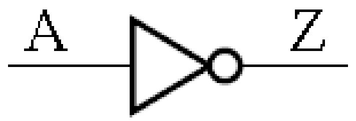
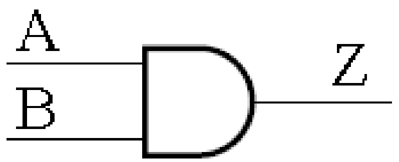
Quick Revision

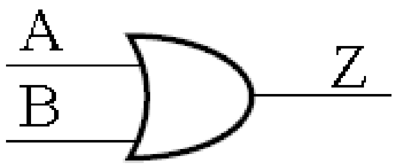
Elementary Logic Gates

Name	Inverter (NOT Gate)	AND Gate	OR Gate

Symbol







Truth Table

Inverter (NOT Gate):

A	Z
0	1
1	0

AND Gate:

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate:

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Logic Equation

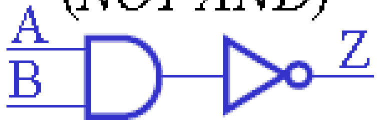
$Z = A' = \overline{A}$

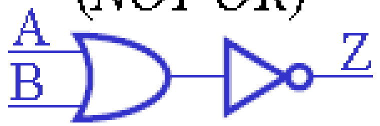
$Z = A \bullet B = AB$

$Z = A + B$

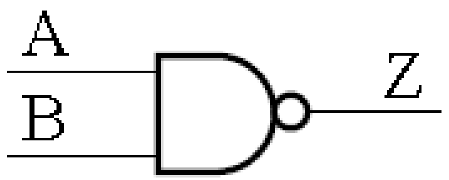
Other Elementary Logic Gates

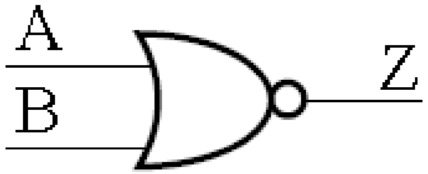
NAND Gate (*NOT AND*)	*Name*	NOR Gate (*NOT OR*)





Symbol





Truth Table

NAND:

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR:

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

$Z = (A \bullet B)' = \overline{AB}$ *Logic Equation* $Z = (A + B)' = \overline{A+B}$

Using Truth Table to Prove theorem

- DeMorgan's Theorems

T8a: $(X+Y)' = X' \bullet Y'$

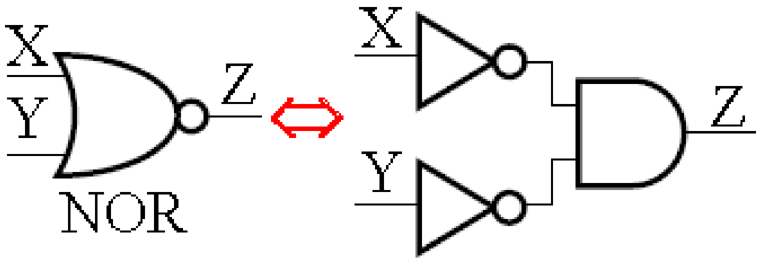
a NOR gate is equivalent to an AND gate with inverted inputs

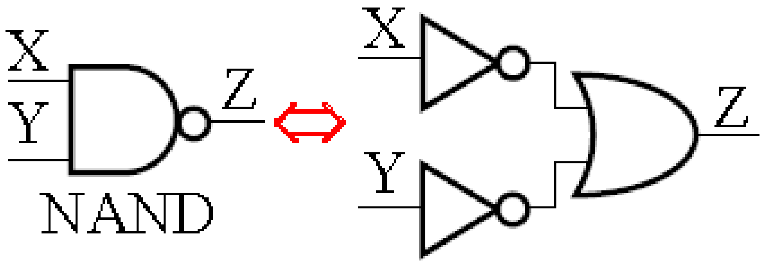
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

