Brought to you by …

Priti Srinivas Sajja

Logic Circuits

Professor

P G Department of Computer Science,
Sardar Patel University, Vidyanagar-388 120, Gujarat, India.

Unit 4

PS01CMCA02
Course Content

Tutorial
Practice Material

Acknowldgement
References

Website
pritisajja.info

Topics

- Basic gates and Universal gates
- Boolean algebra and Truth table
- Circuit Equivalence
- DeMorgan's Theorems

Logic Gates

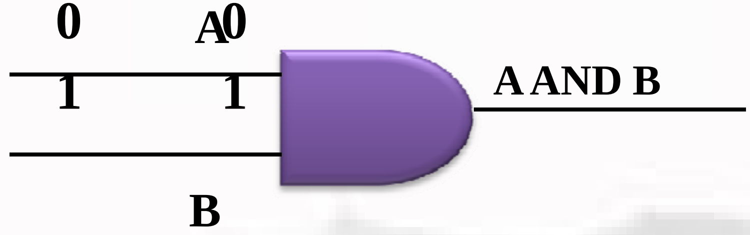
- Logic gates are the **fundamental building blocks** of digital systems.
- Gate is a logic circuit with **one or more inputs but only one output.**
- Gate is a logic circuit in which voltage levels represent logic 1 and logic 0.
- A table which contains **all possible alternatives** of input variables and corresponding output, is called a **truth table.**

AND Gate

An **AND** gate has two or more input signals but only one output signal. **<u>If all inputs are high then output is high.</u>**

A	B	C	A AND B AND C(ABC)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

A	B	A AND B (AB)
0	0	0
0	1	0
1	0	0
1	1	1

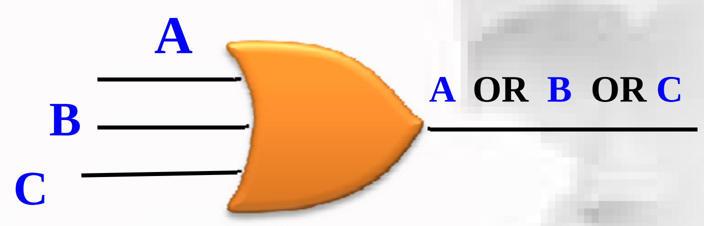


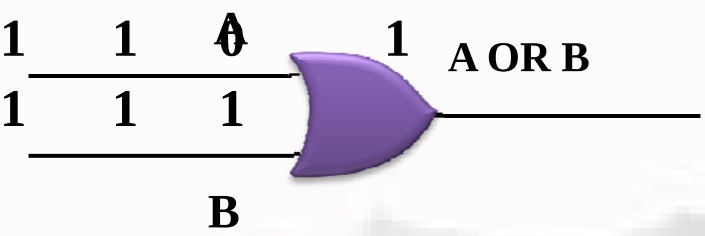
OR Gate

An **OR** gate has two or more input signals but only one output signal. **<u>If any input is high then output is high.</u>**

A	B	C	A OR B OR C(A+B+C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A	B	A OR B (A+B)
0	0	0
0	1	1
1	0	1
1	1	1

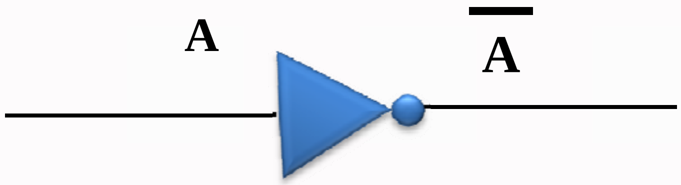




NOT Gate

A **NOT** gate has one input signal and output signal complement to the input signal. <u>**If the input is high then output is low.**</u>

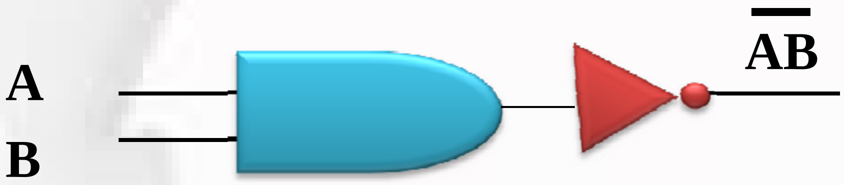
A	**Complement A** (A)
0	**1**
1	**0**



NAND Gate

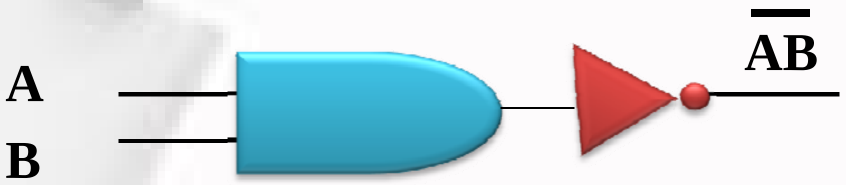
A **NAND** gate means **NOT followed by AND.**

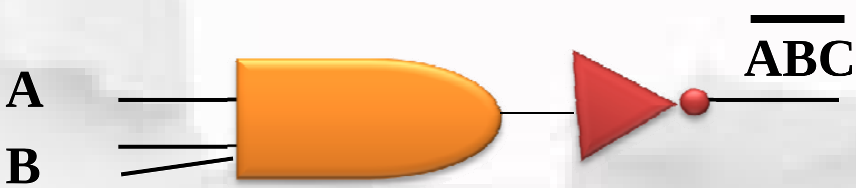
A	B	A AND B(AB)	A NAND B(AB)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



A **NAND** gate means **NOT followed by AND.**

A	B	A AND B(AB)	A NAND B(AB)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

