LECTURE #2 01/07/04

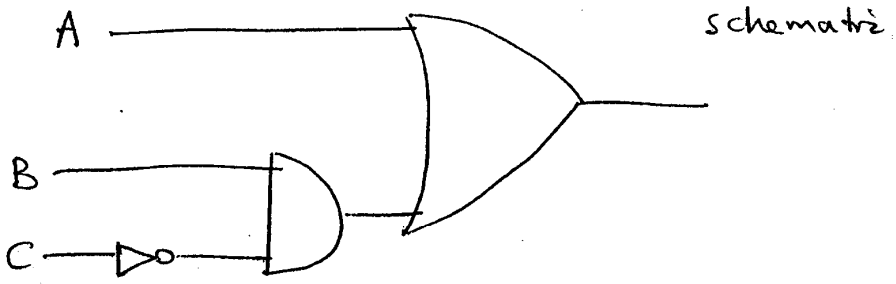
~~ECE 152A Lecture #2~~

[Reading: See syllabus]

- Boolean functions:

$$F = A + BC'$$

- Implementation:



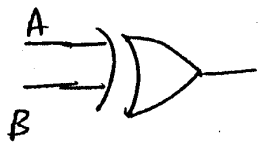
- Truth table:

A	B	C	F	
0	0	0	0	"off-terms"
0	0	1	0	
0	1	0	1	
0	1	1	0	"on-terms"
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	

Shows how to go from ~~the~~ a Boolean expression to the truth table.

— Can we do the reverse?

Ex1 XOR:



A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

$$F = A'B + AB'$$

Ex2

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

$$F = A'B'C' + A'B'C + A'BC + ABC'$$

- The truth table gives a complete characterization of the input/output behavior of a combinational circuit. But it is not compact (grows exponentially in size with the # input variables) and is hard to manipulate. I/O matrix
- Tool that we use is Boolean algebra.

① Associative Law:

$$A + (B + C) = (A + B) + C$$

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

② Commutative Law:

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

③ Distributive Law:

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$\boxed{A + (B \cdot C) = (A + B) \cdot (A + C)} \leftarrow \text{Important.}$$

④ $A + 0 = A$ $\quad A + 1 = 1$

$A \cdot 0 = 0$ $\quad A \cdot 1 = A$

(Ask class)

Boolean simplification (Algebraic):

Review:
Associative: $A+(B+C) = (A+B)+C$
Commutative: $A \cdot B = B \cdot A$, $A+B = B+A$
Distributive: $A \cdot (B+C) = A \cdot B + A \cdot C$

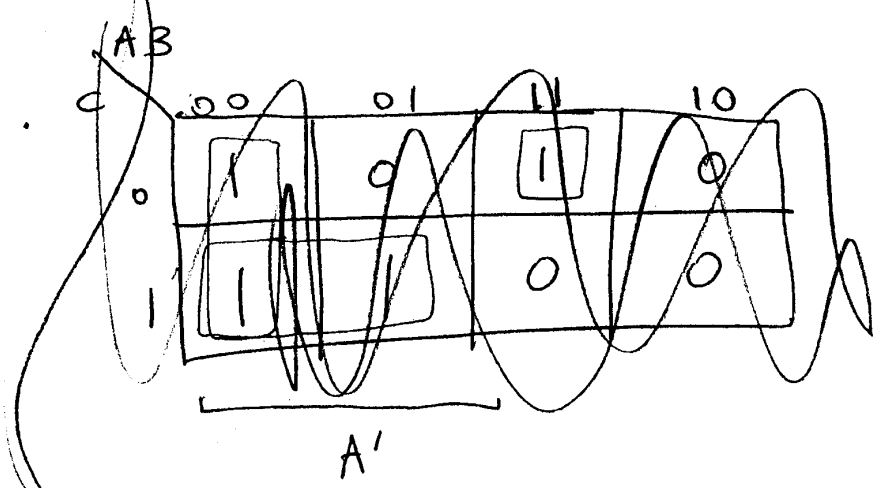
$A+BC = (A+B)(A+C)$

$A \cdot 1 = A$
$A+1 = 1$
$A \cdot 0 = 0$
$A+0 = A$

$$F = A'B'C' + A'B'C + A'BC + ABC'$$

$$= A'B'(\underbrace{C'+C}_{1}) + A'BC + ABC'$$

$$= A'B' + A'BC + ABC'$$



$A'B' + A'C + ABC'$

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

(distributive law #2)

$$= A'(\underbrace{B'+B}_{1})(B'+C) + ABC'$$

$$= \boxed{A'B' + A'C + ABC'}$$

+ Associative law, commutative law, distributive law, {0,1} operation from Lecture #1

Simplification Theorems (Divide board into 7 sections)