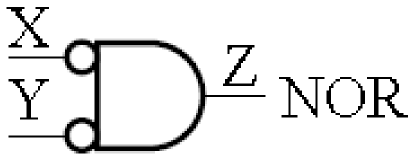
T8b: $(X \bullet Y)' = X' + Y'$

a NAND gate is equivalent to an OR gate with inverted inputs

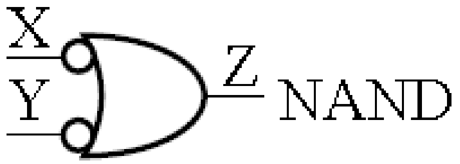
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0







alternate logic symbols



Other Logic Gates

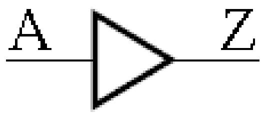
Name

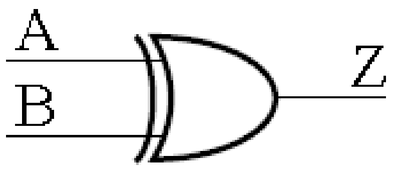
Buffer

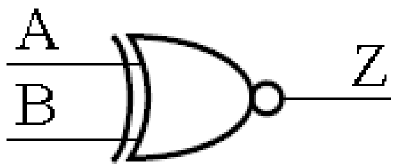
Exclusive-OR Gate
aka XOR Gate

Exclusive-NOR Gate
aka XNOR or NXOR Gate

Symbol







Truth Table

A	Z
0	0
1	1

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

Logic Equation

$Z = A$

$Z = A \oplus B = \bar{A}B + A\bar{B}$

$Z = \overline{A \oplus B} = \bar{A}\,\bar{B} + AB$

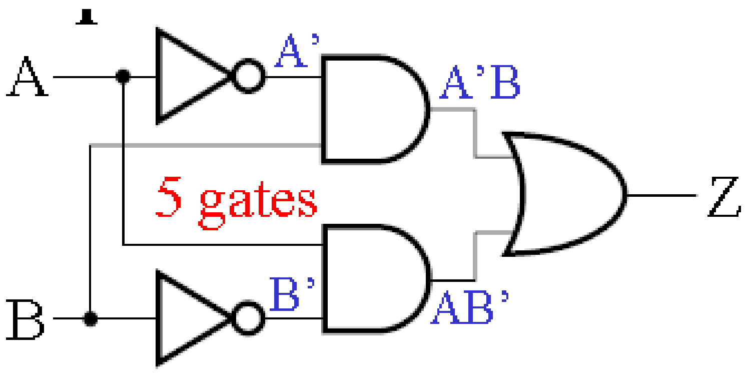
also denoted $Z = A \odot B$

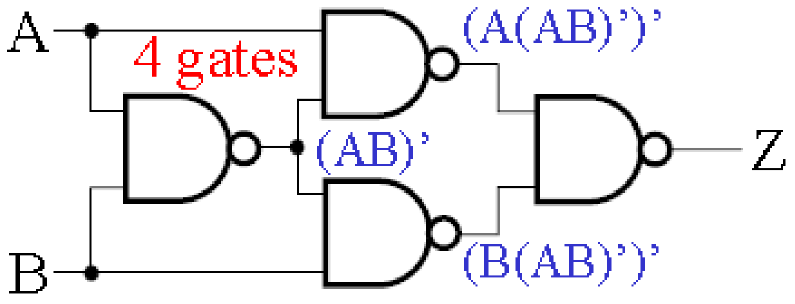
Interesting Properties of XOR

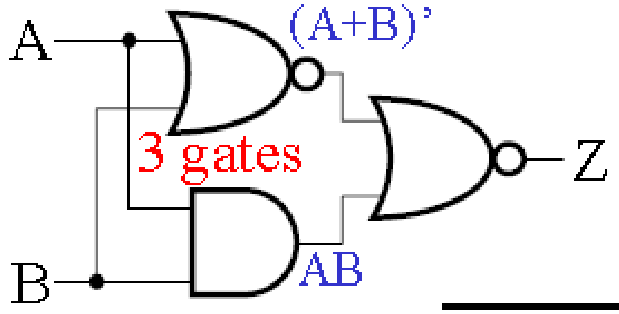
- Controlled inverter
  - $X \oplus 0 = X$
  - $X \oplus 1 = X'$
- XOR with one input inverted = XNOR
  - $X \oplus Y' = X' \oplus Y = (X \oplus Y)'$
- XNOR with one input inverted = XOR
  - $(X \oplus Y')' = (X' \oplus Y)' = X \oplus Y$
- Constant output
  - $X \oplus X = 0$
  - $X \oplus X' = 1$