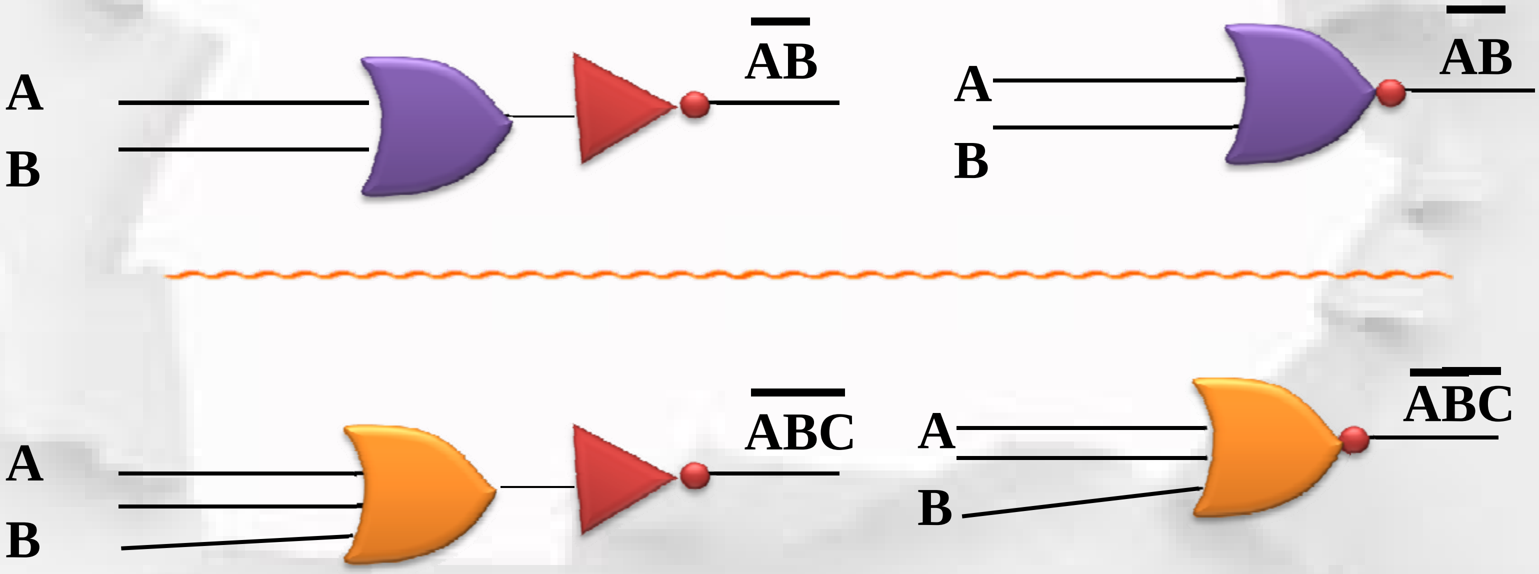




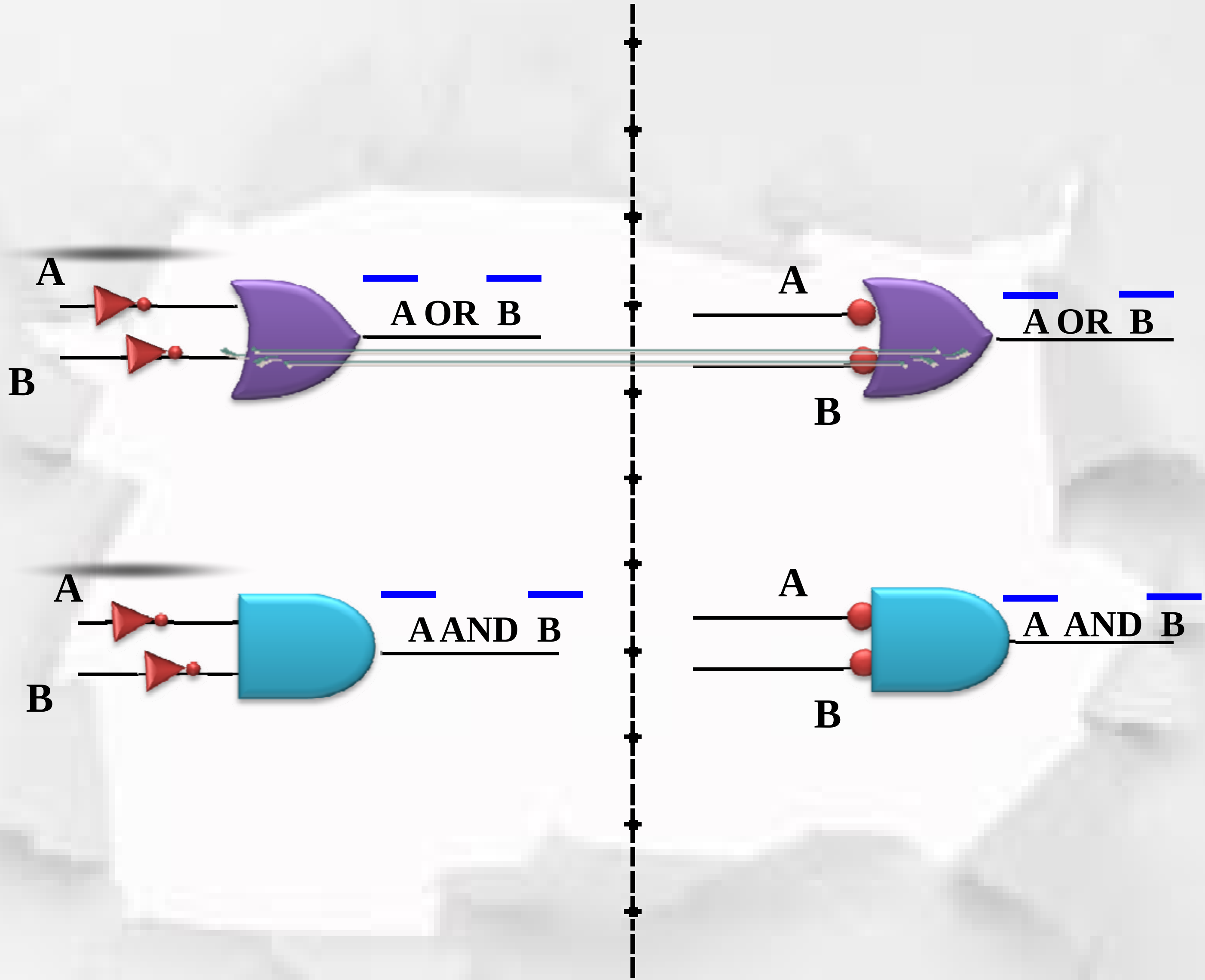
NOR Gate

A **NOR** gate means **NOT followed by OR.**

A	B	A OR B(A+B)	A NOR B(A+B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



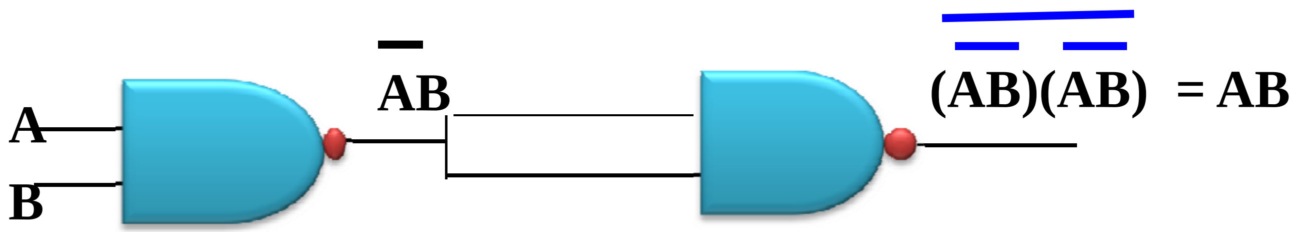
Bubbled Gates



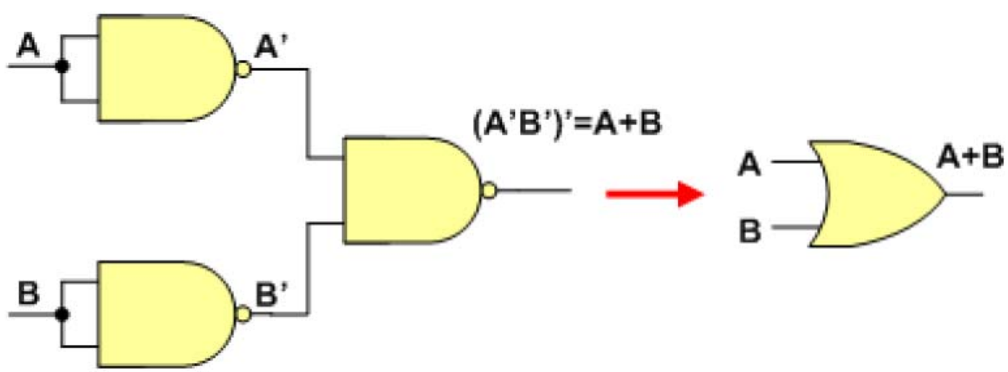
NAND Gate

as NOT gate

as AND gate



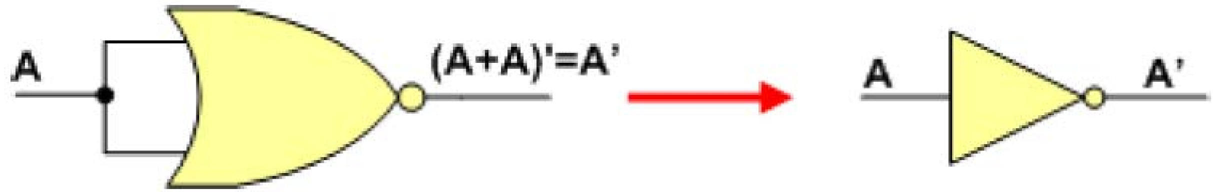
as OR gate



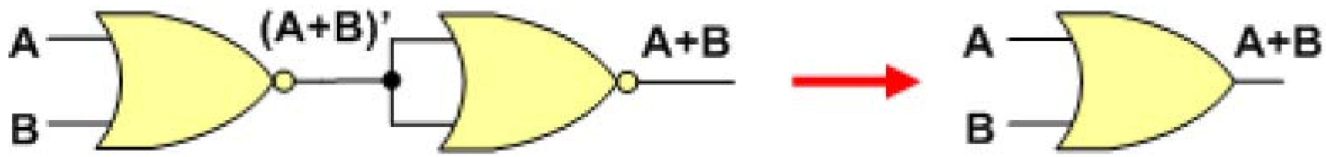
Universal Gate

NOR Gate

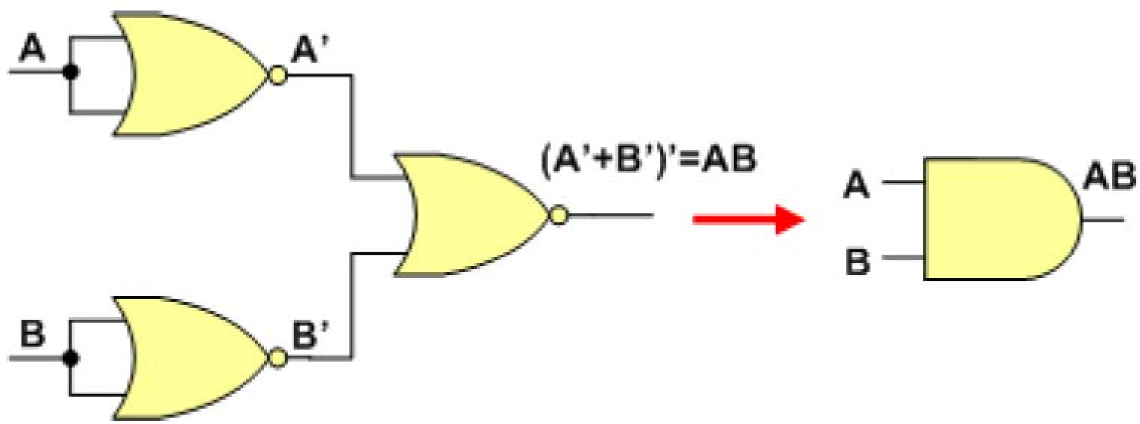
