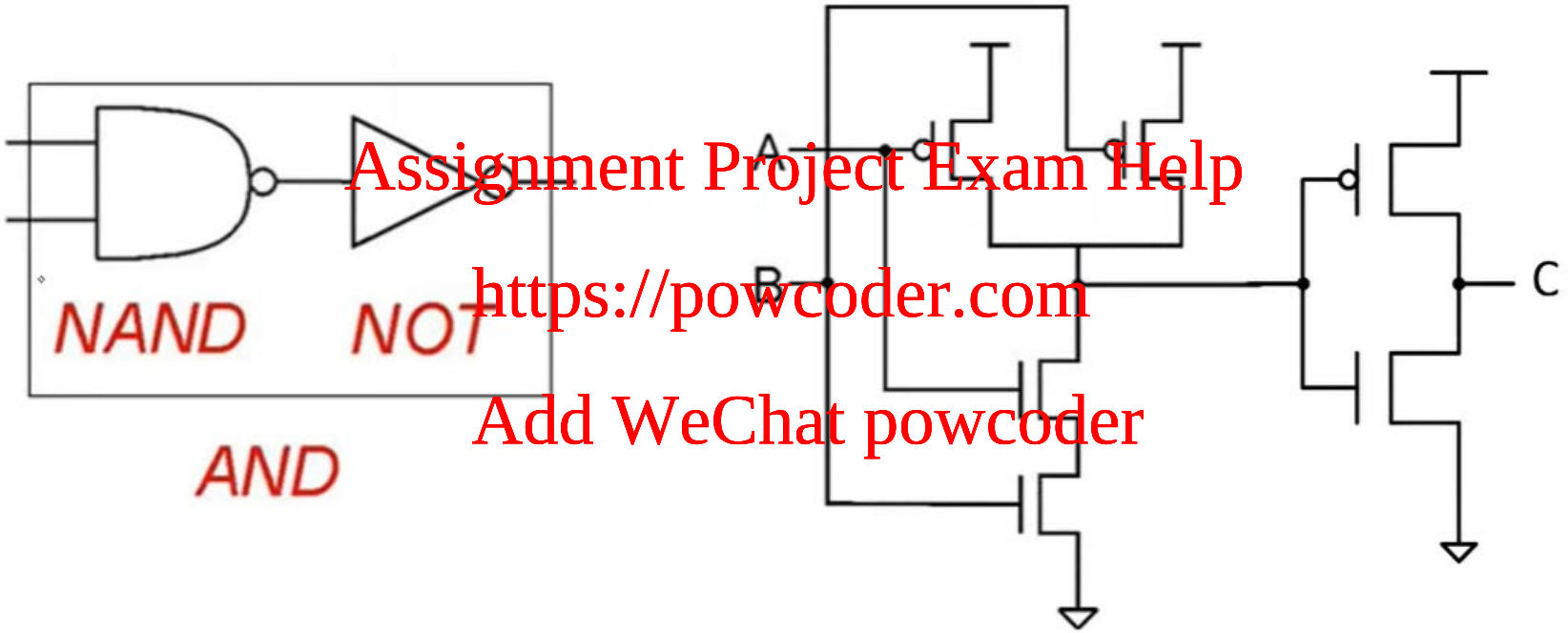
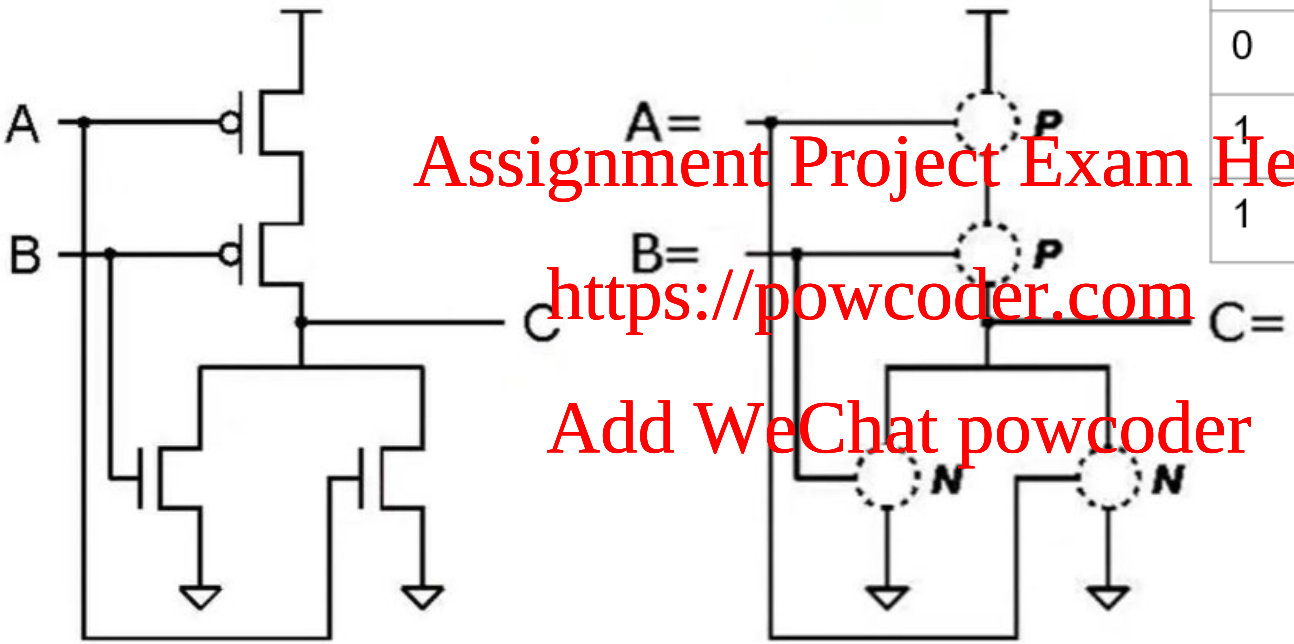