1) $XY + XY' = X(\underbrace{Y+Y'}_{1}) = \boxed{X}$

2) $X + XY = X(\underbrace{Y+1}_{1}) = \boxed{X}$

3) $(X+Y')Y = XY + \underbrace{YY'}_{0} = \boxed{XY}$

4) $(X+Y)(X+Y')$

$= XX + \underbrace{YY'}_{0} + YX + XY'$

$= X + YX + XY'$

$= X(Y+1) + XY'$

$= X + XY' = X(\underbrace{Y'+1}_{1}) = \boxed{X}$

5) $X(X+Y)$

$= XX + XY = X + XY = X(\underbrace{Y+1}_{1}) = \boxed{X}$

6) $XY' + Y$

$= Y + XY' = (\underbrace{Y+Y'}_{1})(Y+X) = \boxed{X+Y}$

BATCH

7) Consensus theorem

$$XY + X'Z + YZ$$

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

$$= XY + X'Z + XYZ + X'YZ$$

$$= XY(Z+1) + X'Z(Y+1)$$

$$= XY + X'Z$$

(Do this yourself)

- we see that the YZ term was redundant.

↓

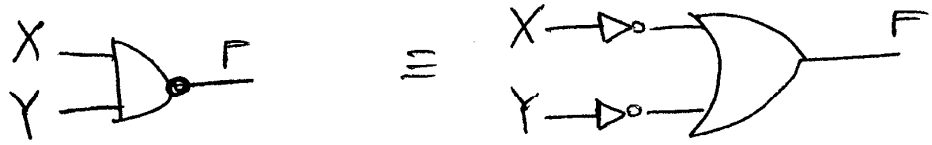
"Consensus term"

De Morgan's Laws:

① $(X+Y)' = X'Y'$.

"Bubble-pushing".

② $(X \cdot Y)' = X' + Y'$



Ex1 $(A'+B)' = (A')' \cdot B' = A \cdot B'$

Ex2 $\left(\boxed{(A'+B)}\boxed{C'}\right)' = (A'+B)' + (C')'$

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

$= [(A')' \cdot B'] + C$

$= A \cdot B' + C$.

Ex3] $((AB'+C)D'+E)'$

$= ((AB'+C)D')' \cdot E'$

$(AB'+C)' + D$

$(AB')' \cdot C'$

$A'+B$

→ Bubble-pushing

$= ((A'+B)C'+D)E'$

Dual of a Boolean expression:

AND → OR

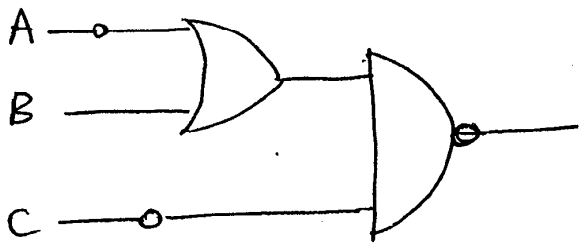
OR → AND

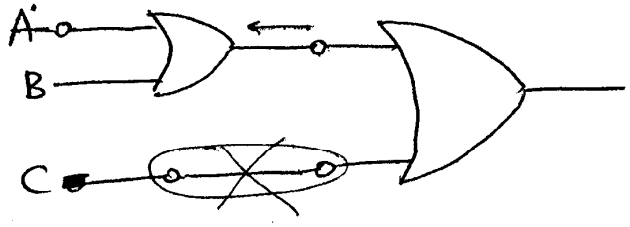
0 → 1

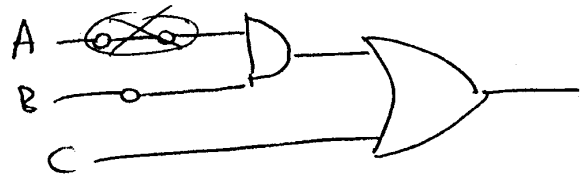
1 → 0

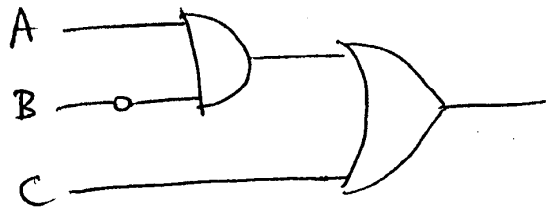
(Variables & complements are unchanged)

Ex2] $((A'+B)C')' =$





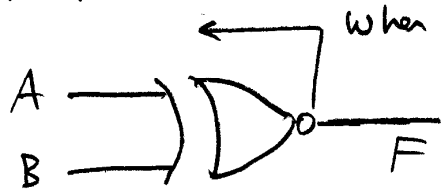




~~A~~ $(AB')+C$. ✓

Bubble-pushing through an XOR gate:

Problem: Derive a bubble-pushing law for the XOR gate.



When I push this bubble, what happens?

