## Binary Logics:

→ Consist of binary variables and logical operation

↳ A, B, C, x, y, z
↳ possible values [0, 1]

↳ AND
↳ OR
↳ NOT
↳ etc.

## Logic Gates:

→ Most Basic Digital Device

→ 1 or more input & 1 outputs

→ Output follows a certain logics

## Truthtable:

→ A list of all possible inputs and outputs

## All logic gates

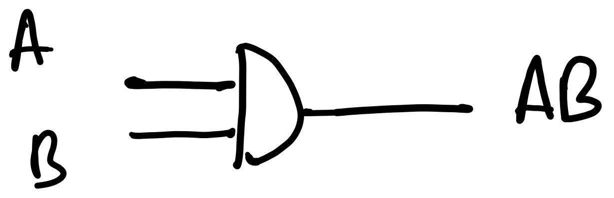
NOT , AND , OR , XOR
NAND , NOR , XNOR

* NOT, AND, OR are Basic gates.

* NAND & NOR are universal gate.

A universal gate is a gate which can implement any other gate.

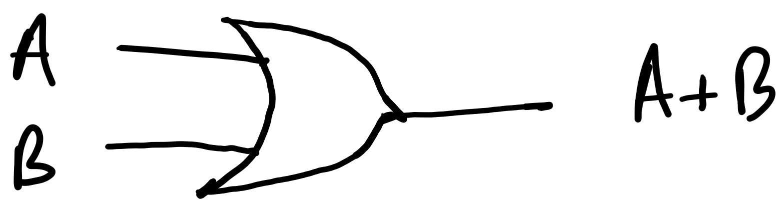
## AND :




| A | B | AB |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

* Output 1 when all inputs are 1,

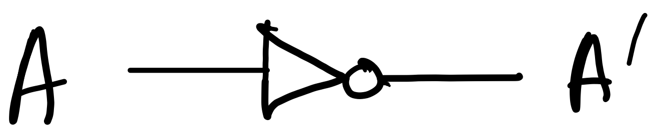
## OR :




| A | B | A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

* Output 1 when atleast one input is 1

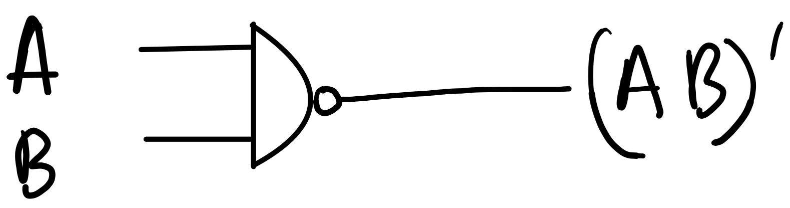
## NOT (Inverter)