as Not gate



as OR gate



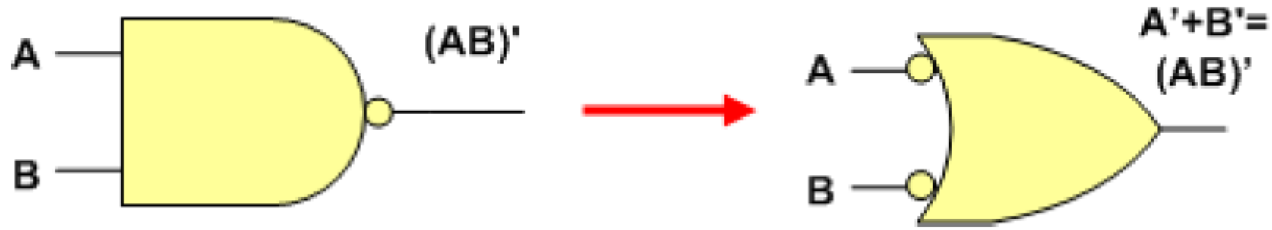
as AND gate



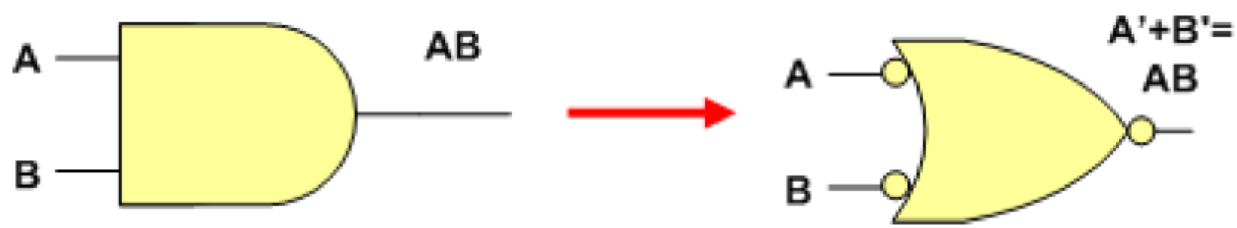
http://faculty.kfupm.edu.sa/COE/abouh/Lesson2_6.pdf

Equivalent Gate/Circuits

A NAND gate is equivalent to an inverted-input OR gate.



An AND gate is equivalent to an inverted-input NOR gate.

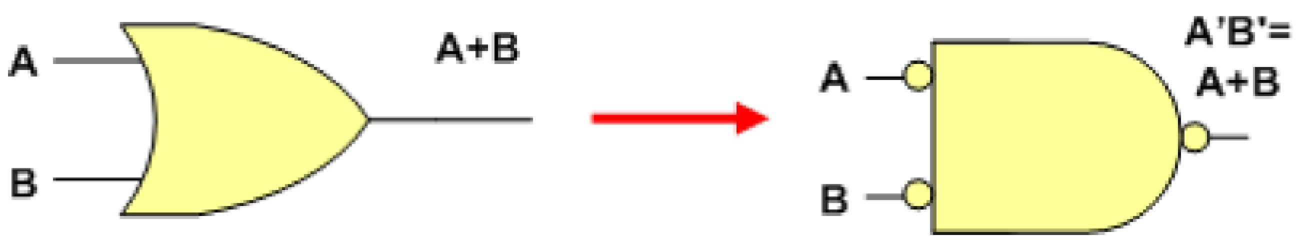


http://faculty.kfupm.edu.sa/COE/abouh/Lesson2_6.pdf

Equivalent Gate/Circuits

A NOR gate is equivalent to an inverted-input AND gate.