(An ill-defined question—

What we mean is:
can you find an equivalent expression to this, that is simple as well?)

$$F = (A \oplus B)' = (A'B + AB')'$$

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

$$= A'B' + BA$$

$$= A' \oplus B$$

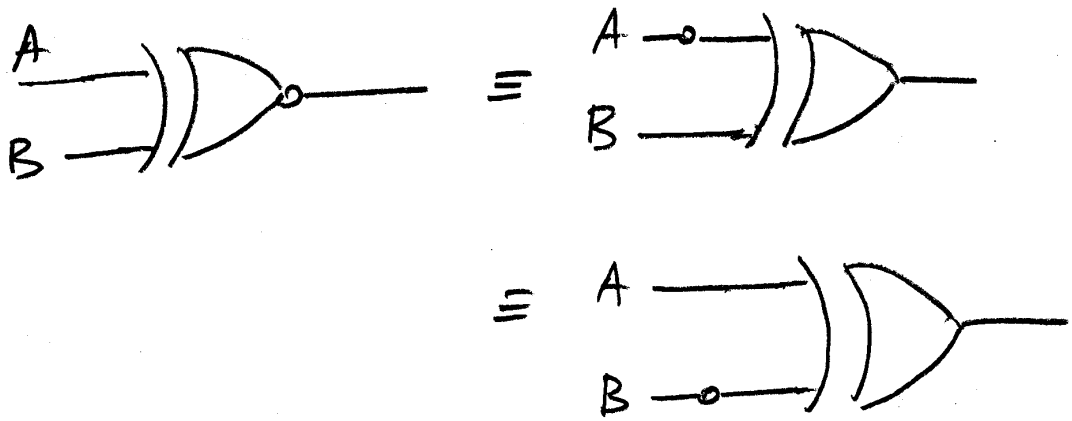
(write out:
$A' \oplus B = AB + A'B'$
by defn of XOR)

OR

key step

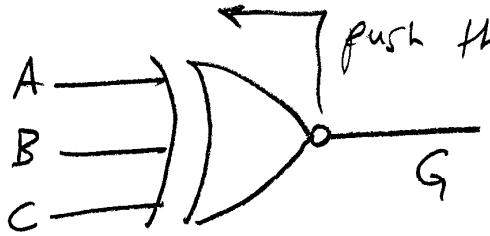
$$= A \oplus B'$$

- Hence:



"The bubble appear at only 1 of the two inputs and leave the gate unaltered."

Question:



Answer: $G = [A \oplus B \oplus C]'$

$= [(A \oplus B) \oplus C]'$

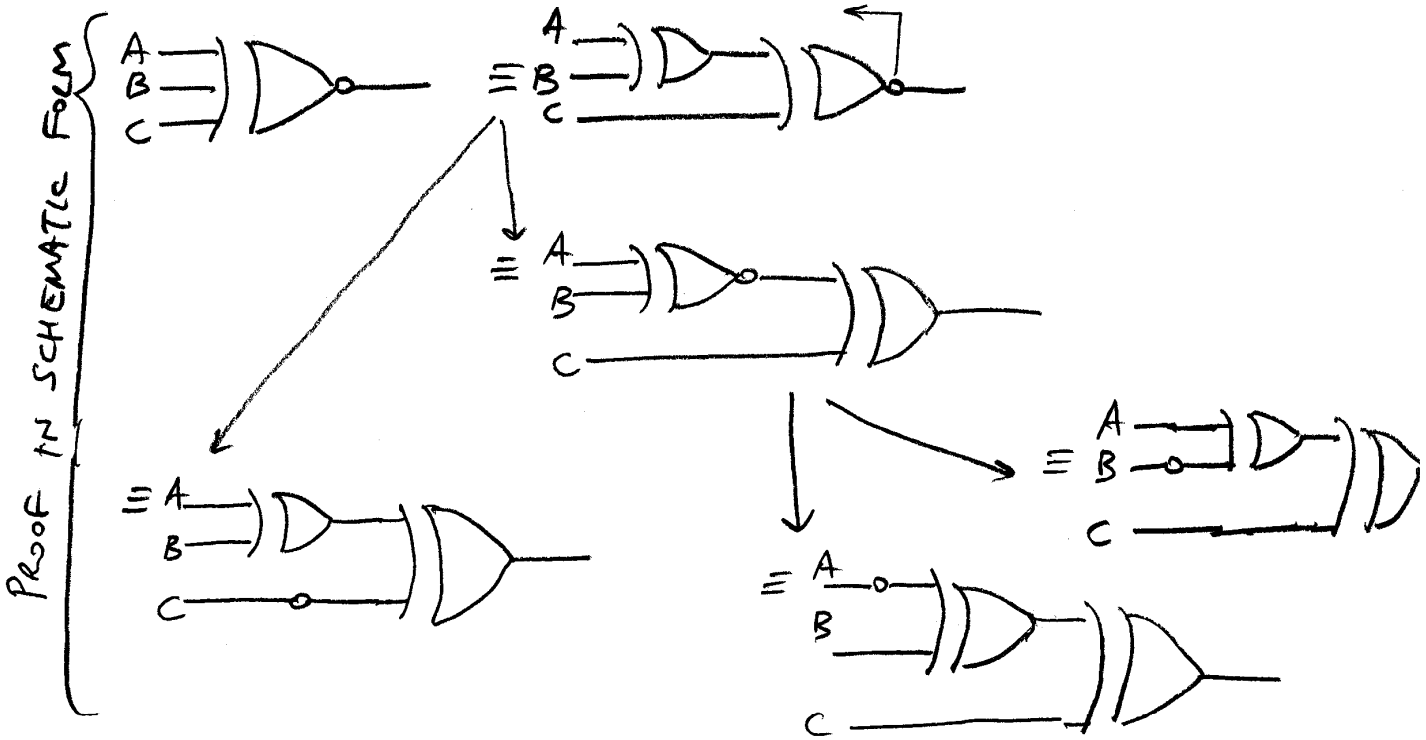
$= [(A \oplus B) \oplus C']$ by our above 2-input bubble-pushing law