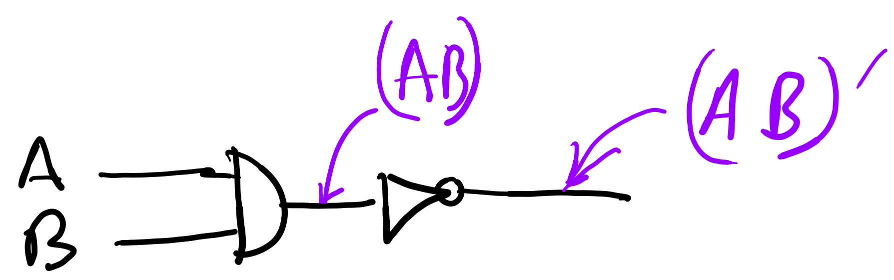

| A | A' |
|---|---|
| 0 | 1 |
| 1 | 0 |

* Output is inverse of input

## NAND



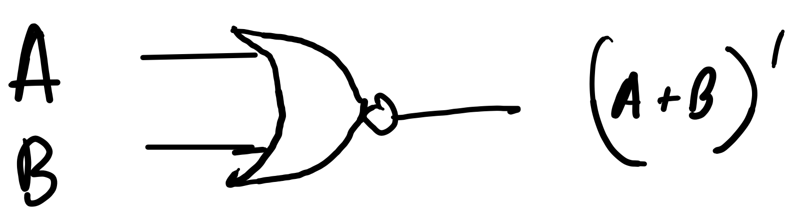

aka



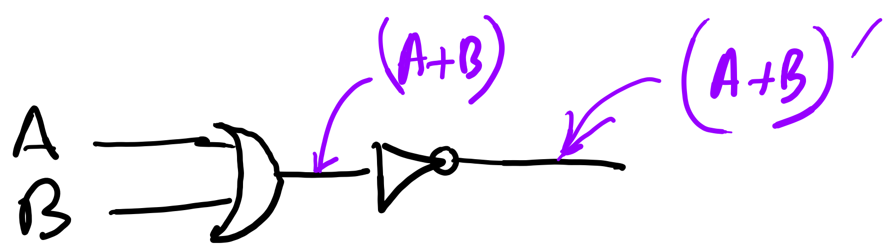

| A | B | (A.B)' |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

* Inverse of AND

## NOR



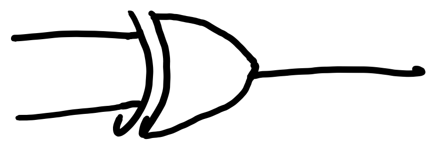

aka




| A | B | (A+B)' |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

* Inverse of OR

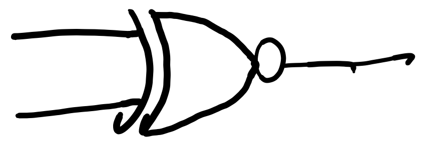
## XOR

A
B

$A \oplus B = AB' + A'B$

| A | B | $(A \oplus B)'$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Output 1 if odd number of input is 1

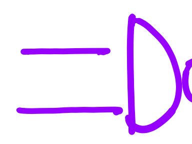
## XNOR :

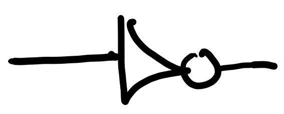
A
B

$A \odot B = AB + A'B$

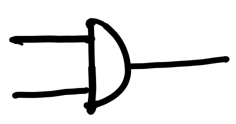
| A | B | $(A \odot B)'$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

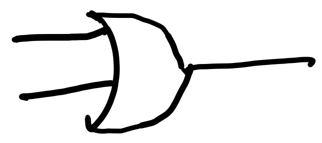
Inverse of XOR

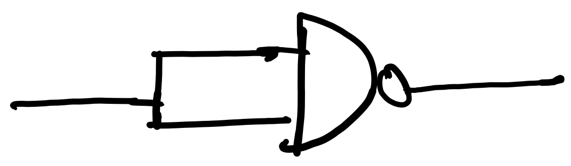
## Basic Gates using NAND :