Exclusive-OR Implementation

- Z=A'B+AB'
- XOR & XNOR not considered elementary logic gates by many designers







Z=((A(AB)')'(B(AB)')')'

$= \overline{\overline{A\,\overline{AB}} \bullet \overline{B\,\overline{AB}}} = \overline{\overline{A\,\overline{AB}}} + \overline{\overline{B\,\overline{AB}}}$

$= A(\overline{A}+\overline{B}) + B(\overline{A}+\overline{B})$

= AA'+AB'+A'B+BB'=AB'+A'B

Z=((A+B)'+AB)' $= \overline{\overline{A+B}+AB}$

$= \overline{\overline{(A+B)}}\bullet\overline{AB} = (A+B)(\overline{A}+\overline{B})$

= AA'+AB'+A'B+BB'

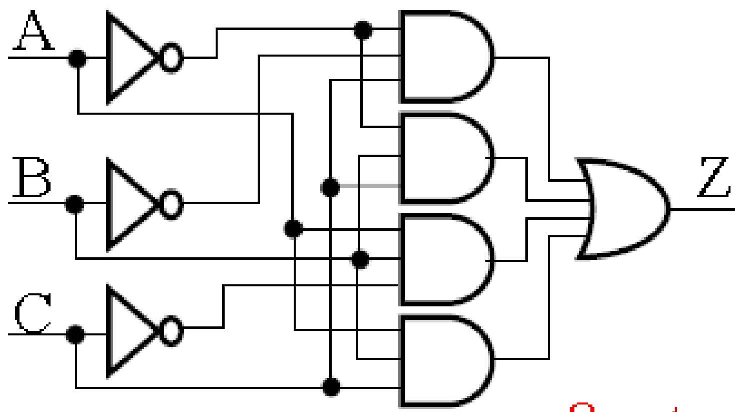
= 0+AB'+A'B+0 = AB'+A'B

Gate-level Representation

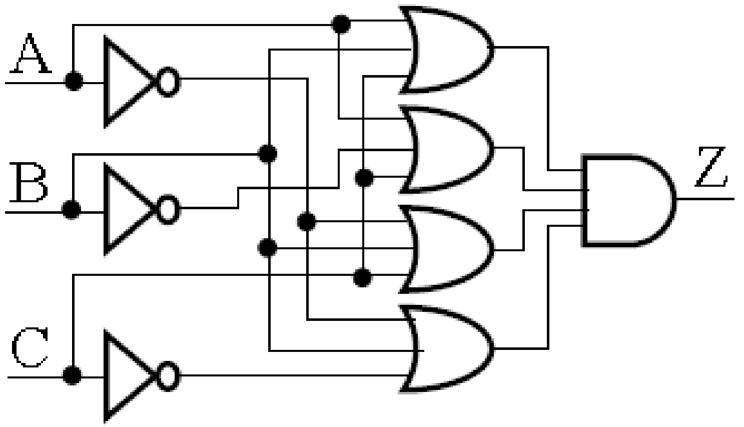
- SOP expressions
  - AND-OR
    - With inverters for complemented literals

      Z=A’B’C+A’BC+ABC’+ABC
  - aka 2-level AND-OR logic representation
- POS expressions
  - OR-AND
    - With inverters for complemented literals

      Z=(A+B+C)•(A+B’+C)•(A’+B+C)•(A’+B+C’)
  - aka 2-level OR-AND logic representation



8 gates



Common Combinational Logic Circuits

- Adders
  - Subtraction typically via 2s complement addition
- Multiplexers
  - *N control signals select 1 of up to 2N inputs as output*
- Demultiplexers
  - *N control signals select input to go to 1 of up to 2N outputs*
- Decoders
  - *N inputs produce M outputs (typically M > N)*
- Encoders
  - *N inputs produce M outputs (typically N > M)*
- Converter (same as decoder or encoder)
  - *N inputs produce M outputs (typically N = M)*