OR

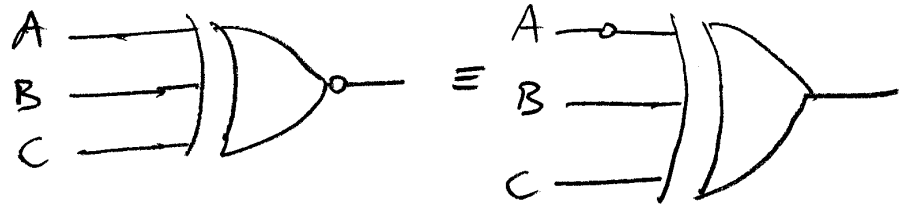
$= [(A \oplus B)' \oplus C]$

$= [(A' \oplus B) \oplus C]$

That is:

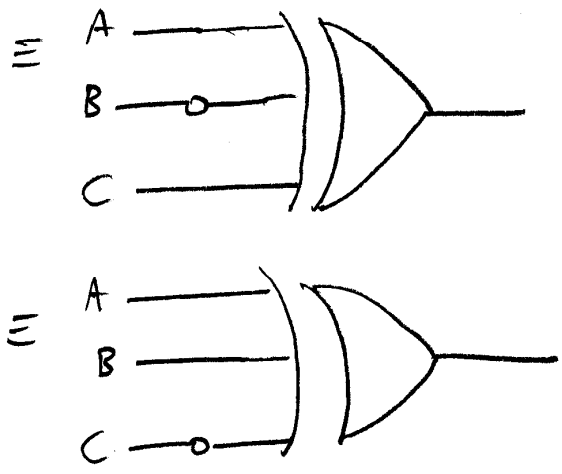


That is:



i.e.

"Bubble appears at only 1 input for multiple-input XOR gates, too"



Proving the Validity of a Boolean equation:

Example 1:

Is the XOR ($\oplus$) operation commutative?

Prove or disprove it, using the def^n of XOR: $A \oplus B \equiv A'B + AB'$

Scratch work: ① Make a guess: [YES, true.]

① write down what precisely we want to show:

Show: XOR commutative

$\Leftrightarrow \forall A, B,\ \underbrace{A \oplus B}_{\equiv LHS} = \underbrace{B \oplus A}_{\equiv RHS}$

② write down further what the LHS means, and RHS means

$$LHS = A \oplus B \overset{\uparrow}{=} A'B + AB'$$

by def^n of $\oplus$

$$RHS = B \oplus A = B'A + BA'$$

Now: $A'B + AB' = B'A + BA'$ (equal)

because: $A'B$ equal BA'

Scratch work complete.

— Now, write down the proof.

Proof: Let A, B be Boolean variables

$A \oplus B = A'B + A \cdot B'$ by def'n of $\oplus$

$= BA' + B' \cdot A$ by commutative law for $\cdot$ (AND)

$= B'A + BA'$ by commutative law for $+$ (OR)

$= B \oplus A$ by def'n of $\oplus$.

(linear flow of arguments from LHS to RHS)

$\therefore \oplus$ is commutative. QED.

Example 2:

Prove: $(A \oplus B) \oplus C = A \oplus (B \oplus C)$

(i.e. prove that $\oplus$ is associative.)

Pf $(A \oplus B) \oplus C = (A'B + AB') \oplus C$ by def'n of $\oplus$

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

$+ (A'B + AB') \cdot C'$ " " " "

$= [(A'B)' \cdot (AB')'] \cdot C$ De Morgan's Law

$+ A'BC' + AB'C'$

$$= [(A + B') \cdot (A' + B)]\, C + A'BC' + AB'C'$$ De Morgan's law

$$= [AB + A'B']\, C + A'BC' + AB'C'$$

$$= A[BC + B'C'] + A'[B'C + BC']$$ distributive law (in reverse)

$$= A \cdot [(B \oplus C)'] + A'[B \oplus C]$$ by defn of X_{NOR}

$$= A \oplus (B \oplus C)$$ by defn of X_{OR}

QED.

Disproving claims:

In order to disprove a claim you have to give (at least) one CONCRETE counterexample.
[not good to wave hands: the counterexample must be a specific concrete one.]

Example 3:

Prove or disprove: (Assume A, B, C are Boolean variables)

$$\text{If } A + B = A + C \text{, then } B = C.$$

(- This statement is FALSE.)

To show this:

Intuition: If $A = 1$ then $1 = 1$, even if $B = 1$ and $C = 0$ i.e. $B \neq C$.