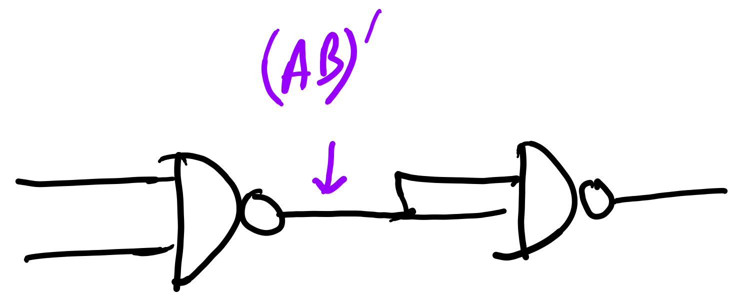
NAND A B  $(AB)'$

NOT A  $A'$

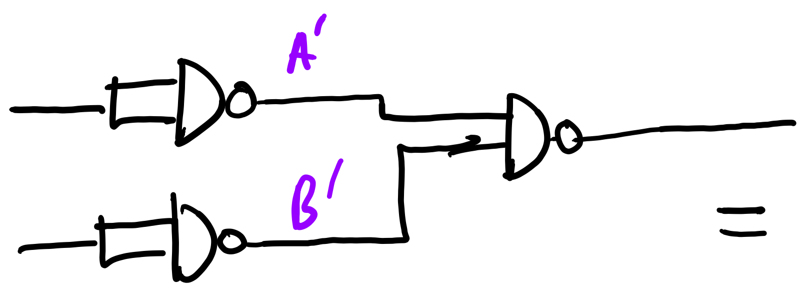
AND A B  $AB$

OR A B  $A+B$

A  $(AA)' = A'$

A B 

$((AB)')' = AB$

A B 

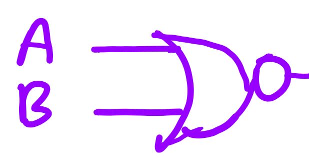
$(A'B')'$

$= (A')' + (B')'$

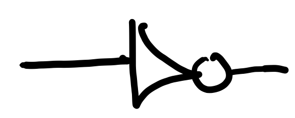
$= A + B$

Using DeMorgan's law

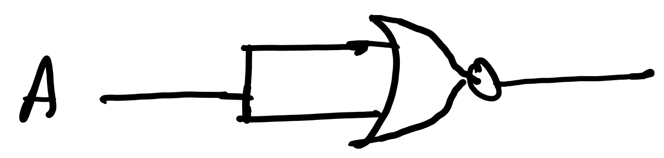
## Basic Gates using NOR :