More Common Circuits

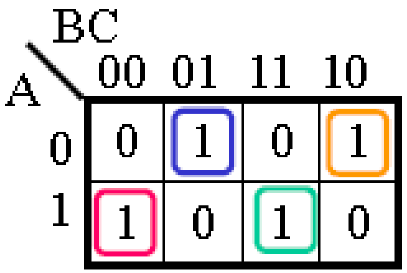
- Comparators
  - Compare two *N-bit binary values*
    - Equal-to or Not-equal-to
      - Easiest to design
    - Greater-than, Less-than, Greater-than-or-equal-to, etc.
      - Require adders
- Parity check/generate circuit
  - Calculates even or odd parity over *N bits of data*
  - Checks for good/bad parity (parity errors) on incoming data

Adder

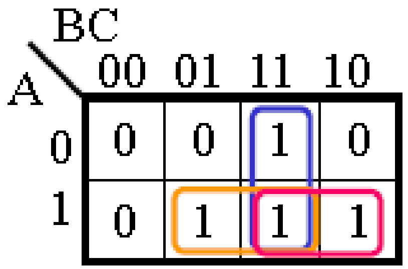
- Consider *i*th column addition of 2 binary numbers (A and B)
  - $A_i + B_i + Cin_i = Cout_i + Sum_i$
  - Derive truth table
  - Populate K-maps
  - Obtain minimized SOPs
  - Draw logic diagram
  - Optimize with P&Ts

Truth Table

A	B	C	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

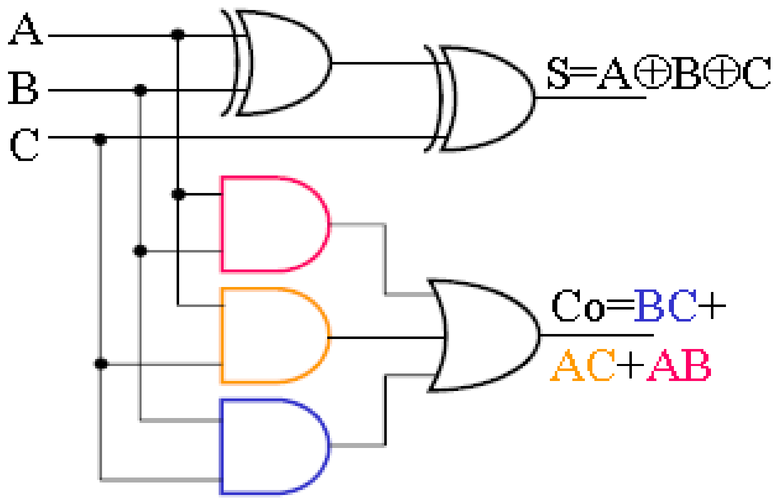


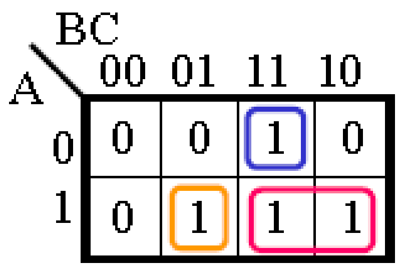
$S = A'B'C + A'BC' + AB'C' + ABC$
$= A'(B \oplus C) + A(B \oplus C)'$
$= A \oplus B \oplus C$