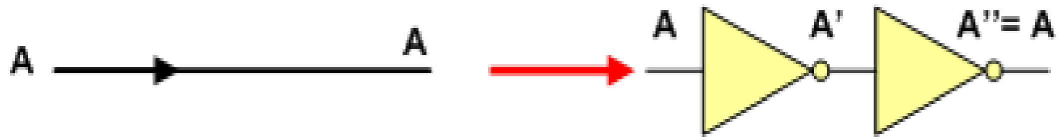
An OR gate is equivalent to an inverted-input NAND gate.



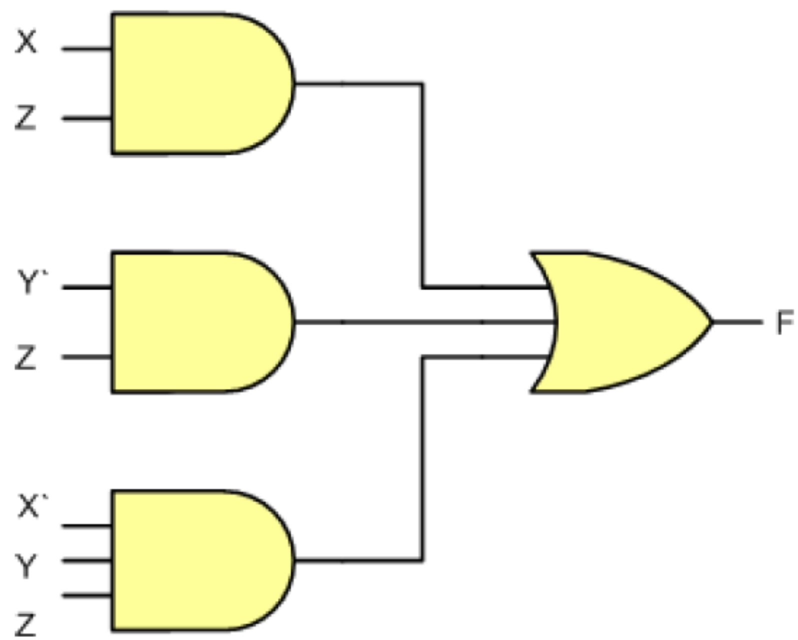
http://faculty.kfupm.edu.sa/COE/abouh/Lesson2_6.pdf

Equivalent Gate/Circuits

Two NOT gates in series are same as a buffer because they cancel each other as A'' = A.



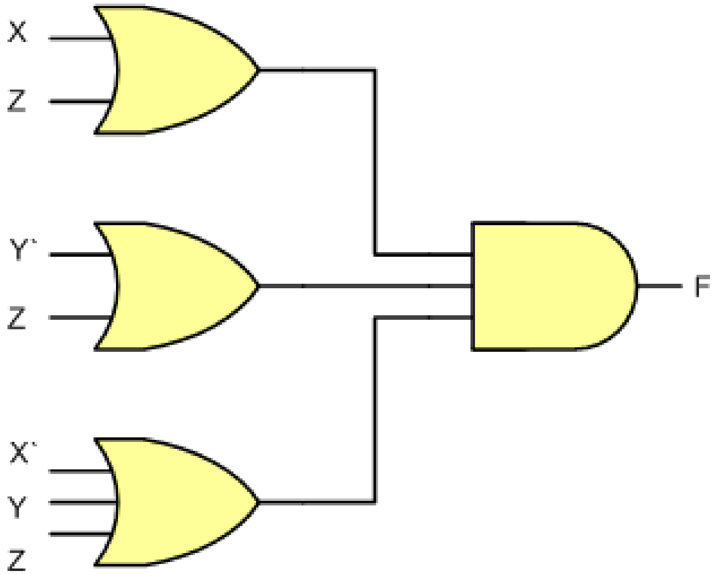
Implement F = XZ + Y'Z + X'YZ



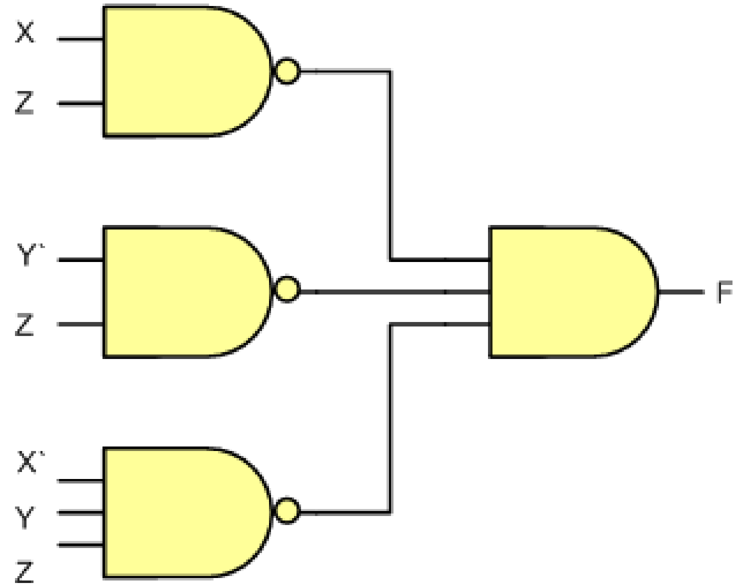
http://faculty.kfupm.edu.sa/COE/abouh/Lesson2_6.pdf

Equivalent Gate/Circuits

Implement F = (X+Z) (Y'+Z) (X'+Y+Z)



Implement F = (ZX+Y'Z +X'YZ)'

