RUTGERS

CS211 Computer Architecture

Fall 2020

Recitation 13

Today

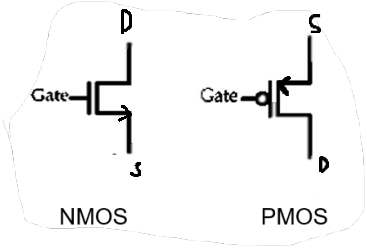
- Transistors overview
- Gates & Truth Tables
- DeMorgan's Law
- Functional Completeness
  - NAND
  - NOR
- Boolean Expressions
  - Drawing a circuit

Transistors

- Transistors are the basic units that build gates
  - Gates are used to implement logic
- They are the used to build computers
- 2 types:
  - NMOS
  - PMOS

* MOS = metal oxide semiconductor

Transistors

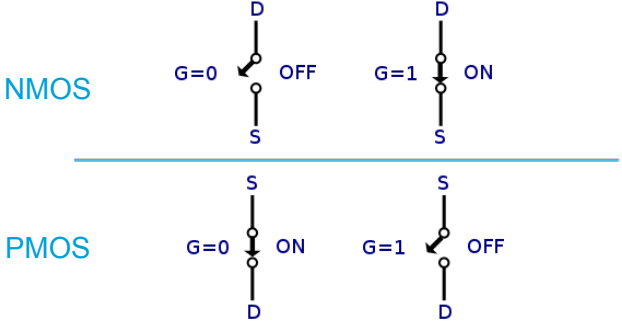


voltage applied to gate

- > 0, ON, acts as short circuit
- < 0, OFF, acts as open circuit

- < 0, ON, acts as short circuit
- > 0, OFF, acts as open circuit

Transistors



Transistors

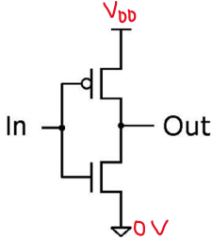
- A CMOS circuit is made up of both NMOS and PMOS transistors
  - N-type is used to pull output down to ground
  - P-type is used to pull output up towards a voltage at the source
- CMOS circuits are preferred to implement gates such as AND, OR, NOT, etc
  - Low static power dissipation
  - Provides pure logic “1” and “0”

Transistors

- The design of a transistor circuit effects input and output of the circuit
  - Input and outputs are voltages
- The voltages correspond to either logic “0” or “1” in digital logic
  - Certain threshold of voltage values will be accepted as “0” or “1” (aka “low” or “high”)

CMOS Inverter Example

- NOT gate using a CMOS inverter
  - Consists of 1 PMOS and NMOS
    - PMOS connected to some voltage supply, let's say VDD
    - NMOS connected to ground (0 V)
    - Input connected to gate of both



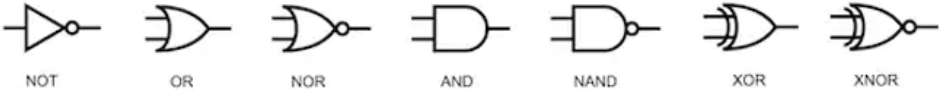
Why?

If input voltage = "high" □ PMOS OFF, NMOS ON
□ output connected to GND = "low"

If input voltage = "low" □ PMOS ON, NMOS OFF
□ output connected to VDD = "high"

Input	Output
"low" / 0 / 0 V	"high" / 1 / 2.9 V
"high" / 1 / 2.9 V	"low" / 0 / 0 V

Gates



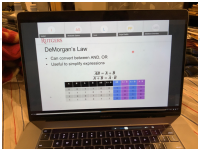
Gates – Truth Tables

A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

xor

A	B	$A + B$	$\overline{A + B}$	$A \oplus B$
0	0	0	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	0	0

DeMorgan's Law



- Can convert between AND,
- Useful to simplify expressior

0	0	1	1	0	0	1	1	1	1
0	1	1	0	0	1	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	1	0	0	1	1	0	0	0	0

$\overline{B}$:

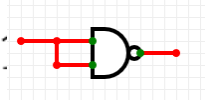
Functional Completeness

- What does it mean?
  - If a gate or set of gates is *functionally complete*, then any Boolean expression can be "realized" using only that set
  - Efficient to use the same type of gate to minimize costs and propagation delays
  - Ex: {AND, OR, NOT} – basic set of gates

- **NOR** and **NAND** are both functionally complete
  - NOT, AND, & OR can be implemented using only NAND or only NOR gates
  - How?