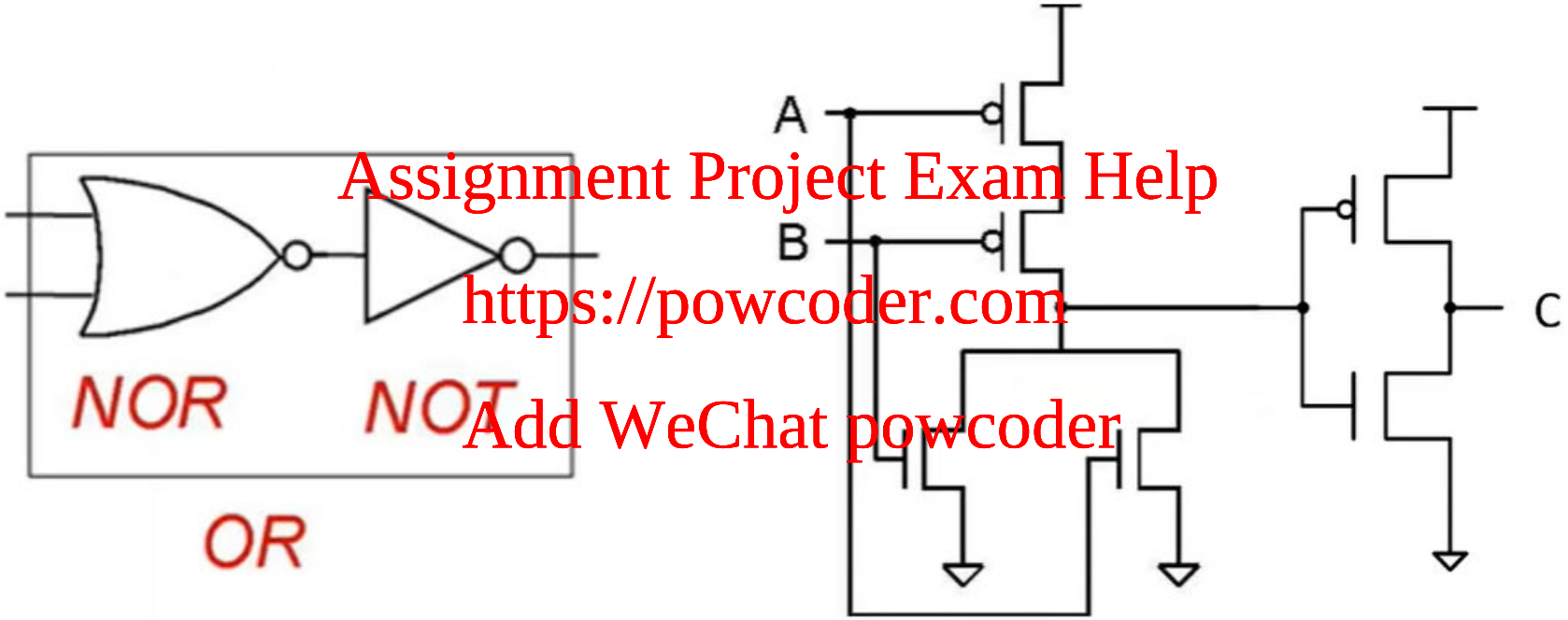
Digital Logic: Boolean Algebra and Gates