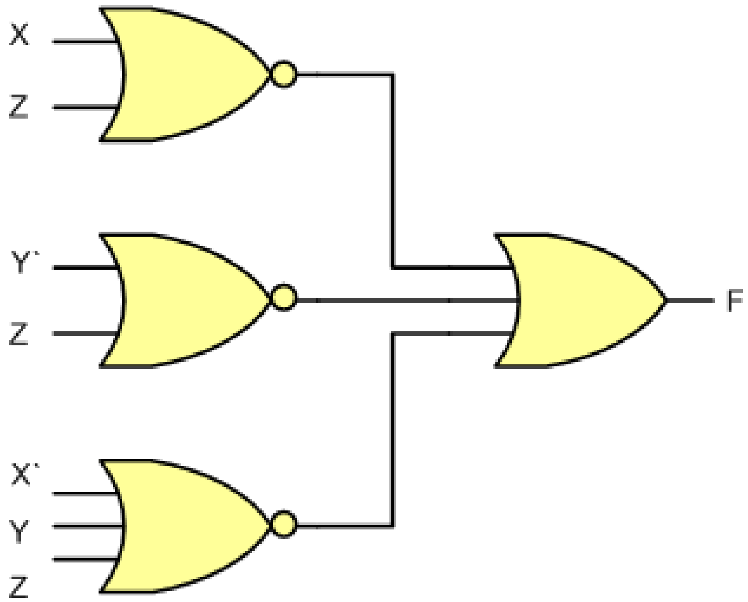


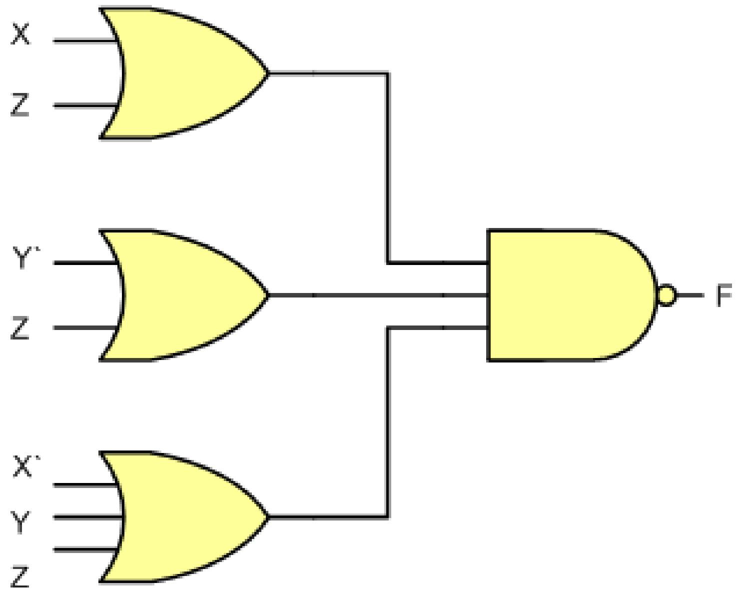
http://faculty.kfupm.edu.sa/COE/abouh/Lesson2_6.pdf

Equivalent Gate/Circuits

$$F = \overline{(X+Z).(\overline{Y}+Z).(\overline{X}+Y+Z)} \quad \textbf{or}$$

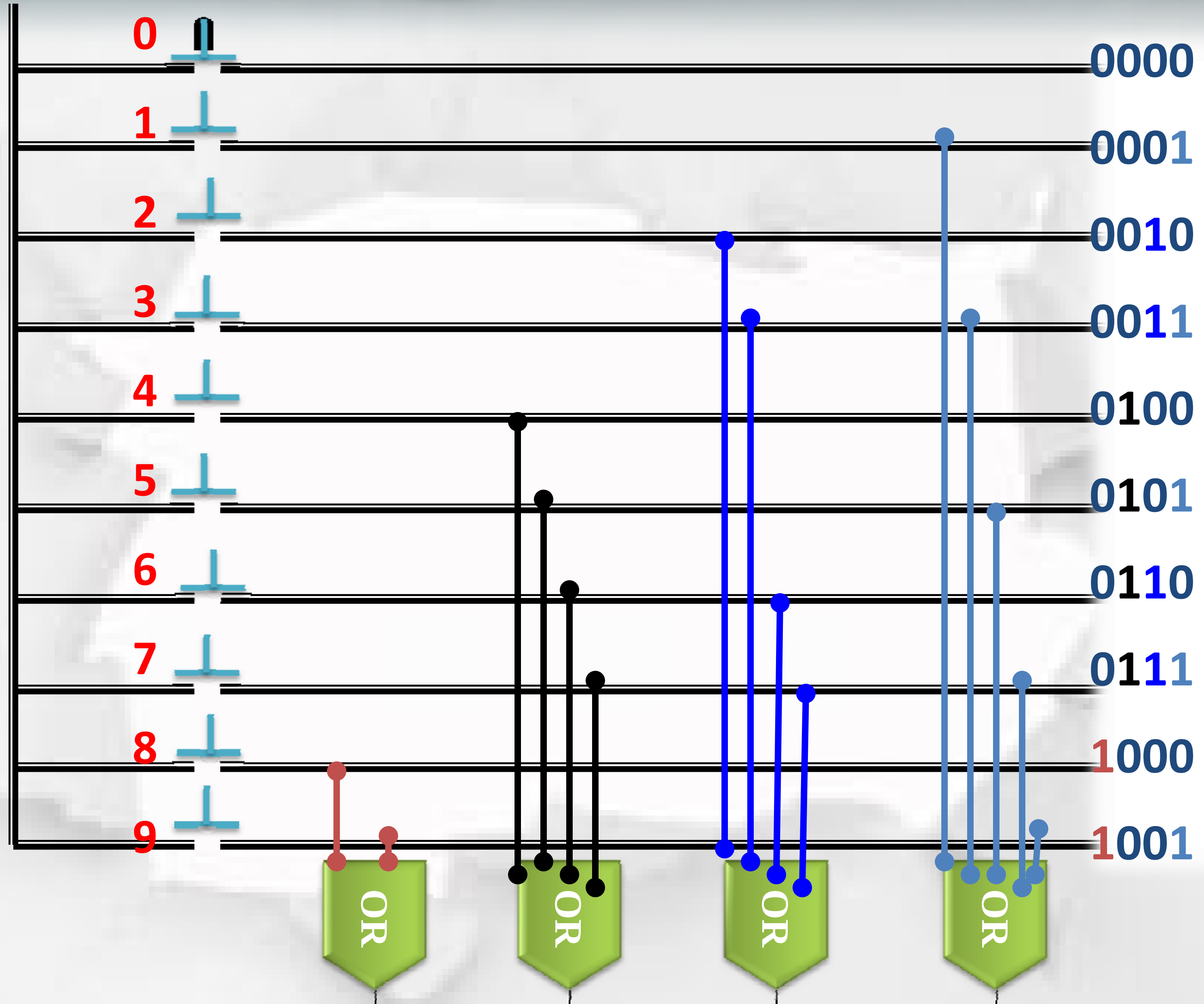
$$\overline{\boldsymbol{F}} = (\boldsymbol{X}+\boldsymbol{Z})(\overline{\boldsymbol{Y}}+\boldsymbol{Z})(\overline{\boldsymbol{X}}+\boldsymbol{Y}+\boldsymbol{Z})$$