Proof: (that it is FALSE):

Consider $A = 1$, $B = 1$, $C = 0$. Then:

$$A + B = 1 + 1 = 1 \quad \text{and} \quad A + C = 1 + 0 = 1$$

(Hence $A + B = A + C$) but $B = 1 \neq 0 = C$

QED

Functional completeness:

A family of logic gates is said to be <u>functionally complete</u> iff every Boolean function can be implemented using (any number of gates) from that family.

EX 1 The family {AND, OR, NOT} is functionally complete because every Boolean function can be written in a sum-of-products or product-of-sums form that uses only AND, OR, NOT operations.

EX 2 {AND, NOT} : (Ask class)

is functionally complete.

Pf First, Implement OR gate using AND and NOT gates. (use De Morgan's Law),

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

Then, since OR can be implemented this way, since AND, OR, NOT $\Rightarrow$ functionally complete by EX 1, QED.

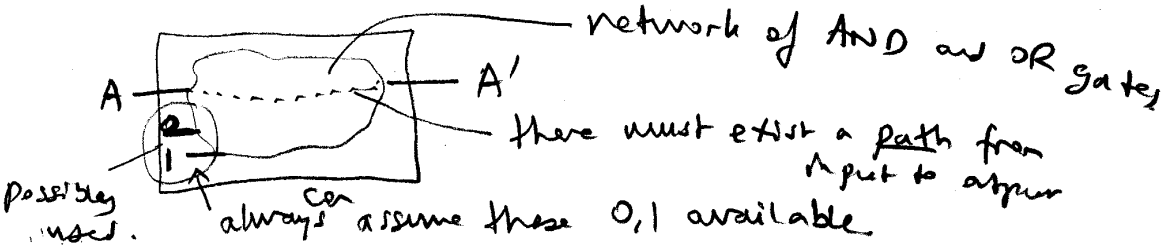
EX 3 {OR, NOT} is functionally complete

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

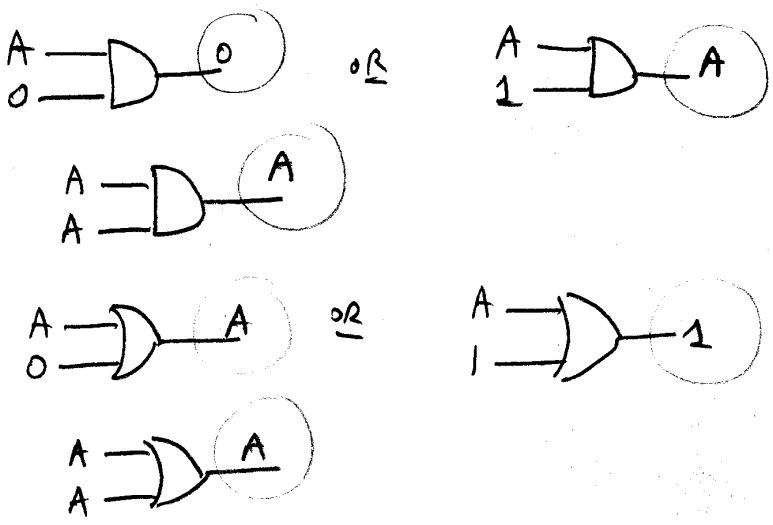
and similar as in Ex 2

EX 4] {AND, OR} : not functionally complete.

[Pf] Show that NOT not implementable by using only AND and OR gates. Assume that it could:



First connection:



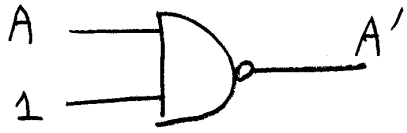
Hence, the first gate in the path can produce only {0, 1, A}, recursively applied: last gate can produce only {0, 1, A) ∴ contradicts that A' can be produced.

EX 5] {NAND} (Ask class)

— If functionally complete.

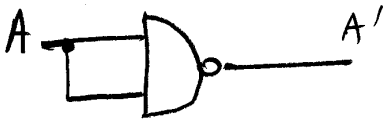
(Useful) will find the TTL chip with 4 NAND gates.

Pf] Show that AND, NOT can be implemented using NAND.

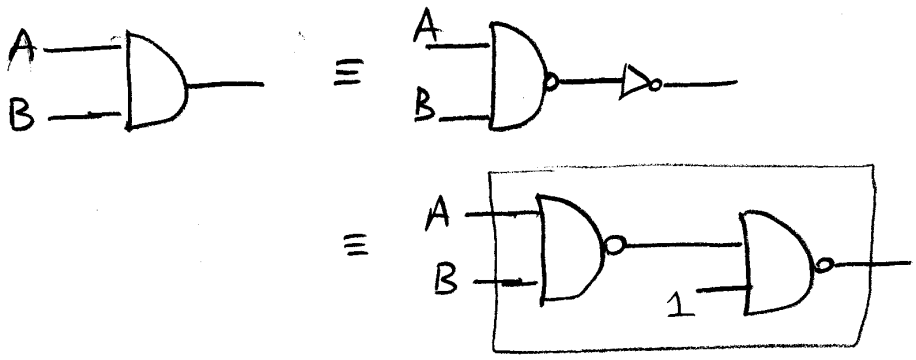


A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

OR



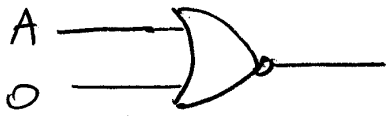
∴ NOT can be implemented using NAND, as above.



