Digital Circuits

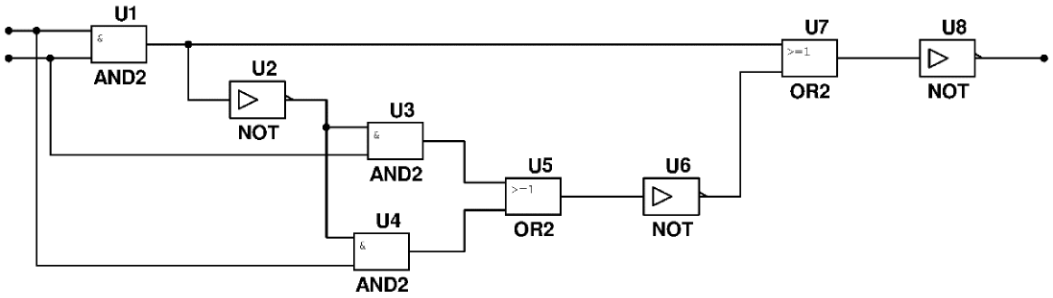
Networks and Embedded Software

Module 3.2.3

by Wolfgang Neff

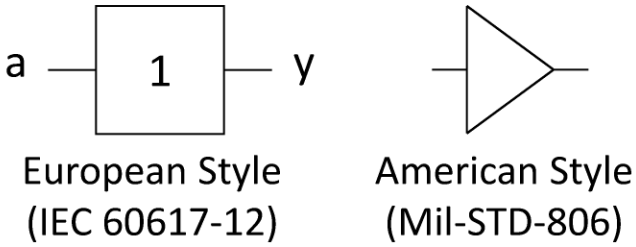
Digital Circuits (1)

- Based on Boolean algebra
  - Logical operators are represented by symbols
  - Logical values are represented voltages
    - Positive logic: 0 → Low voltage, 1 → High voltage
    - Negative logic: 0 → High voltage, 1 → Low voltage
- Graphical representation of truth functions



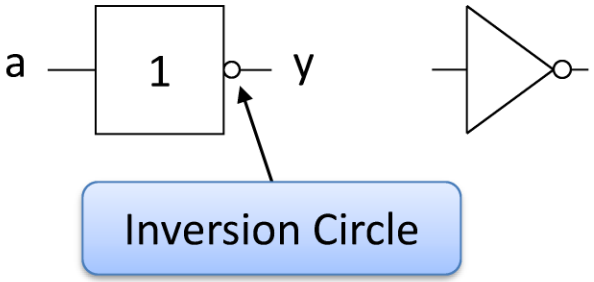
Graphic Symbols (1)

- Buffer



a	y=a
0	0
1	1

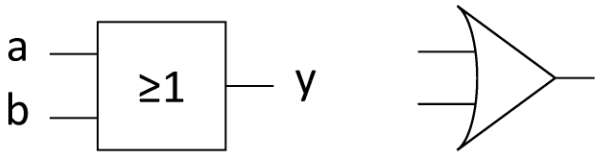
- Negation (NOT, ¬)



a	y=¬a
0	1
1	0

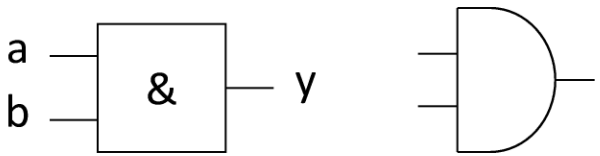
Graphic Symbols (2)

- Disjunction (OR, $\vee$)



a	b	y=a∨b
0	0	0
0	1	1
1	0	1
1	1	1

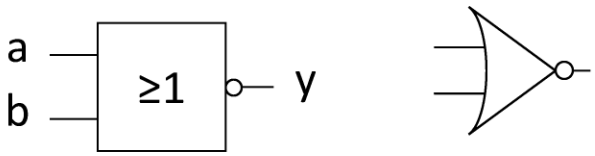
- Conjunction (AND, $\wedge$)



a	b	y=a∧b
0	0	0
0	1	0
1	0	0
1	1	1

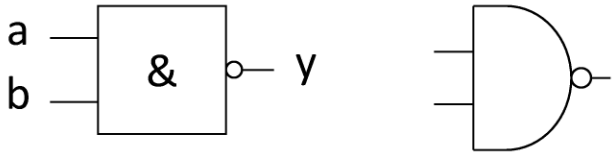
Graphic Symbols (3)