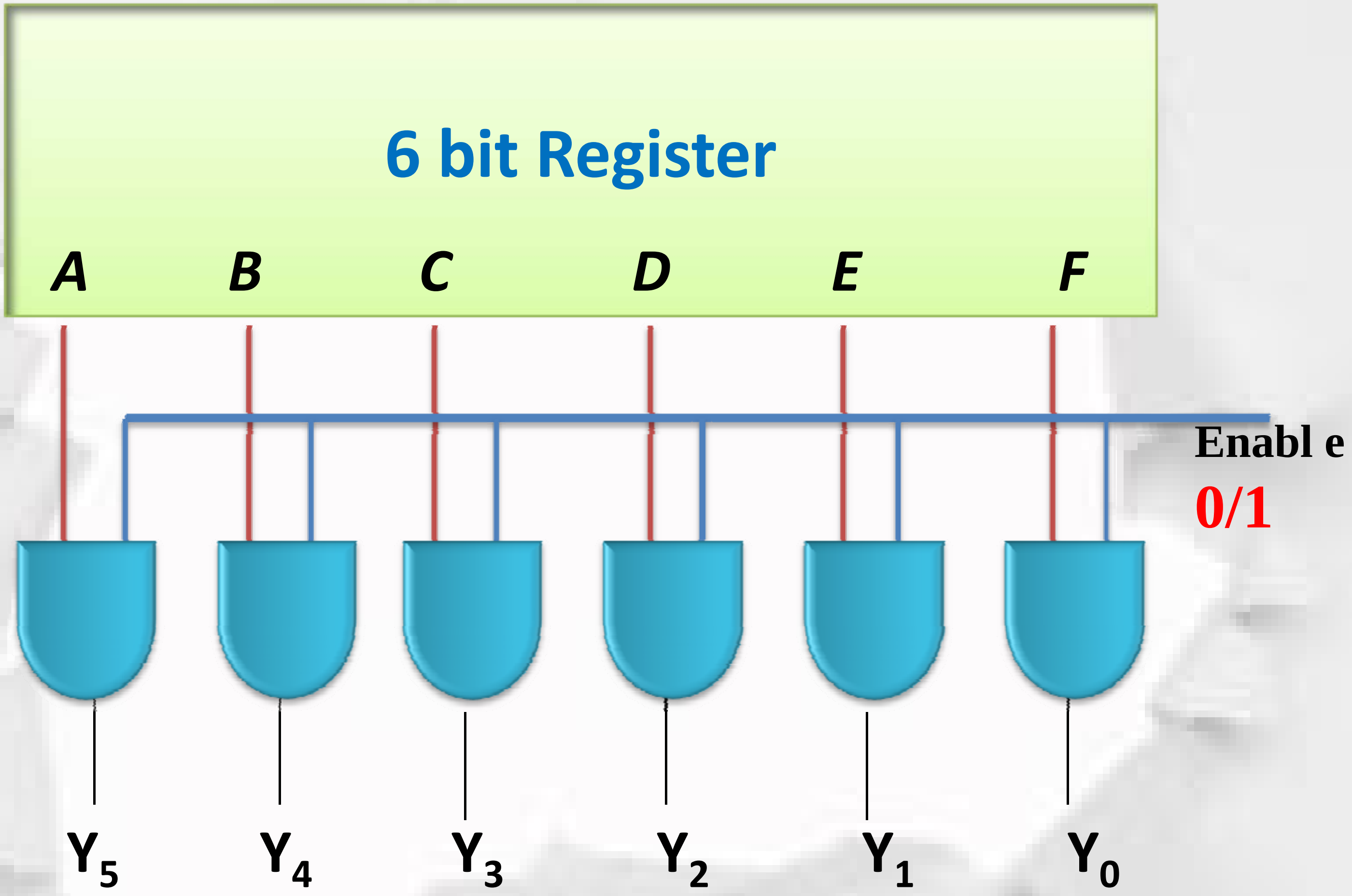


http://faculty.kfupm.edu.sa/COE/abouh/Lesson2_6.pdf

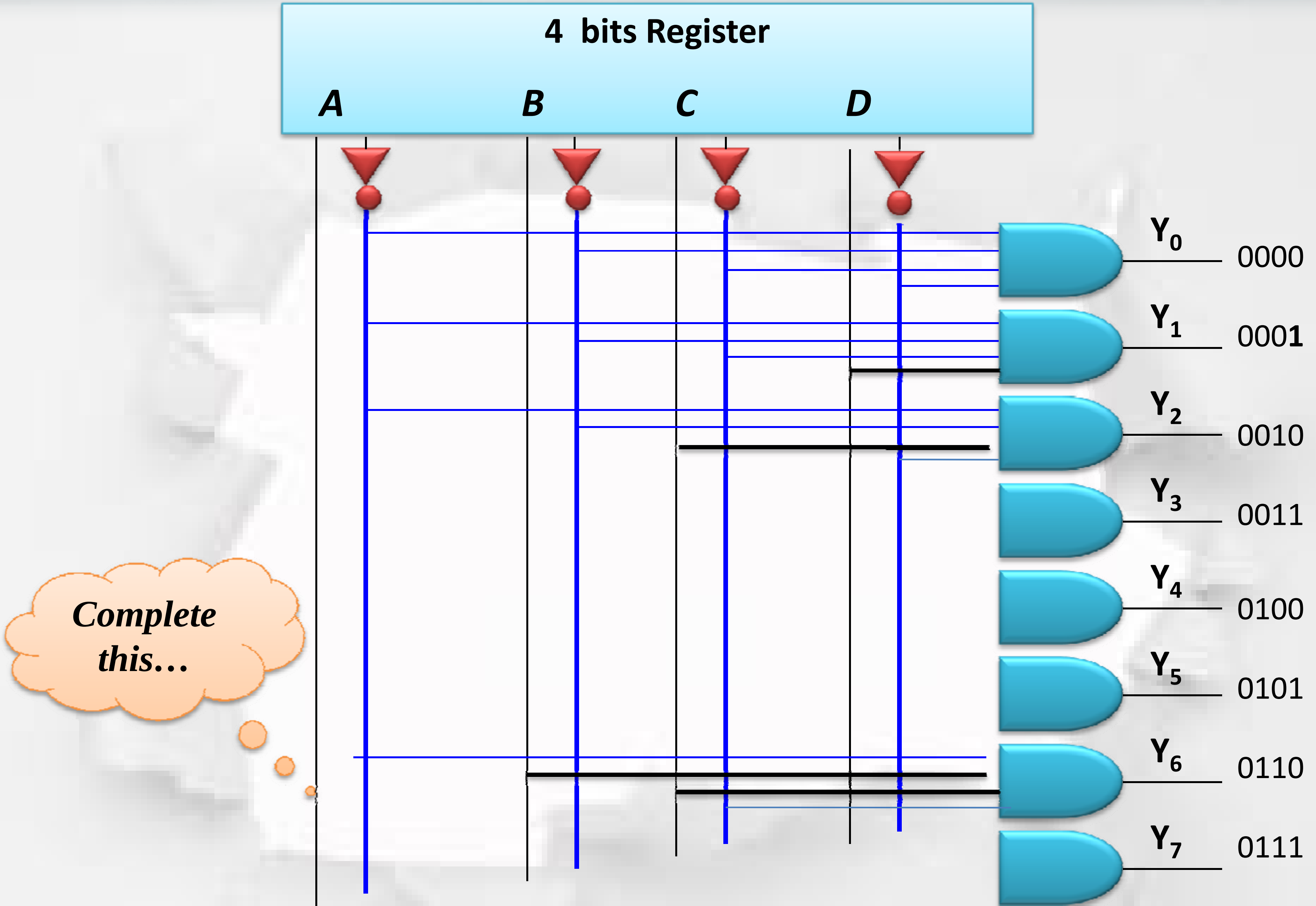
Decimal to Binary encoder



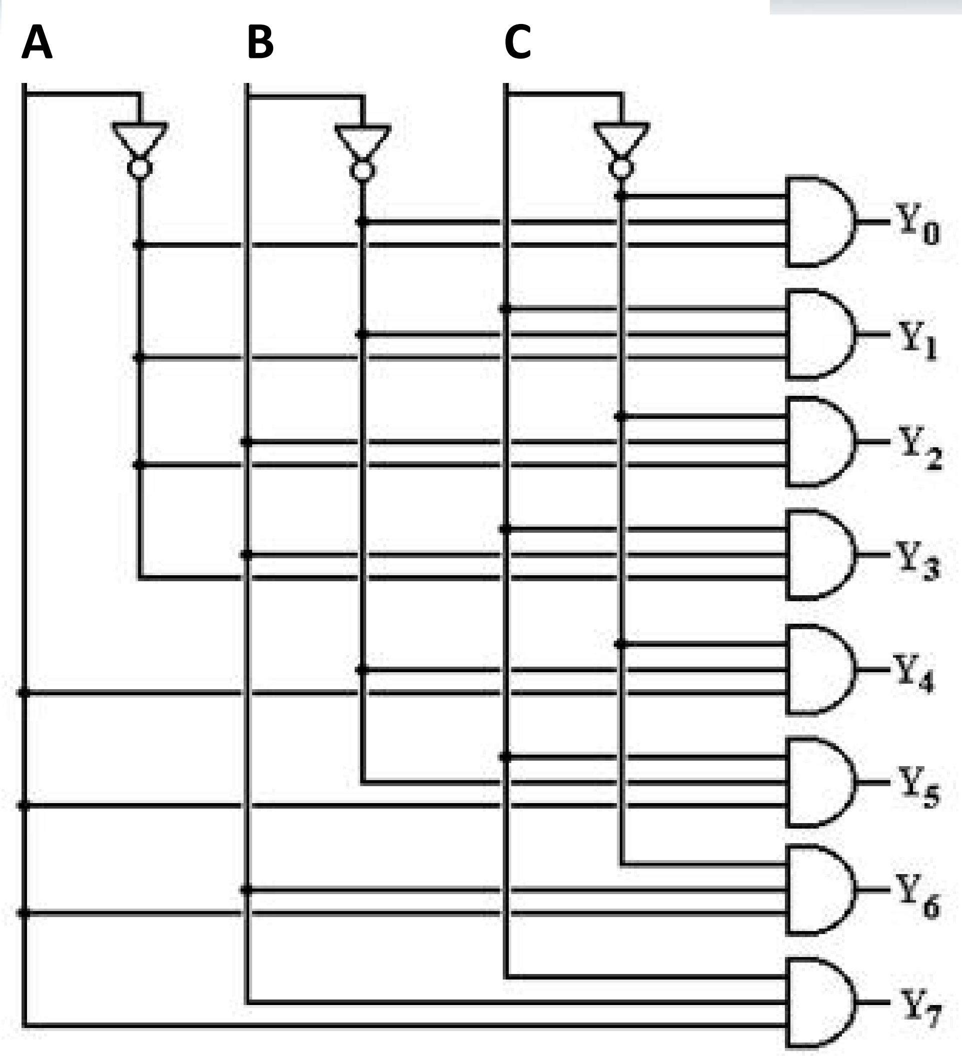
Block/Transmit 6 bit word



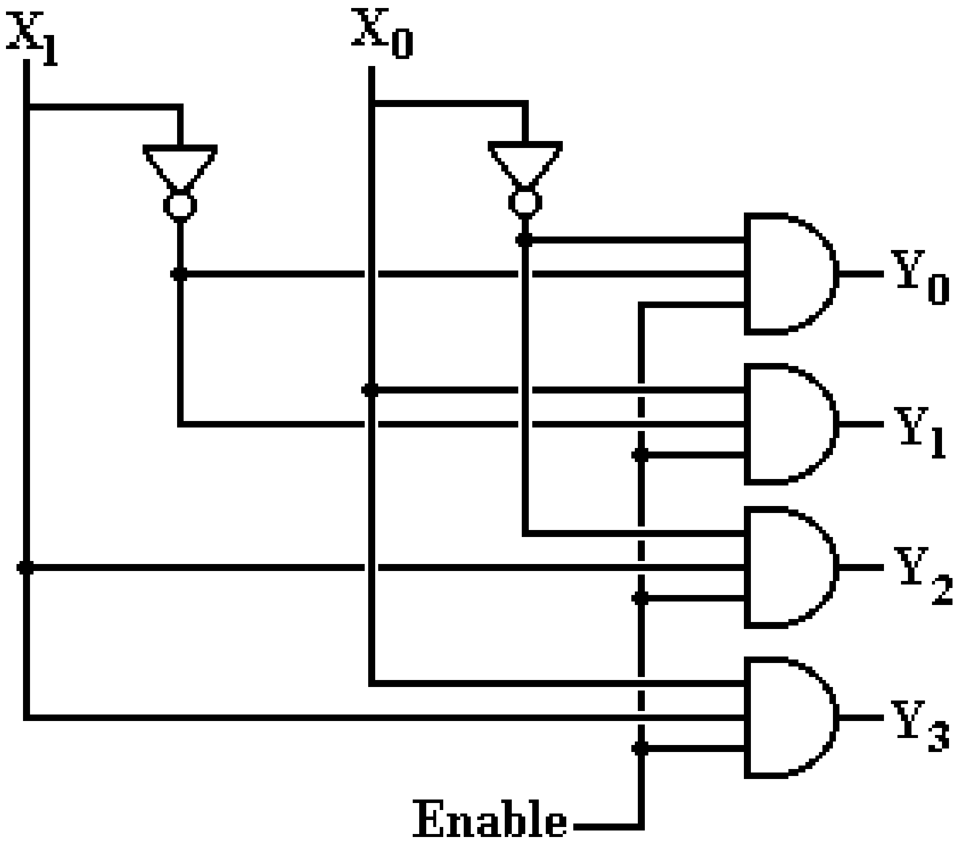
1 of 10 decoder