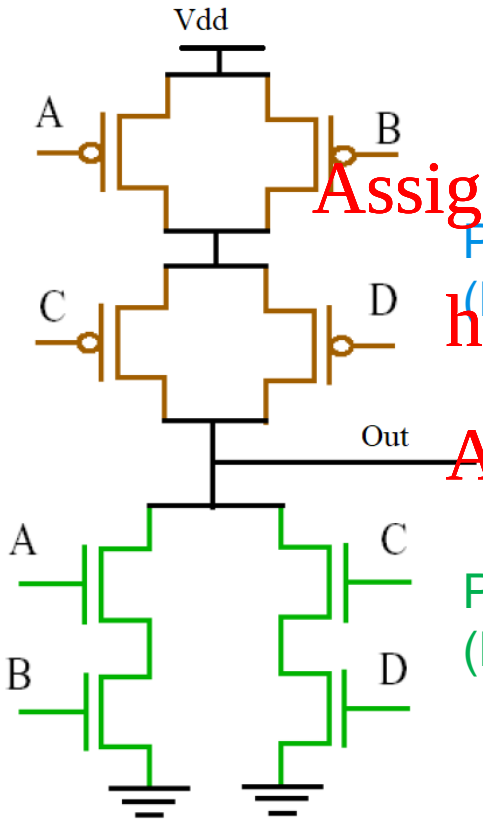




A	B	C
0	0	1
0	1	0
1	0	0
1	1	0



A general CMOS circuit



Pull Up Network (PUN)

Pull Down Network (PDN)

Q. How to determine Out without solving for entire truth table???

A. Follow Output path through either the PUN (towards Vdd) or the PDN (towards gnd)

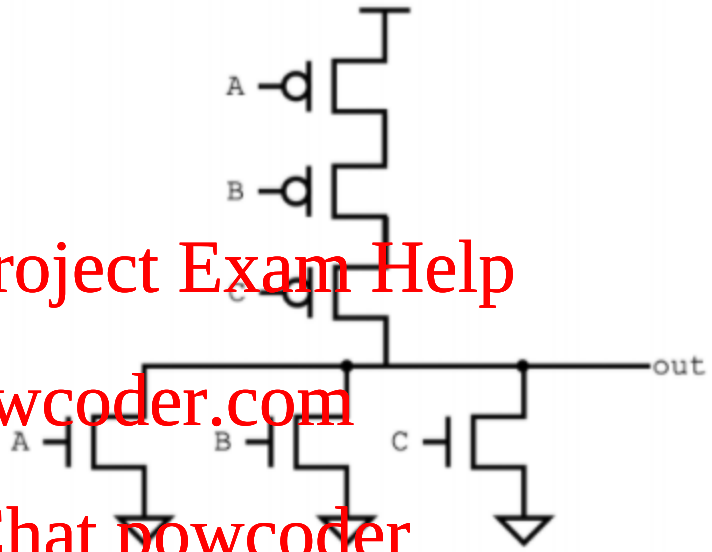
PDN approach: Under what conditions does Out have a direct path to gnd??