- NOR (↓)



a	b	y=a↓b
0	0	1
0	1	0
1	0	0
1	1	0

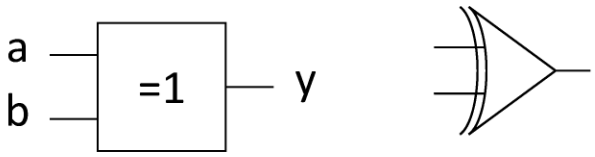
- NAND (|)



a	b	y=a|b
0	0	1
0	1	1
1	0	1
1	1	0

Graphic Symbols (4)

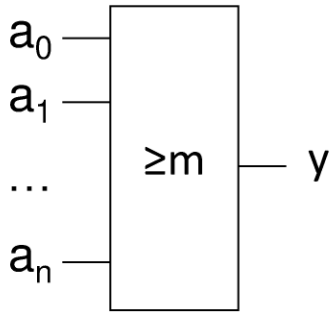
- XOR ($\oplus$)



a	b	y=a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

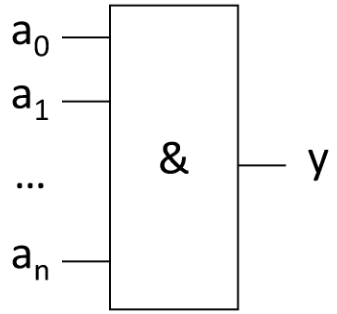
- Threshold Gate

$$(a_0, a_1, \mathrm{K})\ \alpha\ \begin{cases} 1 \text{ if at least m } a_i = 1 \\ 0 \text{ else} \end{cases}$$

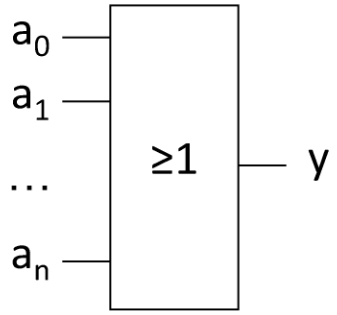


Graphic Symbols (5)

- Compound AND-Gate
  - $y = a_0 \wedge a_1 \wedge a_2 \ldots = \bigwedge_{i=0}^{n} a_i$
  - $(a_0, a_1, \mathrm{K}) \alpha \begin{cases} 1 \text{ if each } a_i = 1 \\ 0 \text{ else} \end{cases}$

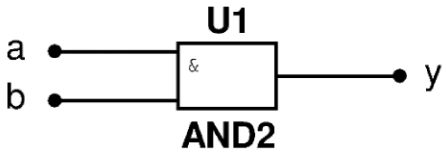


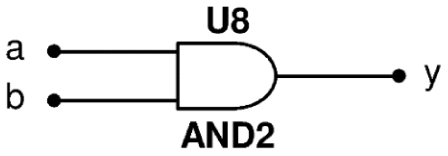
- Compound OR-Gate
  - $y = a_0 \vee a_1 \vee a_2 \ldots = \bigvee_{i=0}^{n} a_i$
  - $(a_0, a_1, \mathrm{K}) \alpha \begin{cases} 0 \text{ if each } a_i = 0 \\ 1 \text{ else} \end{cases}$



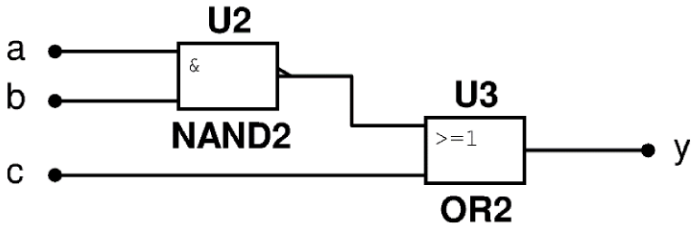
Digital Circuits (2)

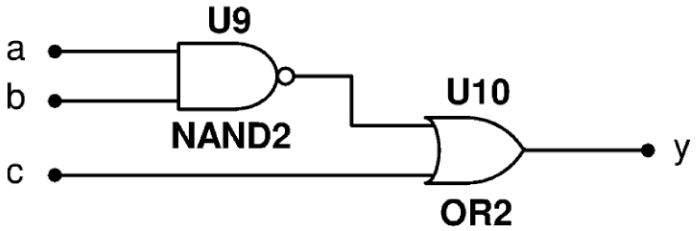
- Examples
  - y = a∧b





  - y = ¬(a∧b) ∨ c





Digital Circuits (3)

- Examples (continued)
  - y = (a∧¬b) ∨ ¬(c∧a) ∨ a