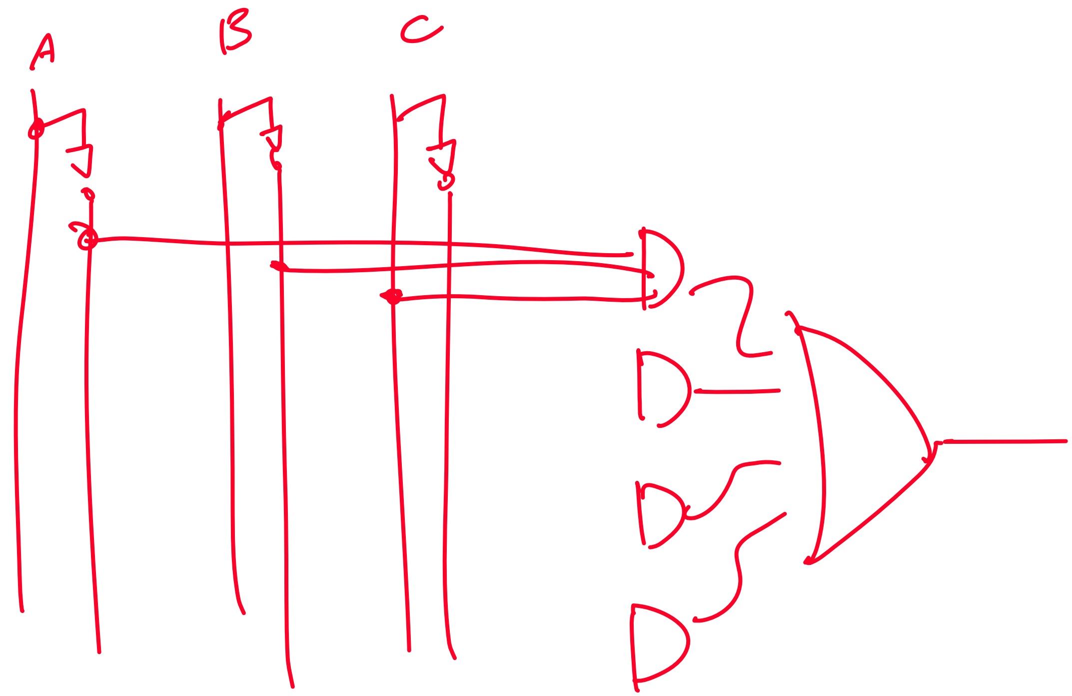
Given an idea of a desired circuit, implement it

Example: Odd parity - inputs: A, B, C, output: Out

ERIC G's

$OUT = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

A	B	C	OUT
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



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

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

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

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

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

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

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

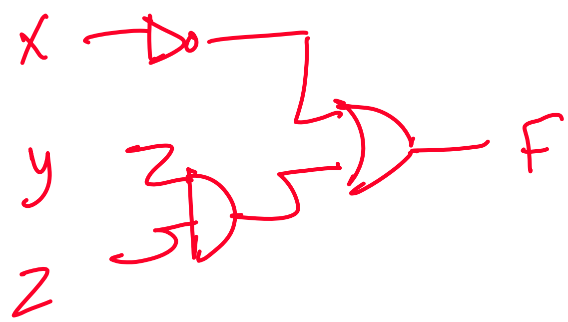
Representations of Boolean Functions

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

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:



$f = \overline{x}\,\overline{y}\,\overline{z} + \overline{x}\,\overline{y}z + \overline{x}y\overline{z} + \overline{x}yz + xyz$

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

$X + 0 = X$

$X * 1 = X$

$X + 1 = 1$

$X * 0 = 0$

$X + X = X$

$X * X = X$

$X + \overline{X} = 1$

$X * \overline{X} = 0$

$\overline{\overline{X}} = X$

Basic Laws

Commutative Law:

$X + Y = Y + X$ $XY = YX$

Associative Law:

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

Distributive Law:

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

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

Reduce Function:

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

$= Y(XZ + \bar{X} + X\bar{Z})$

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

$= Y(X(1) + \bar{X})$

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

$= Y(1)$

$= \boxed{Y}$

Advanced Laws

- $X+XY =$ $X(1+Y) = X(1) = X$
- $XY + X\overline{Y} =$ $X(Y+\overline{Y}) = X(1) = X$
- $X+\overline{X}Y =$ $X+Y$
- $X(X+Y) =$ $XX + XY = X + XY = X$
- $(X+Y)(X+\overline{Y}) =$ $X(X+Y) + \overline{Y}(X+Y) = (XX + XY) + X\overline{Y} + \overline{Y}Y =$
  $= X + X\overline{Y} + \overline{Y}Y$
  $= X + X\overline{Y}$
  $= X(1+\overline{Y})$
  $= X$
- $X(\overline{X}+Y) =$
  $X\overline{X} + XY$
  $= 0 + XY$
  $= XY$

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

Reduce Function:

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

$= z(x + y)$

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	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:

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

$+ \overline{X}\overline{Y}Z + Y\overline{Y}Z$

$+ X\overline{Y}XY + X\overline{Y}\overline{X}Z + X\overline{Y}YZ$

$= XXY + XYZ + \overline{X}\overline{Y}Z$

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

$= XY(1+Z) + \overline{X}\overline{Y}Z$

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

DeMorgan’s Law

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

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

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

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

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

AND

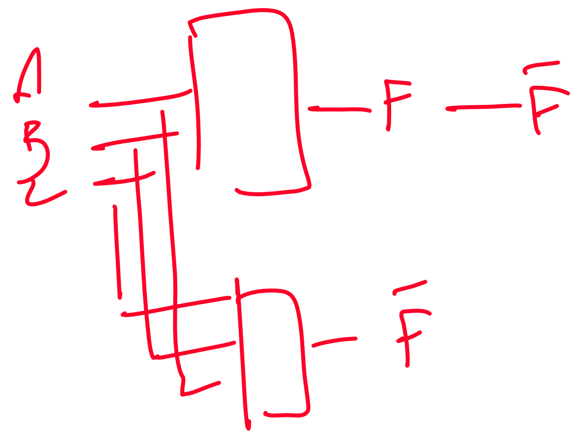
$$\overline{F} = \overline{(XY+Z)(\bar{Y}+\bar{X}Z)(X\bar{Y}+\bar{Z})}$$

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

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

$$= (\bar{X}+\bar{Y})\bar{\bar{Z}} + Y(\bar{\bar{X}}+\bar{Z}) + (\bar{X}+\bar{\bar{Y}}) \cdot Z$$

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

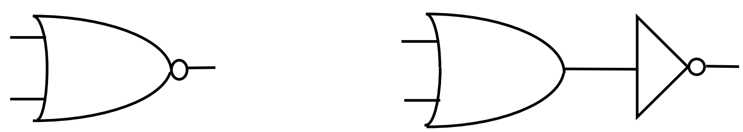


NAND and NOR Gates

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

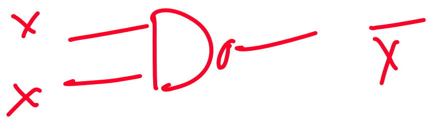
NAND and NOR Gates

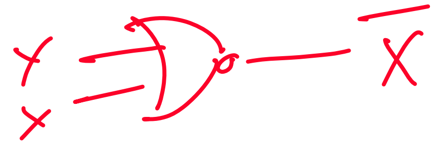
NAND and NOR gates are universal

can implement all the basic gates (AND, OR, NOT)