$Co = BC + AC + AB$

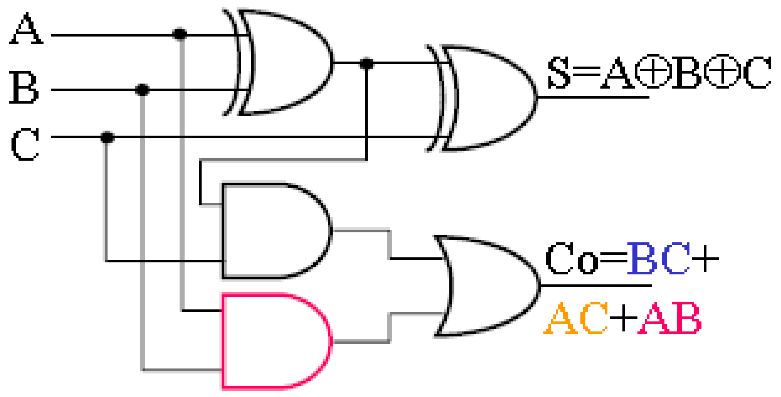
Adder





A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

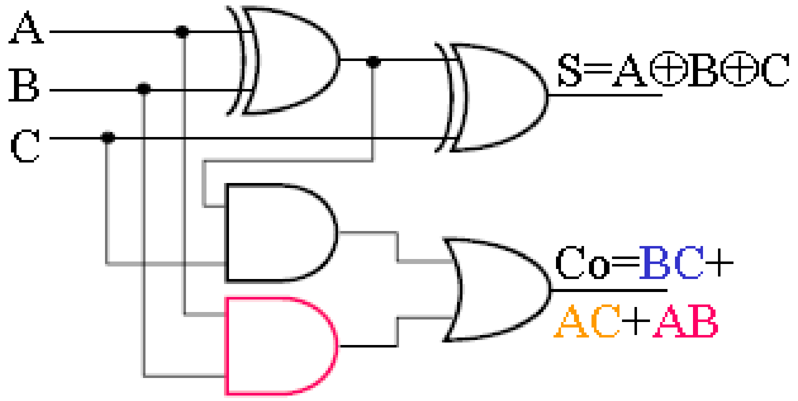
Co=A'BC+AB'C+AB
=C(A'B+AB')+AB
=C(A⊕B)+AB



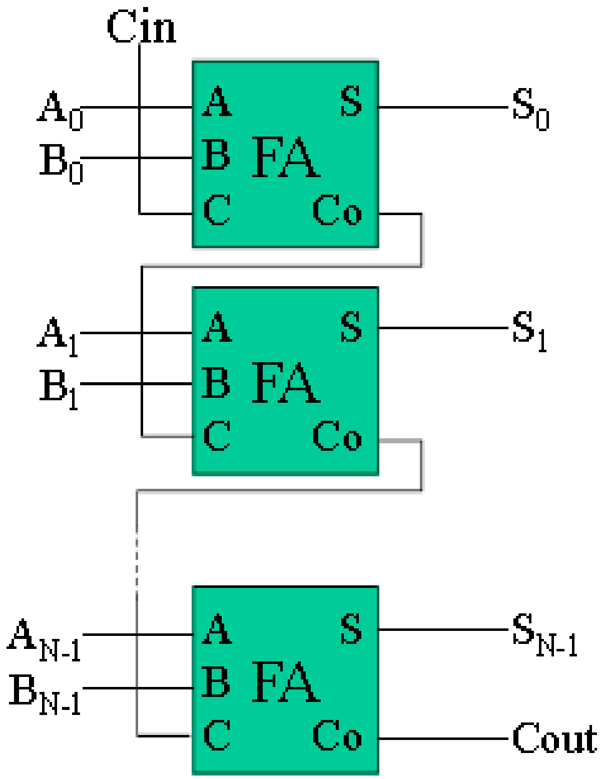
Taking advantage of common product terms between S and Co we see that we can use the XOR gate for A⊕B to reduce the gate count

Adder

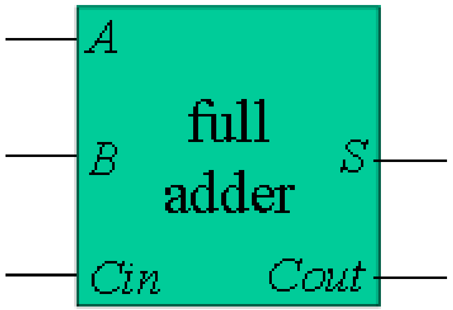
referred to as a ***full adder***



now we can build an *N*-bit adder from *N* full adders



we can let a block represent the full adder



QUESTIONS?