NOR 

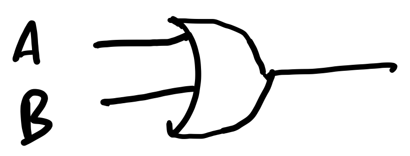
$(A+B)'$

NOT 

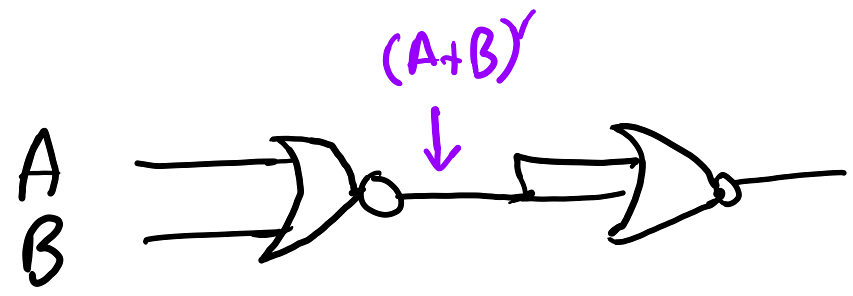
$A'$



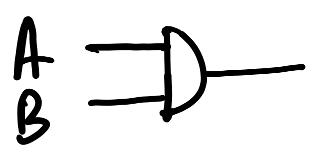
$(A+A)' = A'$

OR 

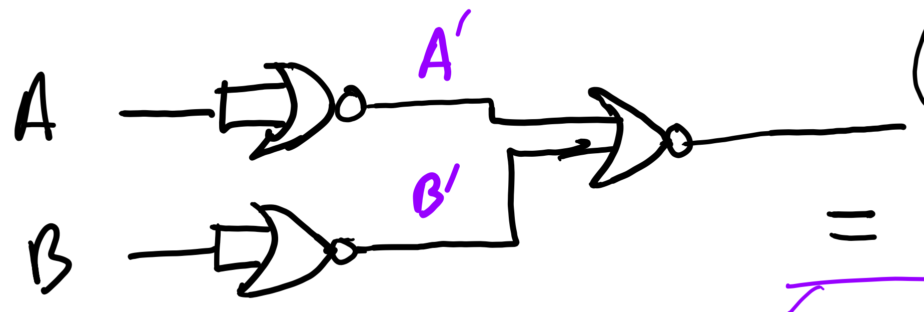
$A+B$



$((A+B)')' = A+B$

AND 

$AB$



$(A'+B')'$

$= (A')' \cdot (B')'$

$= AB$

Demorgan's law

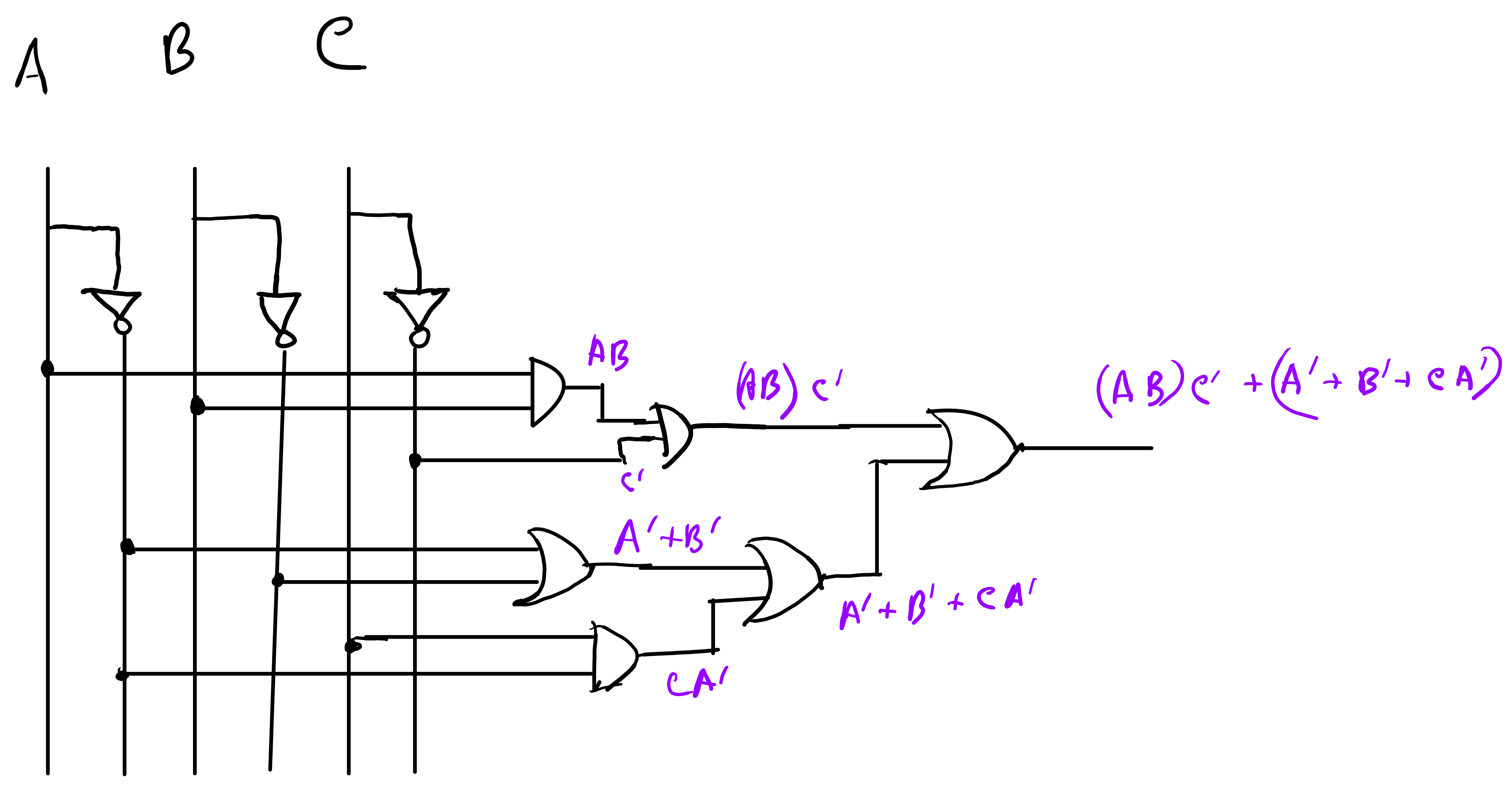
## Proof using Truth table:

☆ Proof that $\underbrace{x \cdot (y+z)}_{LHS} = \underbrace{(x \cdot y) + (x \cdot z)}_{RHS}$

| $x$ | $y$ | $z$ | $y+z$ | $x \cdot (y+z)$ | $x \cdot y$ | $x \cdot z$ | $(x \cdot y) + (x \cdot z)$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

$\therefore$ LHS = RHS.

