

Boolesche Terme und Schaltalgebra

1. Datentyp boolean

Eine Boolesche Variable oder ein Boolescher Ausdruck (Term) nimmt nur zwei Werte an: **True** oder **False**

(abkürzend: 1 oder 0; in Python sind **True** oder **False** zu verwenden)

Insbesondere sind folgende Terme Boolesche Ausdrücke, deren Wert sich auch einer Variablen zuweisen läßt:

8 > 5 hat den Wert **True**

7 == 8 hat den Wert **False**

7 != 8 hat den Wert **True**

x hat den Wert **True** nach der Wertzuweisung **x = 7 < 12**

x hat den Wert **False** nach der Wertzuweisung **x = (0 == 6)**

a or b hat den Wert **True** genau dann, wenn mindestens eine der Variablen **a, b** den Wert **True** hat; andernfalls hat **a or b** den Wert **False**.

Mit **a = 7 != 8** oder **a = (7 != 8)** wird in Python der Booleschen Variablen **a** der Wert des Booleschen Terms **7 != 8** (hier: **True**) zugewiesen.

Wir definieren die Verknüpfungen **and** und **or** sowie die Operation **not** jeweils über eine Wahrheitstafel:

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

a	not a
False	True
True	False

Abkürzende Schreibweisen (a, b, c sind Boolesche Variable oder Boolesche Terme):

$$a \text{ and } b = a \wedge b = a \cdot b = a b$$

$$a \text{ or } b = a \vee b = a + b$$

$$\text{not } a = \neg a = \bar{a}$$

Dabei gelte auch die aus der Algebra bekannte Vereinbarung "Punkt vor Strich", d. h.

$$a + (b \cdot c) = a + b \cdot c = a + b c$$

Die **AND**-Verknüpfung nennen wir auch **Konjunktion**,
die **OR**-Verknüpfung **Disjunktion**.

2. Rechenregeln für Boolesche Variable

Kommutativgesetz

$$(1) \quad a + b = b + a$$

$$(1') \quad a \cdot b = b \cdot a$$

Assoziativgesetz

$$(2) \quad a + (b + c) = (a + b) + c$$

$$(2') \quad a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

Distributivgesetz

$$(3) \quad a \cdot (b + c) = a \cdot b + a \cdot c$$

$$(3') \quad a + b \cdot c = (a + b) \cdot (a + c)$$

Absorptionsgesetz

(4) $a(a + b) = a$

(4') $a + ab = a$

Tautologie

(5) $a \cdot a = a$

(5') $a + a = a$

Gesetz über die Negation

(6) $\bar{a} \cdot a = 0$

(6') $\bar{a} + a = 1$

Doppelte Negation

(7) $\overline{\bar{a}} = a$

Gesetz von De Morgan

(8) $\overline{a \cdot b} = \bar{a} + \bar{b}$

(8') $\overline{a + b} = \bar{a} \cdot \bar{b}$

Operationen mit 0 und 1

(9.1) $a \cdot 1 = a$

(9.1') $a + 0 = a$

(9.2) $a \cdot 0 = 0$

(9.2') $a + 1 = 1$

(9.3) $\text{not } 0 = 1$

(9.3') $\text{not } 1 = 0$

Bemerkung: Die jeweils in einer Zeile stehenden Gesetze sind duale Gesetze voneinander; Beispiel: (3') ist das duale Gesetz von (3), (3) das duale Gesetz von (3').

Beweis von Rechengesetz (3):

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

Da die Spalten zu $a(b + c)$ und $ab + ac$ übereinstimmen, gilt: $a(b + c) = ab + ac$.

Aufgaben:

1. Beweise das Distributivgesetz (3').
2. Beweise die Gesetze von De Morgan.
Hinweis: Wahrheitstafel; außer den Spalten für a und b (4 Zeilen) erstelle Spalten für $a \cdot b$, $\overline{a \cdot b}$, $\bar{a}$, $\bar{b}$, $\bar{a} + \bar{b}$ für Regel (8).
3. Unter der Disjunktion **a or b** versteht man das nichtausschließende **oder** („non-exclusive or“), d. h., **a or b** ist genau dann **True**, falls **a** oder **b** oder sowohl **a** als auch **b True** sind („oder“ im Sinne von lat. vel).
Unter der Verknüpfung **a xor b** (andere Schreibweise: $a \oplus b$) versteht man das ausschließende **oder** (exclusive or), d. h., $a \oplus b$ ist genau dann **True**, falls entweder **a** oder **b** den Wert **True** hat.

