4 Boolesche Algebra

Die boolesche Logik oder auch die boolesche Algebra geht auf den Mathematiker Georg Boole (England, 1815 - 1864) zurück. Als Grundlage dient der booleschen Algebra die Darstellung von Werten in binärer Form. Genauer bedeutet das eigentlich nur, dass der Wert einer Aussage entweder „wahr“ oder „falsch“ sein kann.
Aufgrund der Binärdarstellung existieren in der booleschen Algebra lediglich zwei Grundwerte.
Diese Grundwerte können durch eine Reihe von logischen Operationen[1] miteinander verknüpft werden.
Das Resultat einer solchen Verknüpfung wird in Tabellen, den sogenannten Wahrheitstabellen, definiert. Die uns bekannten logischen Verknüpfungen sind NICHT, ODER und UND.

		NOT	OR	AND
x_1	x_2	$\bar{x_1}$	$x_1 + x_2$	$x_1 \cdot x_2$
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

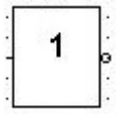
Wahrheitstabellen der booleschen Hauptverknüpfungen

Zusätzlich zu der jeweiligen Wahrheitstabelle finden Sie auch die verwendeten Schaltplan-Symbole. Diese entsprechen der IEC[2] 60617-12 Norm aus der IEC-60617-Schaltzeichen-Datenbank.

[1] Diese werden auch als Verknüpfungen bezeichnet. Siehe hierzu Abschnitt 2: Logische Verknüpfungen.

[2] Die International Electotechnical Commission (IEC) ist eine internationale Normierungsorganisation.

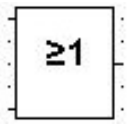
4.0.1 NOT



x_1	$f(x_1)$
0	1
1	0

Beschreibung: Dieses Schaltelement negiert/invertiert das Eingangssignal.

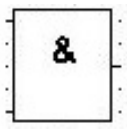
4.0.2 OR



x_1	x_2	$f(x_1; x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Beschreibung: Das Ausgangssignal dieses Schaltsymbol ist genau dann 1, wenn eines oder beide der Eingangssignale 1 sind.

4.0.3 AND



x_1	x_2	$f(x_1; x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

Beschreibung: Das Ausgangssignal dieses Schaltsymbol ist genau dann 1, wenn beide der Eingangssignale 1 sind.

5 Schaltalgebra

Die Schaltalgebra ist ein Spezialfall der booleschen Algebra. In ihr werden alle allgemeinen Aussagen wir „wahr“ und „falsch“ durch Kontaktzustände (also Kontakt geschlossen oder Kontakt offen) ersetzt.
So ist es dann möglich, Schaltnetze durch boolesche Gleichungen zu beschreiben. In der Schaltalgebra ist es zudem üblich, die UND-Funktion als Multiplikationszeichen ($\cdot$) und die ODER-Funktion als Pluszeichen ($+$)darzustellen.

Die NICHT-Funktion wird im allgemeinen als Querstrich ($\bar{x}$) über der Zustandsvariablen oder durch das mathematische Nicht-Zeichen ($\neg$) markiert.

Wir schauen uns die folgende „logische Gleichung“ an:

$$(E_1 + E_2) \cdot (E_3 + E_4) = A_1$$

Lesen kann man das Ganze dann als:
*(E_1 **ODER** E_2) **UND** (E_3 **ODER** E_4) = A_1*

Anwendung für die boolesche Logik

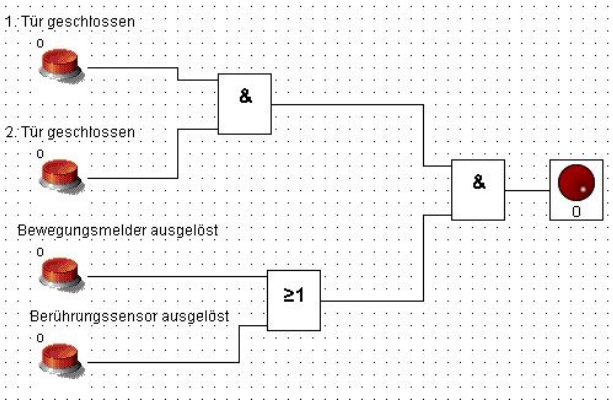
Generell kann die boolesche Logik bei den meisten Logikproblemen angewendet werden, bei denen die Eingangsvariablen jeweils nur zwei Zustände haben.
Betrachten wir das folgende Beispiel:
Ein Raum hat zwei Türen und ein Fenster. Der angeschlossene Warnmelder soll einen Warnton abgeben, wenn beide Türen mit einem Sicherheitsschloss abgeschlossen sind und entweder ein Bewegungsmelder im Raum oder ein Berührungssensor am Fenster anspringt.

Wie gehen wir vor, um die Schaltung aufzustellen?