Implement $(AB)C' + (A' + B' + CA')$ using Basic Gates



# Boolean Algebra

## Basic theorem of Boolean Algebra

* Theorem can be proved using Truthtable or Algebric Manipulation

## Basic Theorems are:

| | | |
|---|---|---|
| ⇒ $x + 0 = x$ | $x \cdot 1 = x$ | identity |
| ⇒ $x + x' = 1$ | $x \cdot x' = 0$ | complement |
| ⇒ $x + x = x$ | $x \cdot x = x$ | |
| ⇒ $x + 1 = 1$ | $x \cdot 0 = 0$ | |
| ⇒ $(x')' = x$ | | |
| ⇒ $x + y = y + x$ | $x \cdot y = y \cdot x$ | commutative |
| ⇒ $x(yz) = (xy)z$ | $x + (y + z) = (x + y) + z$ | |
| ☆ ⇒ $x(y+z) = xy + xz$ | ☆ $x + yz = (x+y)(x+z)$ | Distributive |
| ☆ ⇒ $(x+y)' = x'y'$ | ☆ $(xy)' = x' + y'$ | De Morgan |
| ☆ ⇒ $x + xy = x$ | ☆ $x(x+y) = x$ | Absorption |

☆ All are important but student tend to forget star marked ones while simplifying

# Duality Principle:

* If we interchange operators and elements as follows,

$$\boxed{(OR)\ + \rightleftharpoons \cdot\ (AND)}$$

$$\boxed{0 \rightleftharpoons 1}$$

Example:

if $\quad a + (b \cdot c) + 0 = (a+b) \cdot (a+c) \cdot 1$

then we can find its dual expression

$$a \cdot (b+c) \cdot 1 = (a \cdot b) + (b \cdot c) + 0$$

* Duality Gives us free theorems

If a theorem/expression for example,

$x + 1 = 1$ is valid/Proved.

then its dual expression,

$x \cdot 0 = 0$ is also valid/proved

## Operator Precedence:

Parenthesis $\longrightarrow$ NOT $\longrightarrow$ AND $\longrightarrow$ OR

Highest          Lowest

## Boolean Expression simplification: (Using Boolean Algebric Manipulation)

* Simplify: $(x'y'z) + (x'yz) + (xy')$

$$= x'z(y+y') + xy'$$

$$= x'z \cdot 1 + xy'$$

$$= x'z + xy'$$

* Simplify, $xy + xy'$

$$= x(y+y') = x \cdot 1 = x$$

* Simplify, $BC + AB' + AB + BCD$

$$= BC(1+D) + AB' + AB$$

$$= BC \cdot 1 + AB' + AB$$

$$= BC + A(B'+B)$$

$$= BC + A \cdot 1$$

$$= BC + A$$

**Prove,** $(A \oplus B)' = A \odot B$ using Boolean Manipulation:

$$
\begin{aligned}
& (A \oplus B)' \\
&= (AB' + A'B)' \\
&= (AB')' \cdot (A'B)' && \rightarrow \text{Applied Demorgan} \\
&= (A' + B'') \cdot (A'' + B') && \rightarrow \text{'' \quad ''} \\
&= (A' + B) \cdot (A + B') && \rightarrow \\
&= \underbrace{AA'}_{0} + A'B' + BA + \underbrace{BB'}_{0} \\
&= A'B' + BA \\
&= AB + A'B' \\
&= A \odot B
\end{aligned}
$$

**Practice more & more.**

## Complement of a function:

if $F = (x+y)$, then its complement is $F' = (x+y)'$

However, we can also find complement of a function by:

1. Taking Dual of the function
2. Complement each variables

**Example:**

$F = x'yz' + x'y'z$

1. Dual $= (x' + y + z') \cdot (x' + y' + z)$
2. Complement each var $= (x + y' + z) \cdot (x + y + z') = F'$

* Find complement of $x(y'z' + yz)$

1. Dual $= x + ((y' + z') \cdot (y + z))$

2. Complement $= x' + (y + z) \cdot (y' + z')$

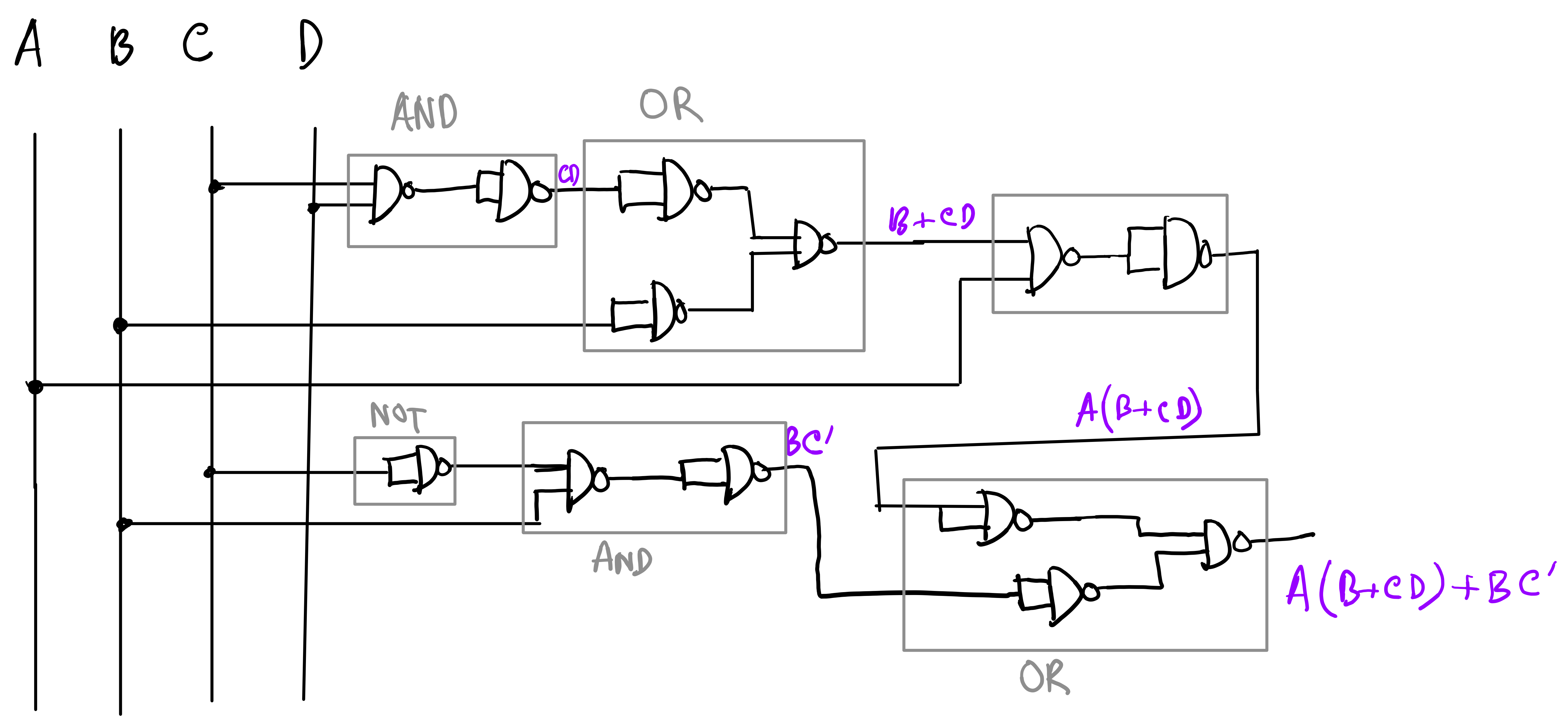
$\therefore \quad F' = x' + (y+z)(y'+z')$

# Using NAND or NOR to Draw circuit of a function:

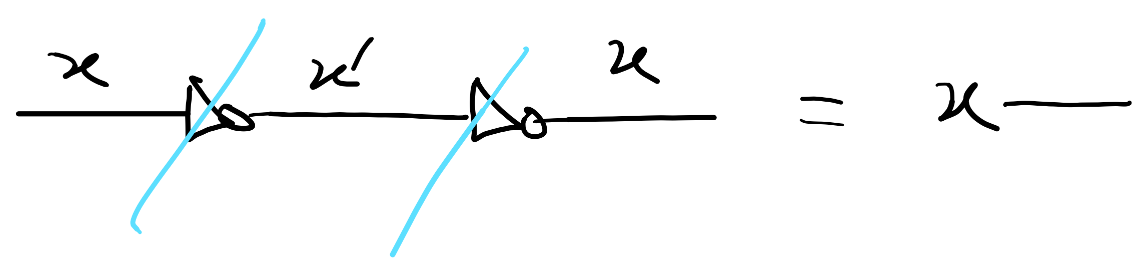
Steps:

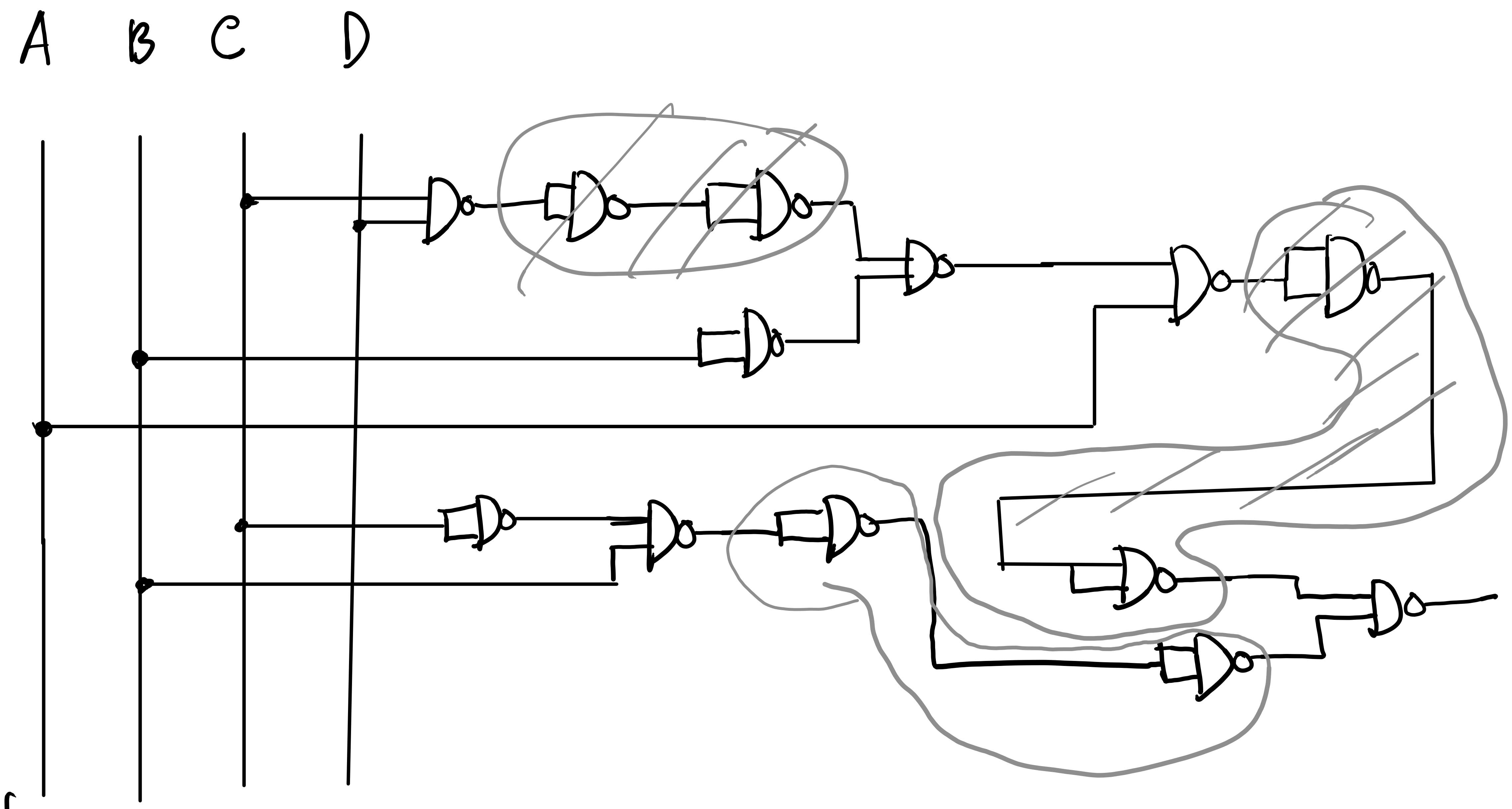
1. Represent the expression using AND, OR, NOT gate.
2. Draw each gate with equavalent NAND / NOR representation.
3. Remove any 2 cascading inverter.
4. Remove inverter from single input connection and replace input with it's complement. (Only if there is a constraint

## Implement $F = A(B+CD) + BC'$ using NAND only.




Now, if we want to reduce number of gates, we can remove cascading inverters.






Therefore