Out = 0, (when A AND B ON) OR (when C AND D ON)

$$Out=(AB+CD)'$$

Combinational Logic

7. The following CMOS diagram represents which logic gate?



- ○ A. NAND
- ○ B. NOR
- ○ C. AND
- ○ D. OR
- ○ E. Answer not listed.

Axioms of Boolean Algebra (CMPE 100!)

- Algebra : study of mathematical symbols and the rules for manipulating these symbols
- Different types of algebra exists depending on what mathematical space you are working with ; e.g. real algebra, vector algebra, linear algebra, Boolean algebra
- The three important tenets of algebra: **symbols, rules/operators, constants**
- We are already pretty familiar with the rules of real algebra, i.e. working with real numbers e.g. x ,y,z,k are real numbers: x(y+z-k) = xy +xz – xk
- **Above example used distributive law and "plus", "subtract", "multiply" operators.**
- **Boolean algebra is the only algebra understood by a computer. The computer physically solves problems in Boolean algebra which are based on high level problems (in various types of algebra)**
- **Two constants: 1,0; Three fundamental operators: AND (&), OR(+), NOT**

Axioms of Boolean Algebra (CMPE 100!)

- 0 · 0 = 0
- 1 + 1 = 1
- 1 · 1 = 1
- 0 + 0 = 0
- 0 · 1 = 1 · 0 = 0
- 1 + 0 = 0 + 1 = 1
- if x = 0 then x' = 1
- if x = 1 then x' = 0

Let's have a closer look at what exactly those transistors were computing in the previous slides!

Single-Variable Theorems (CMPE 100!)

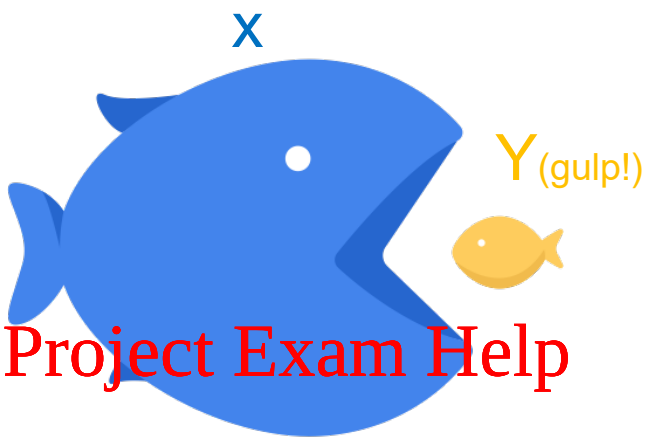
- $x \cdot 0 = 0$
- $x + 1 = 1$
- $x \cdot 1 = x$
- $x + 0 = x$
- $x \cdot x = x$
- $x + x = x$
- $x \cdot x' = 0$
- $x + x' = 1$
- $(x')' = x$

Properties of Boolean Algebra (CMPE 100!)

- Commutative
  - $x \cdot y = y \cdot x$
  - $x + y = y + x$
- Associative
  - $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
  - $x + (y + z) = (x + y) + z$
- Distributive
  - $x \cdot (y + z) = x \cdot y + x \cdot z$
  - $x + y \cdot z = (x + y) \cdot (x + z)$

Properties of Boolean Algebra (CMPE 100!)