Als Eingangssignale haben wir die beiden Türen T_1, T_2 (UND), sowie den Bewegungsmelder B_1 und den Berührungssensor B_2 (ODER).
Die Türschlösser und die externen Sensoren (UND) lösen zusammen den Warnton aus. Das Ausgangssignal ist der Warnton W_1. So ergibt sich dann die folgende Funktionsgleichung

$$W_1 = (E_1 \cdot E_2) \cdot (B_1 + B_2)$$

Die Funktionalität lässt sich dann mit Hilfe der Gleichung in einer Simulationssoftware[3] umsetzten.



6 Schaltungen aufbauen

Jetzt haben wir die *boolesche Algebra* und die *Schaltalgebra* kennengelernt. Stellt sich jetzt die Frage: *Wie kann ich das alles anwenden?*

Von der Anforderung zur Schaltung

In der zuvor dargestellten Beschreibung ging das Aufstellen der Gleichung direkt aus dem Text. Dies ist aber nicht immer der Fall. Daher wollen wir das Vorgehen näher betrachten:
Beispiel: Die Primzahl-Maschine kann die Zahlen $0 - 7$ verarbeiten. Dabei soll die LED besagter Maschine nur dann aufleuchten, wenn die ‘gefütterte‘ Zahl eine Primzahl ist.

`Wie realisieren wir die Schaltung, die die Primzahl-Maschine repräsentiert?`

(I) Zunächst müssen wir überlegen, **wie viele Eingangsvariablen benötigen wir?**
Im Fall der Primzahl-Maschine können wir die Zahlen $0 - 7$ darstellen, also benötigen wir 3 Bit $\hat{=}$ 3 Eingangsvariablen.

(II) **Wie viele Ausgangsvariablen benötigt**

[3] Zum Beispiel LogicSim.

unsere Schaltung?
Da wir lediglich eine LED haben, die leuchtet oder eben nicht, benötigen wir auch nur **eine** Ausgangsvariable.

(III) Als nächstes müssen wir herausfinden, **bei welcher Variablenbelegung (*Eingangsvariablen*) ist unser *Ausgangssignal* 1?**
Für unser Beispiel erstellen wir eine Wahrheitstabelle und tragen in den entsprechenden Stellen eine 1 ein.

x_2	x_1	x_0	LED
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

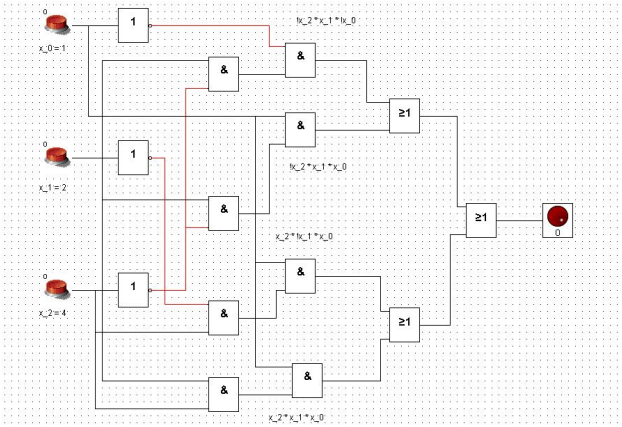
(IV) Aus der Tabelle, schreiben wir die **Variablenbelegungen jeweils als Konjunktion** (UND-Verknüpfung).
Für die obige Tabelle ergibt sich somit:

$$LED_2 = \overline{x_2} \cdot x_1 \cdot \overline{x_0}$$
$$LED_3 = \overline{x_2} \cdot x_1 \cdot x_0$$
$$LED_5 = x_2 \cdot \overline{x_1} \cdot x_0$$
$$LED_7 = x_2 \cdot x_1 \cdot x_0$$

(V) Aus dieser Auflistung **stellen wir die Gleichung auf**, die Grundlage für die Schaltung ist.

$$\begin{aligned} LED &= LED_2 + LED_3 + LED_5 + LED_7 \\ &= (\overline{x_2} \cdot x_1 \cdot \overline{x_0}) + (\overline{x_2} \cdot x_1 \cdot x_0) + \\ &\quad (x_2 \cdot \overline{x_1} \cdot x_0) + (x_2 \cdot x_1 \cdot x_0) \end{aligned}$$

(VI) Unter Verwendung der Gleichung **realisieren wir die Schaltung in der Simulationssoftware**[4].



Die Vorgehensweise im Überblick

(I) Anzahl der Eingangsvariablen bestimmen

(II) Anzahl der Ausgangsvariablen bestimmen

(III) Wertetabelle anlegen und füllen

(IV) Konjunktionen der 1-Zeilen aufstellen

(V) Gleichung aufstellen (ODER-Verknüpfung der Konjunktionen)

(VI) Schaltung realisieren

[4] LogicSim