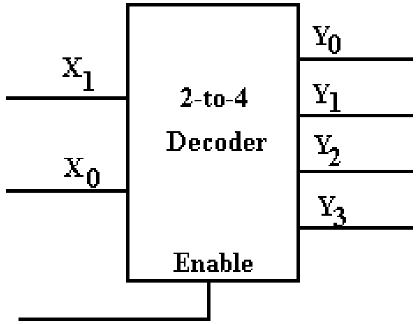


3 to 8 decoder



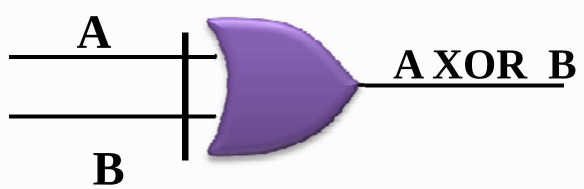
2 to 4 decoder





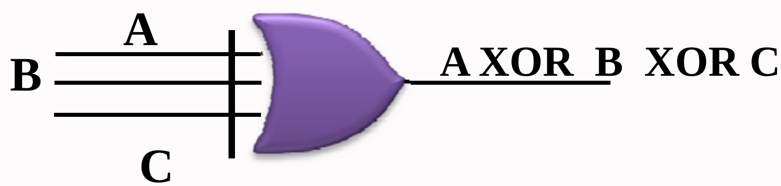
XOR Gate

A	B	$\bar{A}B+$	$A\bar{B}$	A XOR B($A \oplus B$)
0	0	0	0	0
0	**1**	**1**	**0**	**1**
1	0	0	1	1
1	**1**	**0**	**0**	**0**