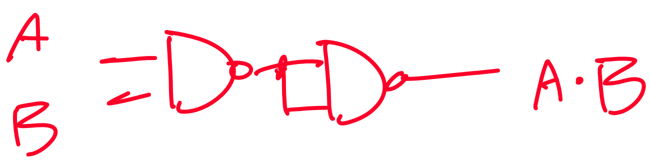
NAND | NOR

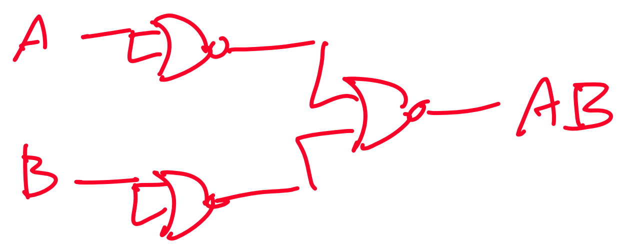
NOT



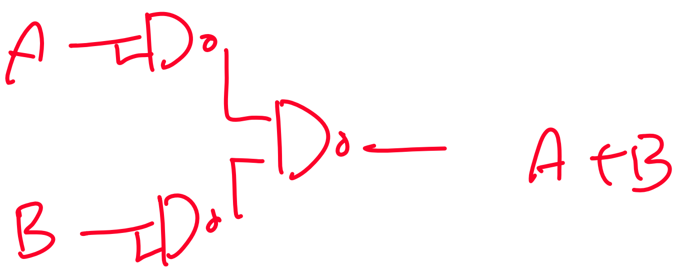


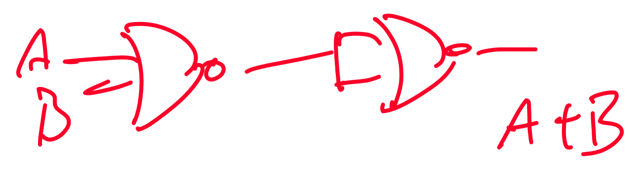
AND





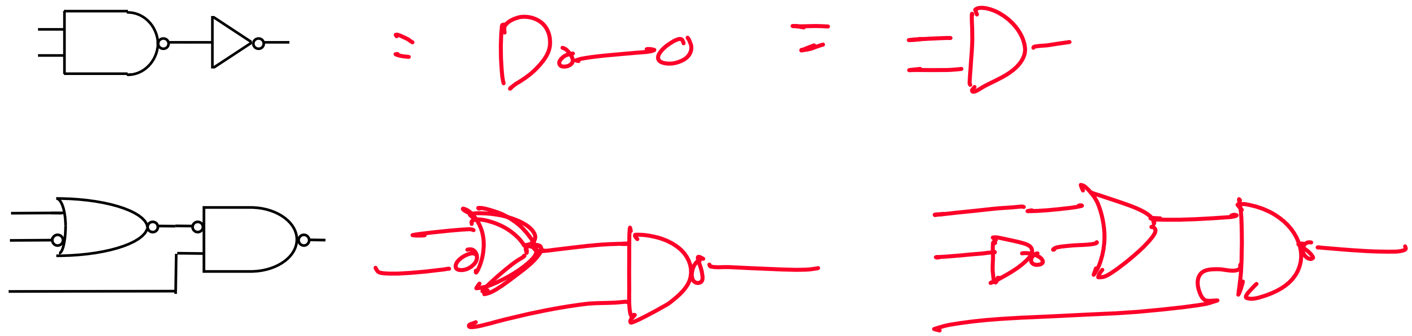
OR



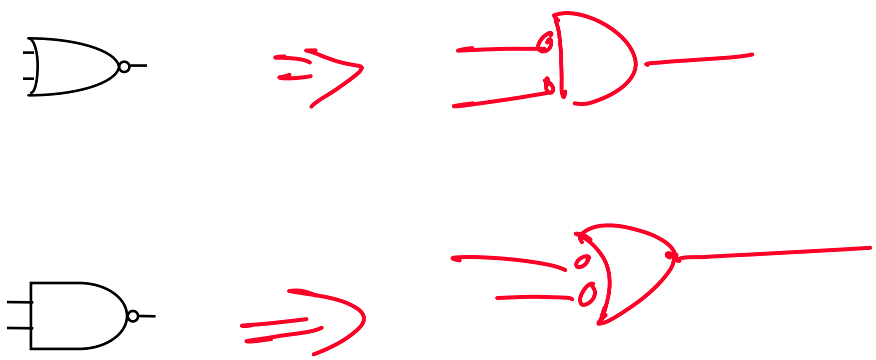


Bubble Manipulation

Bubble Matching

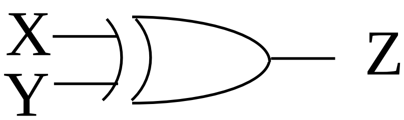


DeMorgan’s Law



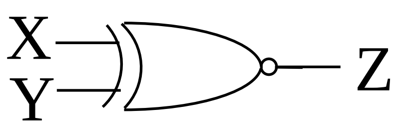
XOR and XNOR Gates

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

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

$$= y(xz + \overline{x} + x\overline{z})$$

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

$$= y$$

$$y \longrightarrow F$$

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

$$= xy + xw$$

$$= x(y + w)$$