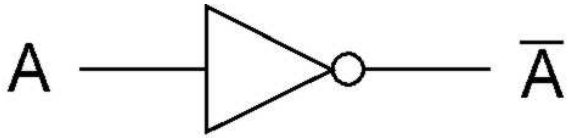
- Absorption
  - $x + x \cdot y = x$
  - $x \cdot (x + y) = x$
- Combining
  - $x \cdot y + x \cdot y' = x$
  - $(x + y) \cdot (x + y') = x$



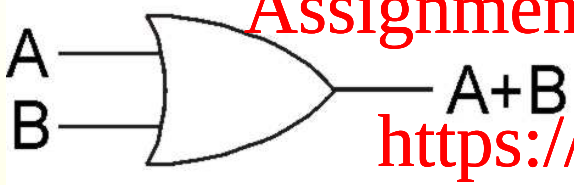
Properties of Boolean Algebra (CMPE 100!)

- De Morgan's Laws
  - $(x \cdot y)' = x' + y'$
  - $(x + y)' = x'y'$
- Other
  - $x + x' \cdot y = x + y$
  - $x \cdot (x' + y) = x \cdot y$

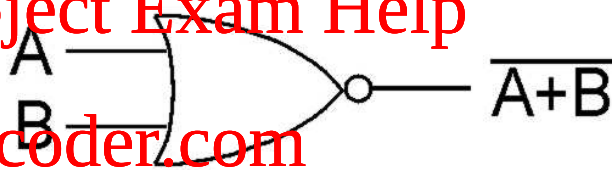
Basic Logic Gates



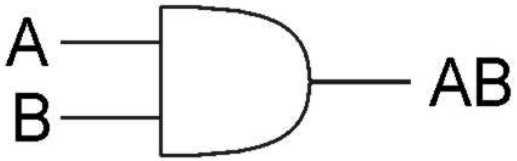
NOT



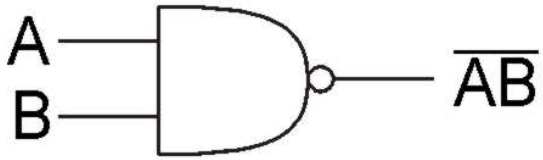
OR



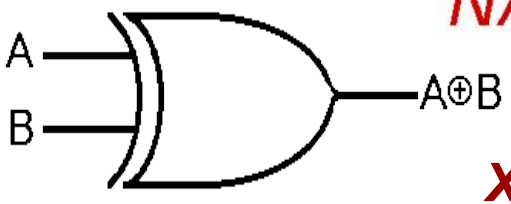
NOR



AND

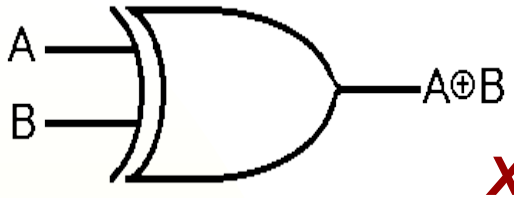


NAND



XOR

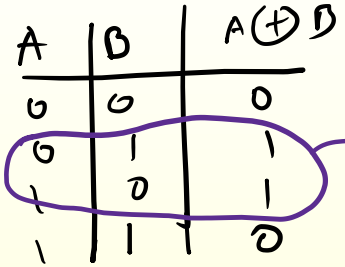
XOR gate

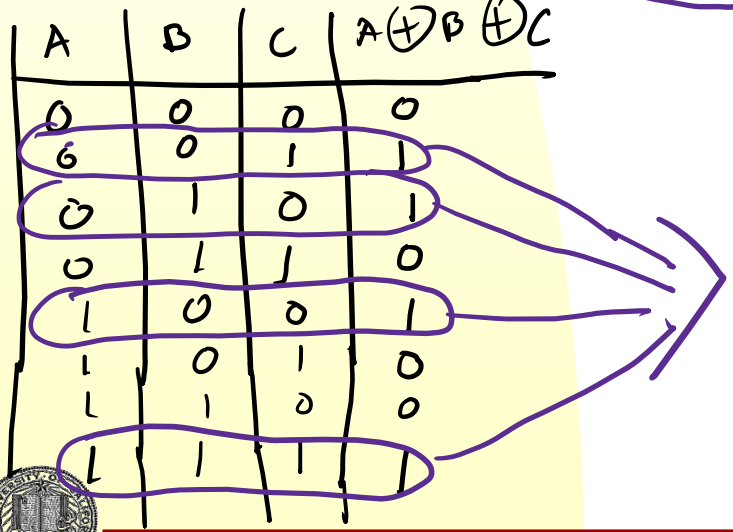