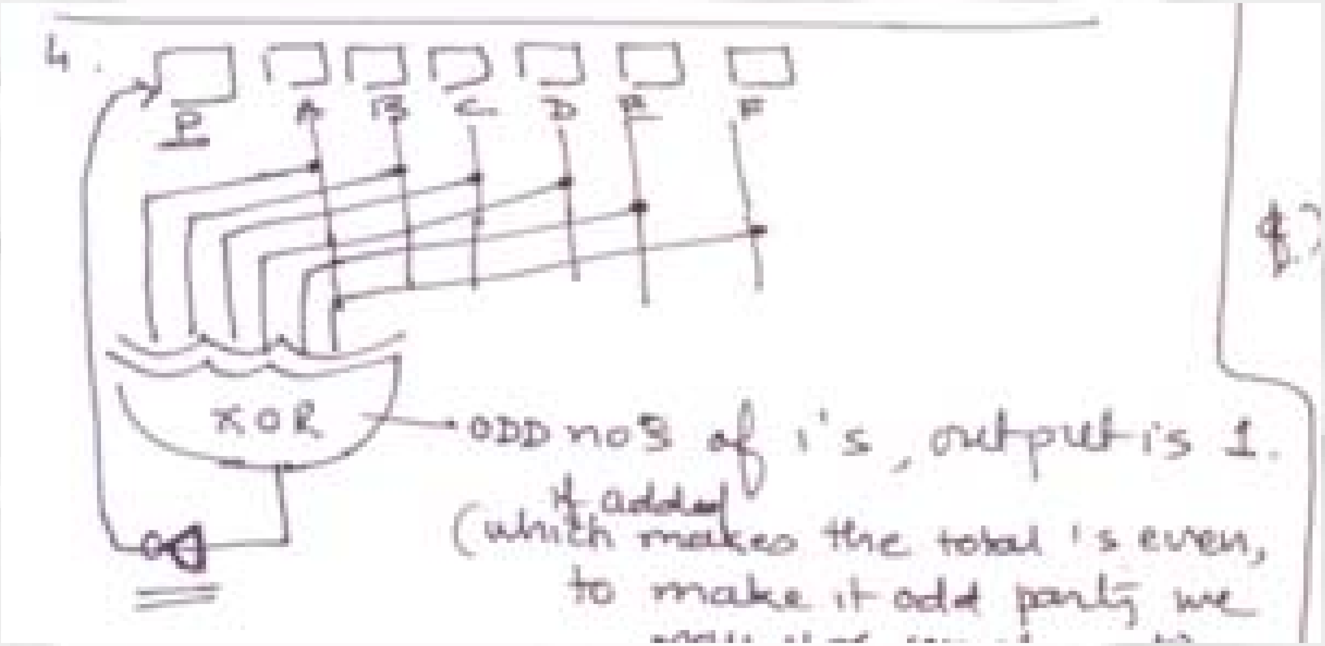
XOR Gate

$$A \oplus B \oplus C = \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\,B\,C$$



- **Draw logic circuit and truth table for the same.**
- **If number of 1’s are odd what will be the output?**

odd parity generator for a six bit word



Can you draw odd parity tester for six bit word?

Practice

Draw logic circuits for followings:

- $\overline{A}$ OR B or $\overline{A} + B$
- A OR $\overline{B}$
- $\overline{A}$ OR $\overline{B}$
- $\overline{A}$ AND B or $\overline{A}\ B$
- A AND $\overline{B}$
- $\overline{A}$ AND $\overline{B}$

Check $\overline{A + B} = \overline{A}\ \overline{B}$ using truth table.

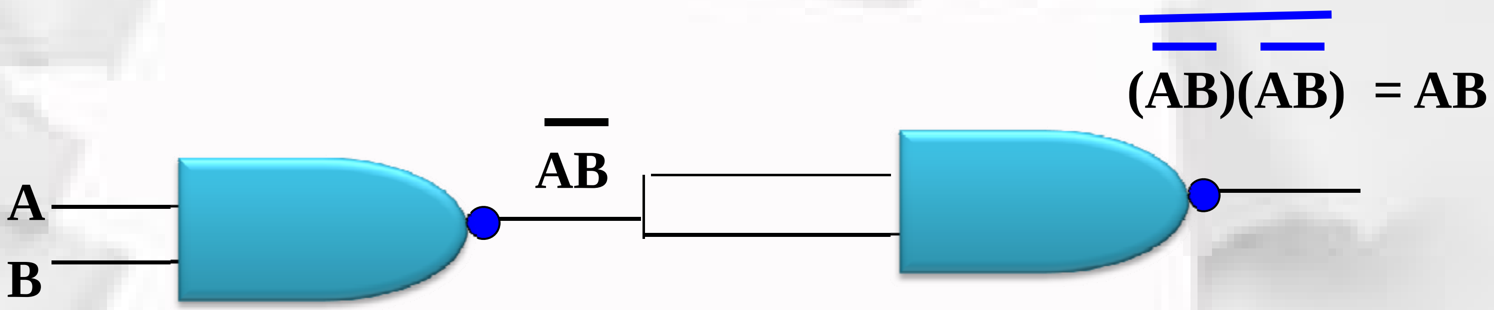
De Morgan's Theorem 1

- $\overline{A+B} = \overline{A}\ \overline{B}$
- Prove with help of truth table and draw logic circuits fro LHS and RHS.
- This can be extended for any number of variable eg.
- $\overline{A+B+C+D} = \overline{A}\ \overline{B}\ \overline{C}\ \overline{D}$
- $\overline{A+BC+DEF} = \overline{A}\ \overline{BC}\ \overline{DEF}$

De Morgan’s Theorem 2

- $\overline{AB} = \overline{A} + \overline{B}$
- Prove with help of truth table and draw logic circuits fro LHS and RHS.
- This can be extended for any number of variable eg.
- $\overline{A\,B\,C\,D} = \overline{A} + \overline{B} + \overline{C} + \overline{D}$
- $\overline{(A)\,(BC)\,(DEF)} = \overline{(A)} + \overline{(BC)} + \overline{(DEF)}$

Realization of AND using NAND



- Try to realize OR gate and NOT gate using NAND gate
- Try to realize AND, OR gate and NOT gate using NOR gate

Boolean Relations

- Commutative Law:
  - A+B=B+A
  - AB=BA
- **Associative Law:**
  - A+ (B+C) = (A+B) +C
  - A (BC) = (AB)C
- **Distributive Law:**
  - A(B+C)=AB+AC

Boolean Relations

- $A+0=A$
- $A+A=A$
- $A+1=1$
- $A+\overline{A}=1$
- $A0=0$
- $AA=A$
- $A1=A$
- $A\overline{A}=0$

$$\overline{\overline{A}} = A$$

Prove that A+BC=(A+B)(A+C)

- RHS=
- (A+B)(A+C)
- AA +AC+AB+BC
- A +AC+AB + BC
- A(1+C+B) + BC (adding 1 into anything is 1)
- A(1) + BC
- A+BC
- =LHS

Prove that A(B+C)=AB+AC with help of truth table

A	B	C	B+C	A(B+C)	AB	AC	AB+AC
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**