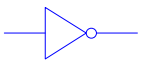
Logic Circuits

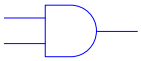
Elementary Logic Rules and Theorems

- Boolean algebra
- Axioms
  - commutativity: a ET b=b ET a, a OU b=b OU a
  - associativity: a ET (b ET c)=(a ET b) ET c, a OU (b OU c)=(a OU b) OU c
  - distributivity: a ET (b OU c)=a ET b OU a ET c, a OU (b ET c)=(a OU b) ET (a OU c)
  - identity element: VRAI ET a=a ET VRAI=a, FAUX OU a=a OU FAUX=a
  - complement: a ET a'=FAUX, a OU a'=VRAI
- Theorems
  - a ET FAUX=FAUX, a OU VRAI=VRAI
  - a ET (a OU b)=a, a OU (a ET b)=a
  - a ET a=a, a OU a=a
  - (a')'=a
  - De Morgan: (a ET b)'=a' OU b', (a OU b)'=a' ET b'
- Shannon's results
- Logic information coding: TRUE/FALSE → 1/0
- Notation:
  - AND → **.**
  - OR → **+**

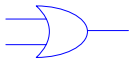
Elementary Gates Schematic



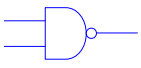
NOT



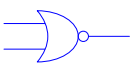
AND



OR

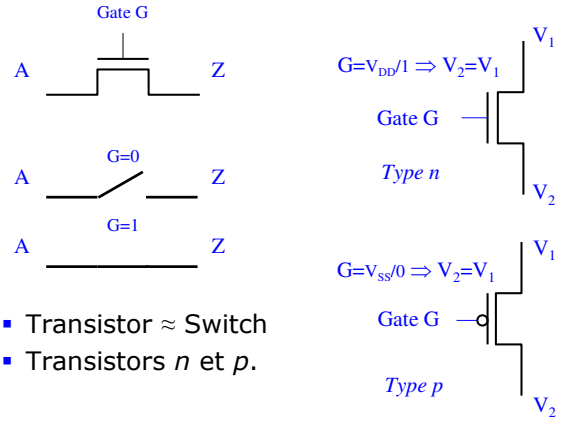


NOT-AND (*NAND*)



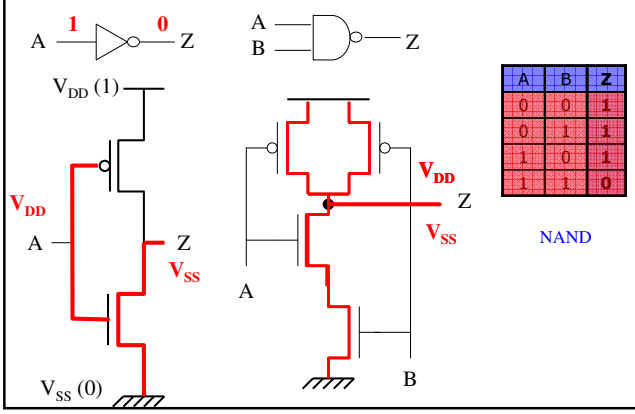
NOT-OR (*NOR*)

The Transistor



- Transistor ≈ Switch
- Transistors *n* et *p*.

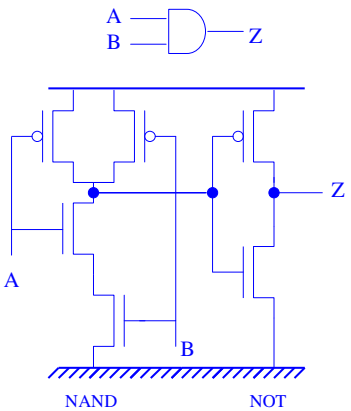
Simple Logic Functions Using Transistors



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

NAND

Simple Logic Functions Using Transistors



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

AND

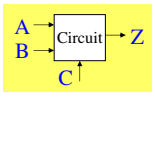
NAND NOT

Logic Circuit Design

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

Truth Table

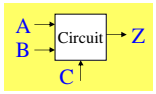
C=0 ⇔ C'
A=1 ⇔ A
C=0 AND A=1 AND B=0 ⇔ C'.A.B'
Z=1 ⇔ C'.A.B' + C'.A.B + C.A'.B + C.A.B

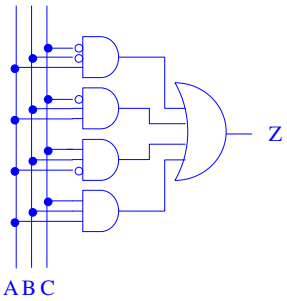


Z = C'.A.B' + C'.A.B + C.A'.B + C.A.B

- Example:
  - Input: two binary signals A and B, and control signal C; Output: Z.
  - If C=0, Z=A; If C=1, Z=B

Logic Circuit Design





Z = C'.A.B' + C'.A.B + C.A'.B + C.A.B

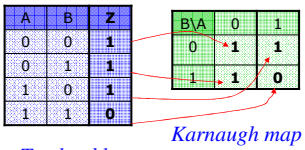
Simplifying Boolean Expressions

- Circuit cost:
  - Number of gates
  - Number of inputs (*fan-in*) and outputs (*fan-out*)
- A few simplification rules:
  - *XY+XY' = X*
  - *X+X'Y = X+Y*
  - *XY+X'Z+YZ = XY+X'Z* (consensus)

Previous example:
S=AB'C'+ABC'+A'BC+ABC
= AC'(B+B')+BC(A+A')
=AC'+BC

Karnaugh Maps

- An intuitive tool for circuits with few inputs
- Different spatial arrangement of 0s/1s than in truth table
- Group two neighbor 1s ⇔ remove a variable and a term
- Function = covering all 1s = OR of all 1s

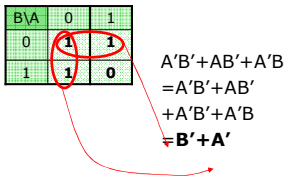


A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

Truth table

B\A	0	1
0	1	1
1	1	0

Karnaugh map

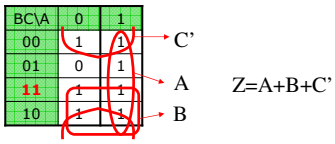


B\A	0	1
0	1	1
1	1	0

A'B'+AB'+A'B
=A'B'+AB'
+A'B'+A'B
=**B'+A'**

Karnaugh Maps with 3 and 4 variables

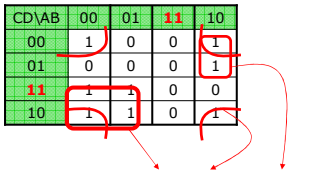
- Expression of two neighbor cells identical except for one variable
- Covering using heuristic.
- Quine-McCluskey more scalable



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

C'
A
B

Z=A+B+C'

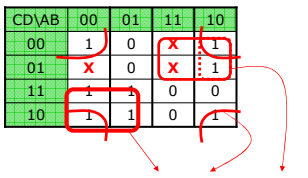


CD\AB	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	1	1	0	0
10	1	1	0	1

Z=A'C+B'D'+AB'C'

Un-Assigned Values

- Output value not defined (case does not exist)
- Notation = ***X*** or ***d*** (*don't care*).
- Can be covered if allows further simplification



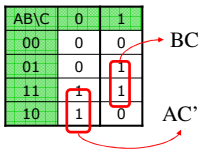
CD\AB	00	01	11	10
00	1	0	X	1
01	X	0	X	1
11	1	1	0	0
10	1	1	0	1

Z=A'C+B'D'+**AC'**

Example

C	A	B	Z
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

Truth table



Z=AC'+BC