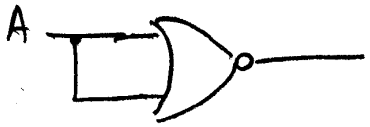
QED

EX5] {NOR} : functionally complete.

Pf] Show: OR, NOT can be implemented using NOR

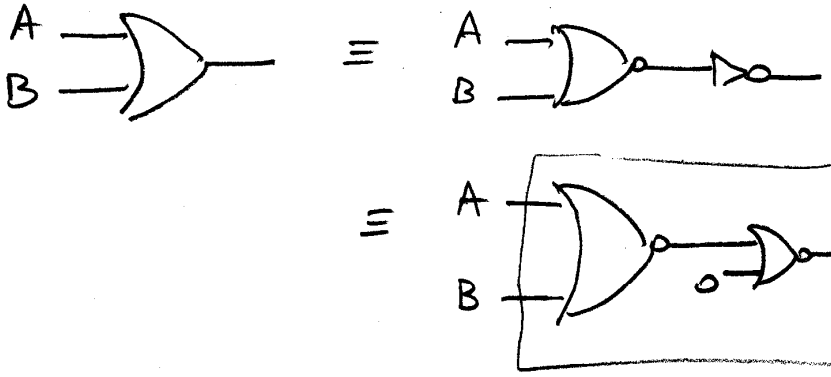


OR



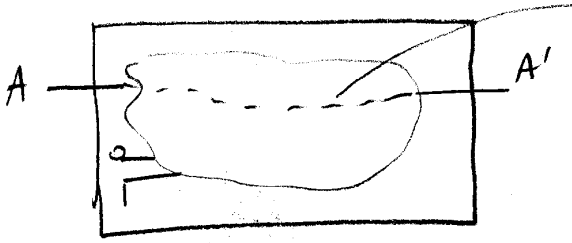
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Now,



EX 6] {XOR} is not functionally complete

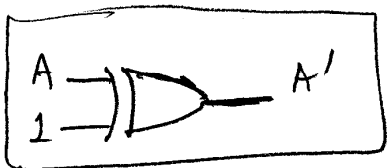
Pf] Assume that it is. Then, the NOT should be implementable as:



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

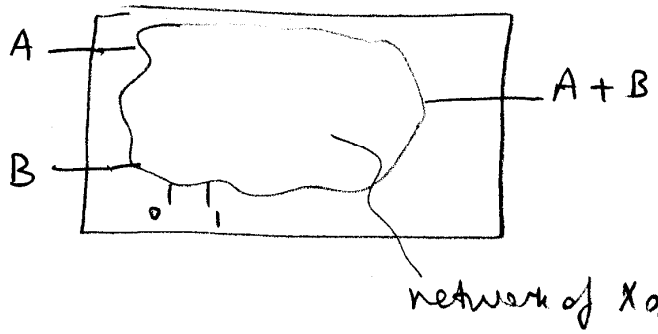
First gate on this path:

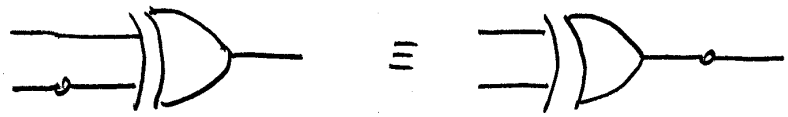
OR



∴ NOT is implementable.

- Try to implement OR: using {XOR, NOT}



Recall:  ≡

• Push all bubbles forward to the terminal point on RHS.

Then:  

useless

$$A + B = A\left[\underbrace{\oplus A \oplus \ldots \oplus A}_{\text{possibly many times}}\right] \oplus B\left[\underbrace{\oplus B \oplus B \ldots \oplus B}_{\text{possibly many times}}\right]$$

$$[\oplus 0 \ldots \oplus 0] \oplus [1 \oplus \ldots \oplus 1]$$

only possibilities that can come out =

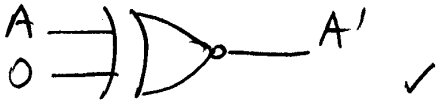
- A
- B
- $A \oplus 1 = A'$
- $B \oplus 1 = B'$
- $A \oplus B$
- $A \oplus B \oplus 1 = A \oplus B'$

∴ Impossible to produce OR or AND.

QED.

Ex7 {XNOR} not functionally complete

A	B	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

Pf NOT: 

✓
implementable

By way of contradiction:
Assume that OR is implementable. Then:

$$A+B = A[\oplus A \ldots] \oplus B[\oplus \ldots B]$$

$$[\oplus 1[\oplus 1 \ldots \oplus 1]) \oplus 0 \oplus \ldots]$$

By same argument as in Ex 6, not implementable. only forms that can come out are: $\{A \oplus B,\ (A \oplus B)' = A' \oplus B = B' \oplus A,\ A',\ B'\}$.