XOR

- Output is a(XOR)b=a'b+ab' (for 2 operands)
- In general, we say output is
  - 1 if odd number of inputs are 1
  - 0 if even number of inputs are 1

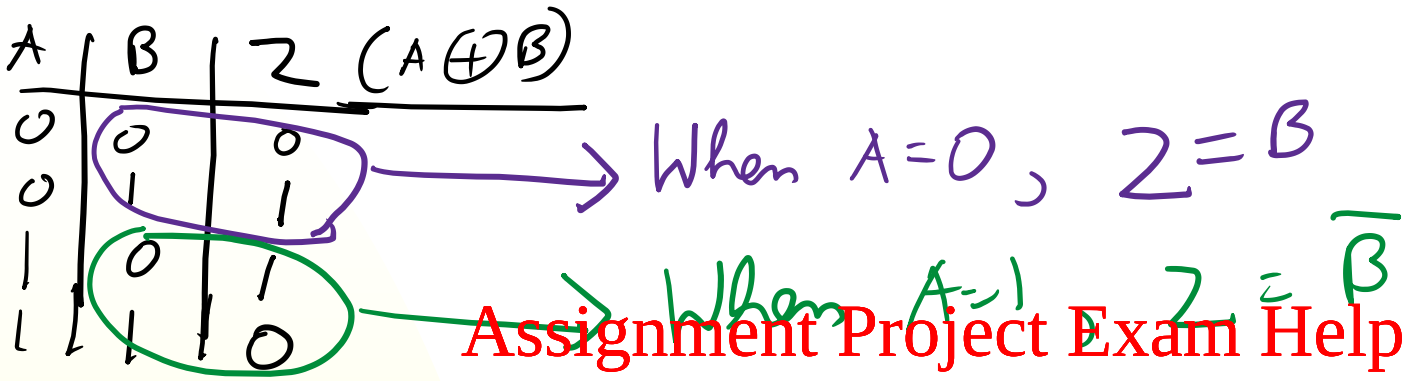
- Output is a(XOR)b=a'b+ab' (for 2 operands)
- In general, we say output is
  - 1 if odd number of inputs are 1
  - 0 if even number of inputs are 1



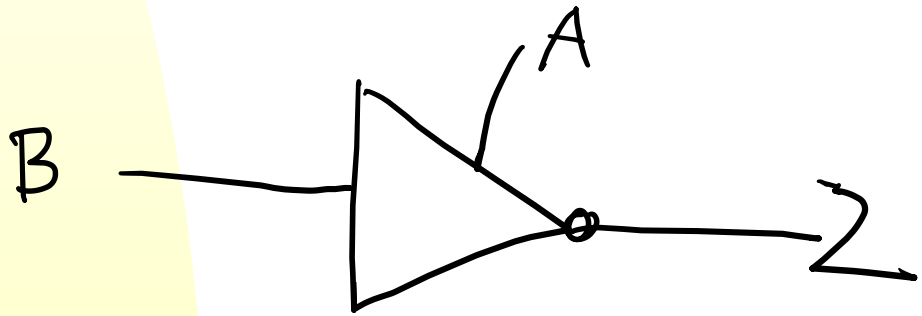


odd no. of inputs are 1, therefore output is 1

XOR gate as a “programmable "inverter (NOT gate)



- Thus, we can “program” Z to either be inverse of B, or simply be equal to B, depending on the value of A



- Above is an alternative symbol for XOR gate as a programmable inverter. But in general, we will stick to the normal XOR gate symbol