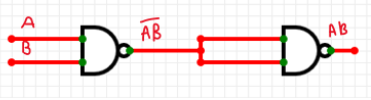
Functional Completeness - NAND

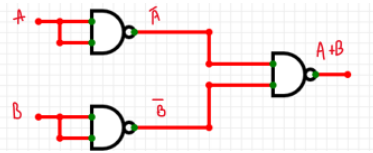
- NOT
  - To invert an input, just N
  - $\overline{00} = 1$
  - $\overline{11} = 0$



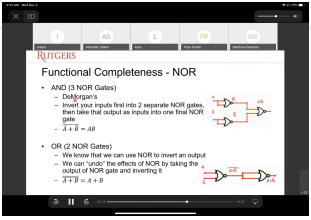
Functional Completeness - NAND

- AND (2 NAND Ga
  - We know that we ca
  - We can “undo” the ef output of NAND gate
  - $\overline{\overline{AB}} = AB$

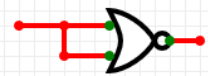




Functional Completeness - NOR



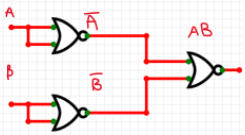
- NOT
  - To in[illegible]ut, just NOR it
  - $\overline{0+0}$

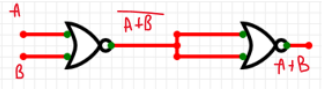


  - $\overline{1+1} = 0$

Functional Completeness - NOR

- AND (3 NOR Gates)
  - DeMorgan's
  - Invert your inputs firs
    then take that output
    gate

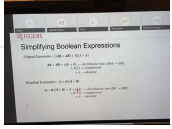




Boolean Expressions

- Convert this expres
  diagram

$$((A\bar{B} + A\bar{B})$$



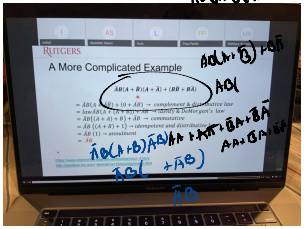
Simplifying Boolean Expressions

- Original Expression - $\big((\boldsymbol{AB} + \boldsymbol{A\overline{B}}) + B\big)(\bar{A}$

$$\boldsymbol{AB} + \boldsymbol{A\overline{B}} = A(B + \bar{B}) \rightarrow$$
$$= A(1) \rightarrow$$
$$= A$$

Simplified Expression - $(\boldsymbol{A} + \boldsymbol{B})(\overline{\boldsymbol{A}} + \boldsymbol{B})$

A More Complicated Example

- 

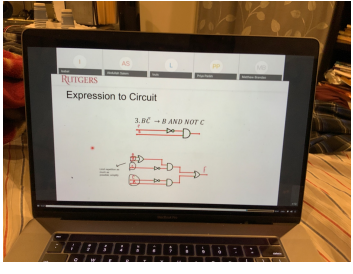
$$\bar{A}B(A+\bar{B})(A-$$

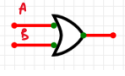
$$= \bar{A}B(A+\bar{B})(1) + (0+$$

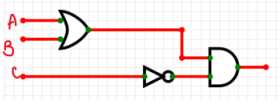
$$= \bar{A}B(A+\bar{B}) + \bar{A}B \rightarrow i$$

Expression to Circuit

-





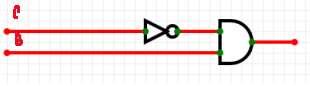


$$\mathbf{F} = (\boldsymbol{A} +$$

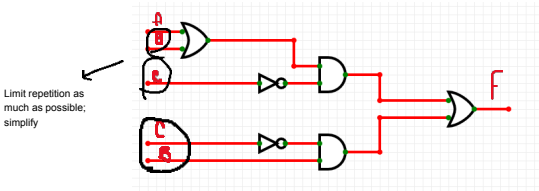
1. (

Expression to Circuit

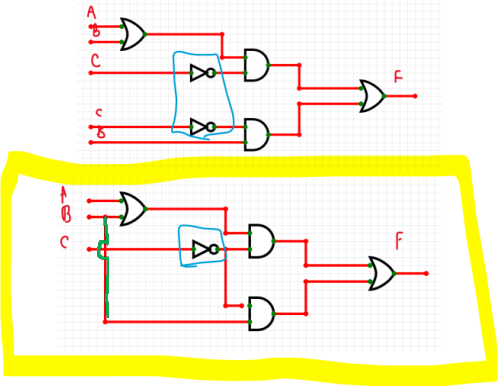
•



$3. B\bar{C} \rightarrow B$

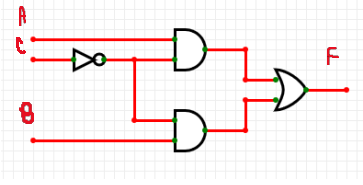


Expression to Circuit



Simplifying Circuits

•

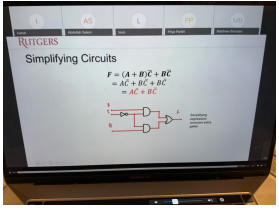


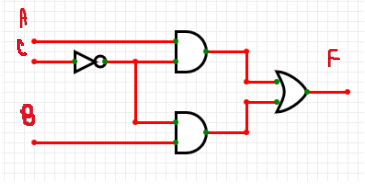
$$F = (A +$$
$$= A\bar{C} +$$
$$= A($$

Simplifying expression removes extra gates

Simplifying Circuits

•





$$F = (A +$$
$$= A\bar{C} +$$
$$= A($$

Simplifying expression removes extra gates