Zeige: $a \oplus b = \bar{a} \cdot b + a \cdot \bar{b}$

BEISPIEL 1

Die Boolesche Funktion
 $z = f(a, b, c)$
 ist durch nebenstehende
 Wahrheitstafel
 gegeben:

a	b	c	z
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	0
1	1	1	1

- Ermittle die disjunktive Normalform (**DNF**; Disjunktion von Konjunktionen) für **z**.
- Vereinfache den Funktionsterm unter Anwendung der Booleschen Rechengesetze.
- Zeichne den Schaltplan für die optimierte Funktion **z**.

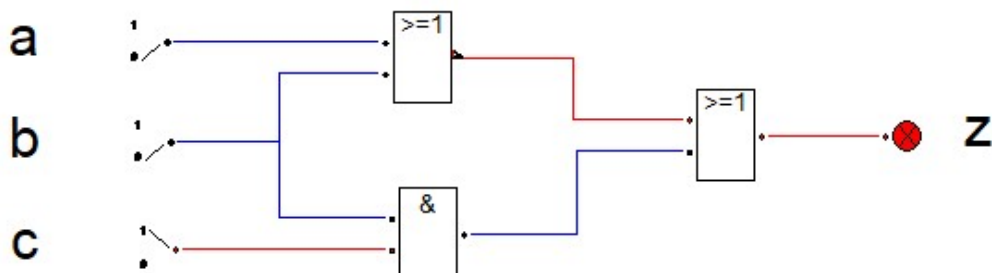
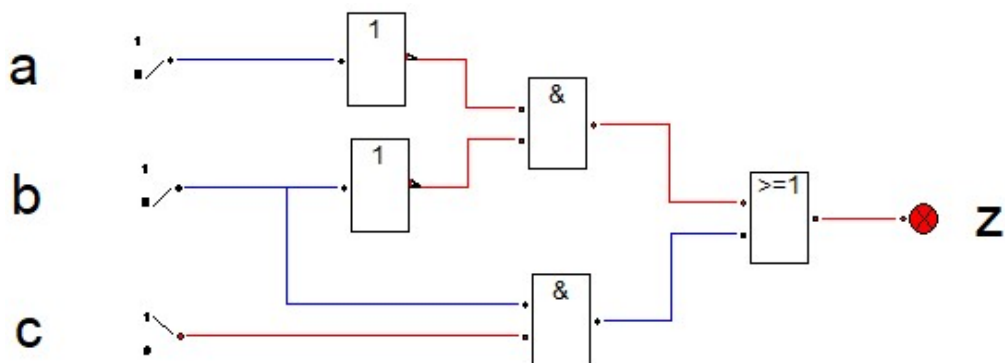
Lösung:

$$a) \quad z = \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot c + a \cdot b \cdot c$$

$$\begin{aligned}
 b) \quad z &= \bar{a} \cdot \bar{b} \cdot (\bar{c} + c) + b \cdot c \cdot (\bar{a} + a) \quad \text{Kommutativ- und Distributivgesetz} \\
 &= \bar{a} \cdot \bar{b} \cdot 1 + b \cdot c \cdot 1 \\
 &= \bar{a} \cdot \bar{b} + b \cdot c \\
 &= \overline{a+b} + b \cdot c \quad \text{de Morgan's Gesetz}
 \end{aligned}$$

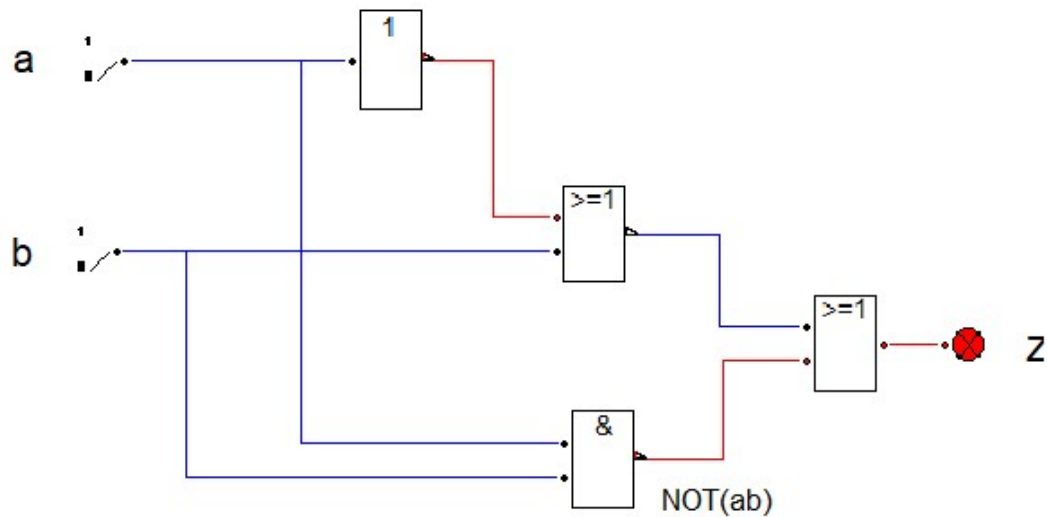
$$c) \quad z = \bar{a} \cdot \bar{b} + b \cdot c \quad (\text{oben})$$