Q.E.D.

SUMMARY of Examples:

functionally complete	f. incomplete
{AND, OR, NOT}	{XOR}
{AND, NOT}	{XNOR}
{OR, NOT}	{AND, OR}
{NAND}	
{NOR}	

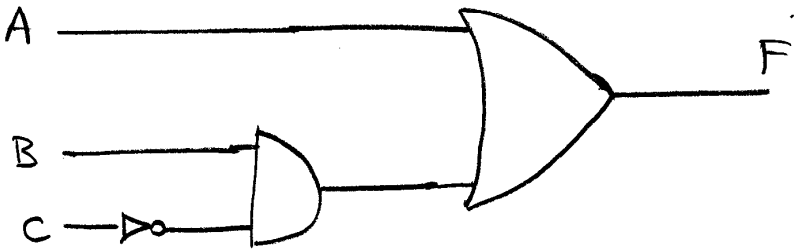
universal gates

A logic gate that by itself forms a functionally complete family is said to be a universal gate.

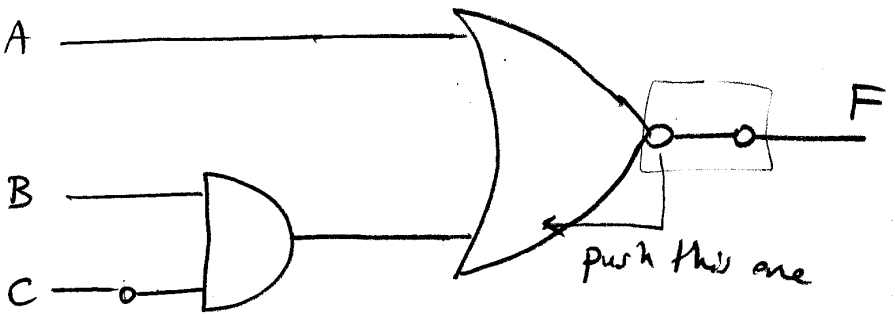
NAND/NOR - only circuit designs

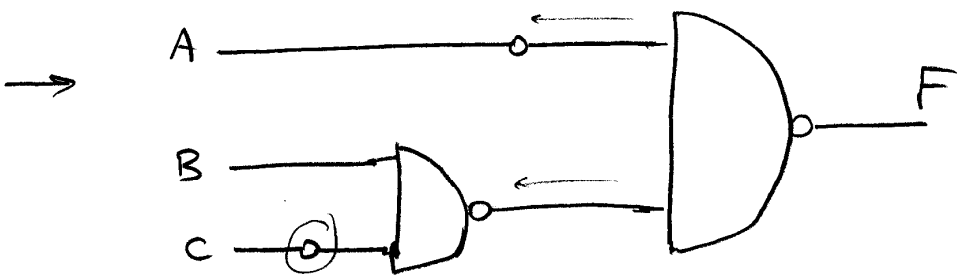
- Since NAND and NOR gates are universal gates, it is possible to implement any Boolean circuit using only NAND gates OR using only NOR gates.

EX 1 Implement $F = A + BC'$ using only NAND gates.

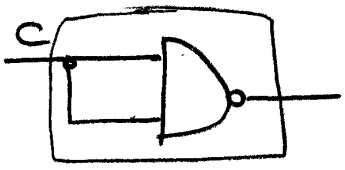


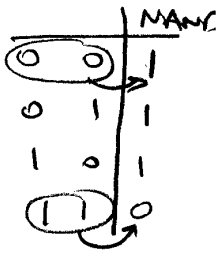
Idea: - create double-bubbles and then push them as necessary to eliminate OR gates.



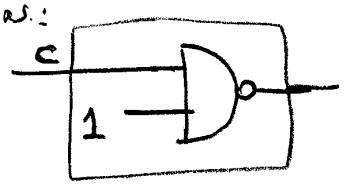


Finally, the remaining inverter C ——▷○—— can be implemented as:





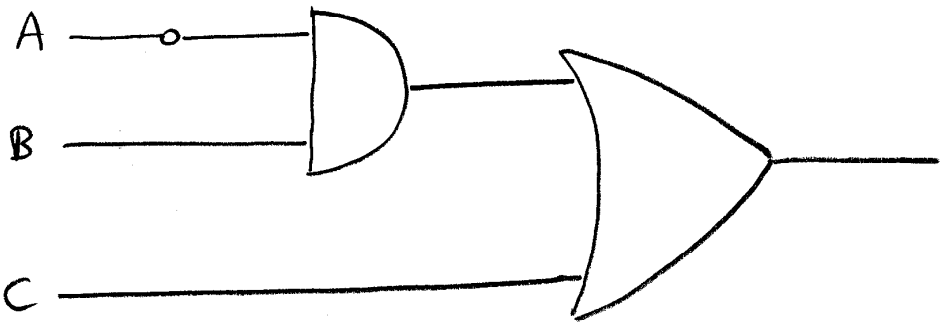
OR as:



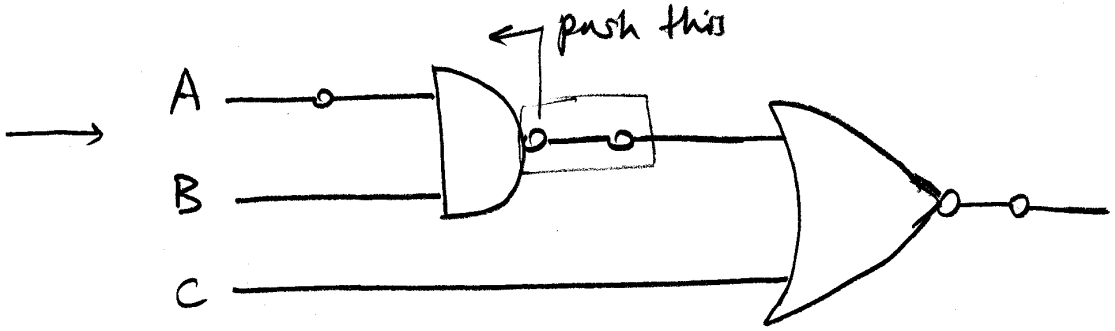
∴ A total of 3 NAND gates needed in this design

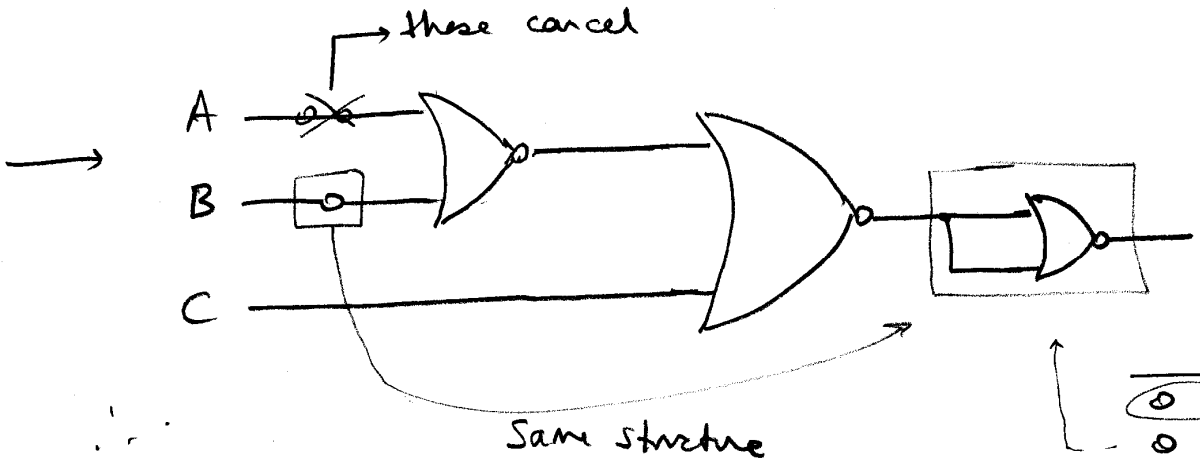
EX2| Implement: $G = A'B + C$

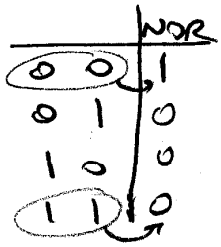
using only NOR gates.



- Idea: Push bubbles backward toward AND gates to create NOR gates





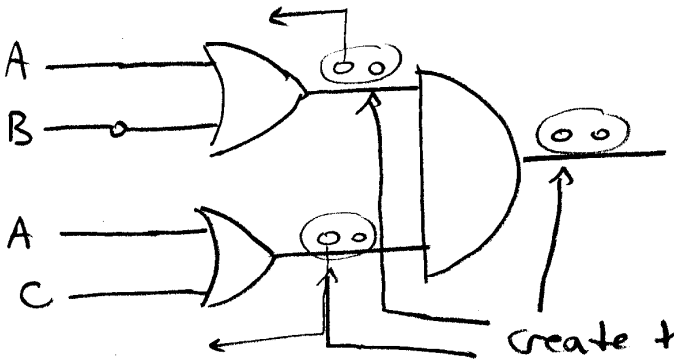


∴ 4 NOR gates are used in this design

(2 to implement inverter)

EX 3] Implement: $H = (A + B') \cdot (A + C)$

using only NAND gates.



create these double bubbles and push one back through OR

Dual of a Boolean expression:

~~AD A~~

$f^D(0, 1, A, B, C, \cdot, +, ')$

$= f(1, 0, A, B, C, +, \cdot, ')$

Ex1 $(X+Y)^D = X \cdot Y$

$(AB' + C)^D = (A+B') \cdot C$

Ex3. Prove: $XY' + Y = X + Y$

Dual ↓ ↑ Dual

$(X+Y') \cdot Y = XY + \underbrace{YY'}_{0} = XY$

[$(f^D)^D = f$.]

Thm: 2- $(f(x_1, .., x_n))^D = f'(x_1', .., x_n')$.

Homework: Solve all problems at the end of Ch2.

~~(2.1 - 2.22)~~