$$z = \overline{a+b} + b \cdot c \quad (\text{unten})$$



BEISPIEL 2

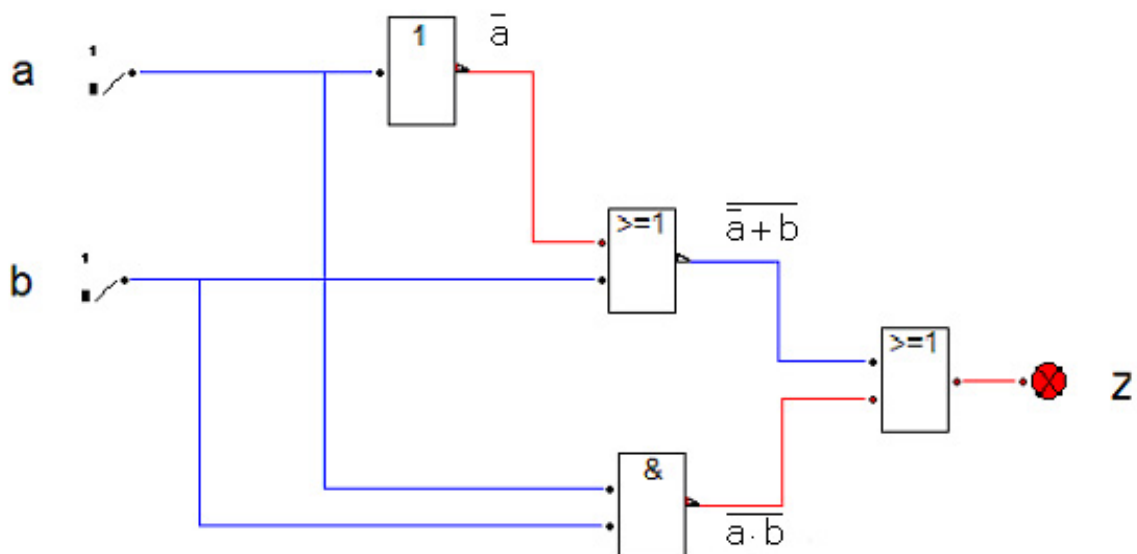
Gegeben ist folgende digitale Schaltung mit den Eingangsvariablen **a**, **b** und der Ausgangsvariablen **z**:



- Ermittle den Booleschen Term für die Boolesche Funktion $z = f(a,b,c)$. Hinweis: Notiere am Ausgang jedes Gatters jeweils den Booleschen Term (Beispiel: $\overline{a} \cdot b$ am Ausgang des NAND-Gatters).
- Vereinfache den in a) erhaltenen Term unter Verwendung der Rechenregeln für Boolesche Ausdrücke;
- Erstelle die Wahrheitstafel und zeichne das Schaltbild für den vereinfachten Funktionsterm; teste beide Schaltungsvarianten mit einem Digitalsimulationsprogramm.

Lösung:

zu a):

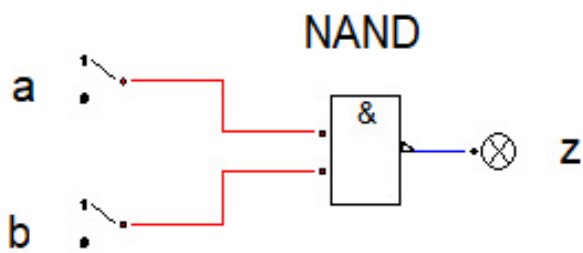


zu b):

$$\begin{aligned}
 z &= \overline{\overline{a+b} + \overline{a \cdot b}} \\
 &= \overline{\overline{a} \cdot \overline{b} + (\overline{a} + \overline{b})} && (2\text{-mal de Morgan}) \\
 &= \overline{a \cdot \overline{b} + \overline{a} + \overline{b}} && (\text{wegen } \overline{\overline{a}} = a) \\
 &= \overline{\overline{a} + \overline{b} \cdot a + \overline{b}} && (\text{Kommutativgesetze}) \\
 &= \overline{\overline{a} + \overline{b} \cdot a + \overline{b} \cdot 1} && (\text{wegen } a = a \cdot 1) \\
 &= \overline{\overline{a} + \overline{b} \cdot (a + 1)} && (\text{Distributivgesetz}) \\
 &= \overline{\overline{a} + \overline{b}} && (\text{wegen } a + 1 = 1) \\
 &= \overline{\overline{a \cdot b}} && (\text{de Morgan})
 \end{aligned}$$

zu c):

optimierte Schaltung:



Wertetabelle:

a	b	z
0	0	1
0	1	1
1	0	1
1	1	0