Übersicht

Zahlensysteme

Kapitel 1

Grundlagen des Binär-, Dezimal- und Hexadezimalsystems

Binär bedeutet zweiwertig. Das binäre Zahlensystem verfügt also über zwei Werte, 0 und 1.
Computer rechnen im Binärsystem (Dualsystem) vor allem deshalb, weil es sich elektrisch sehr einfach darstellen lässt.

0 → (keine elektrische Spannung)
1 → (elektrische Spannung)

Auch digitale Signale sind größtenteils binär codiert (z.B.: ASCII-Code). Aber dazu lernt man mehr in den Fächern für Hochsprachen.
Zur kürzeren und einfachen Darstellung von Binärzahlen verwendet man das Hexadezimalsystem. Da die Basis im Binärsystem 2 ist und 2^4 der Zahl 16 entspricht, erklärt sich von selbst, wieso man gerade das hexadezimale Zahlensystem zur Darstellung verwendet. Immer vier Stellen einer Binärzahl können zu einer Hexadezimalziffer zusammengefasst werden.

Um die Zahl 2 (hex) darzustellen, schreiben wir → 0010

Um die Zahl F3 (hex) darzustellen, schreiben wir → 1111 0011

Da die Basis des Dezimalsystems 10 keine 2er Potenz ist, erfolgt der Stellenwertwechsel nicht synchron zu dem der Binärzahlen, was für uns soviel heißt, dass binäre Zahlen im Dezimalsystem nicht in vollem Umfang dargestellt werden können. Der Stellenwertwechsel im Hexadezimalen erfolgt synchron.

Bsp:

9 (dez) → 1001 (binär)
10 (dez) → 1010 (binär)

F (hex) → 1111 (binär)
10 (hex) → 1 0000 (binär)

Aus diesen Gründen ist es für AIKs wichtig, sich mit den Grundlagen dieser Zahlensysteme auseinanderzusetzen und Umrechnungen von einem ins andere Zahlensystem zu beherrschen.

Die folgende Tabelle zeigt, dass das Dezimalsystem den Darstellungsumfang binärer Zahlen nicht vollständig ausschöpft. Das Hexadezimalsystem im Gegensatz dazu, schon.

Zudem sind Hex-Zahlen viel kürzer und einfacher bei der Programmierung zu verwenden. Stellen Sie sich vor, Sie müssen ein Register in binärer Form verändern. Das würde etwa so aussehen (*Nur ein x-beliebiges Beispiel)

mov al, 1001 1110 (→ in binärer Form)

mov al, 9E (→ in hex viel kürzer)

U.a. ist es deswegen üblich, in Programmcodes zur Darstellung von Zahlen das Hexadezimalsystem zu verwenden.
Da aber aus den Hex-Zahlen die binären Einsen und Nullen nicht auf Anhieb erkennbar sind (zumindest nicht ohne Übung), sind Kenntnisse über Umrechnung von verschiedenen Zahlensystemen notwendig.

Vergleich der Zahlensysteme		
Dezimal	Binär	Hexadezimal
0	**0 0 0 0**	0
1	**0 0 0 1**	1
2	**0 0 1 0**	2
3	**0 0 1 1**	3
4	**0 1 0 0**	4
5	**0 1 0 1**	5
6	**0 1 1 0**	6
7	**0 1 1 1**	7
8	**1 0 0 0**	8
9	**1 0 0 1**	9
	1 0 1 0	**A**
	1 0 1 1	**B**
	1 1 0 0	**C**
	1 1 0 1	**D**
	1 1 1 0	**E**
	1 1 1 1	**F**
	1 0 0 0 0	**10**

Schauen wir uns einmal die **Dezimal**zahl **2597** etwas genauer an.
Diese Zahl lässt sich auch folgendermaßen darstellen:

$$
\begin{array}{ll}
 & 7 * 10^0 = 7 \\
+ & 9 * 10^1 = 90 \\
+ & 5 * 10^2 = 500 \\
+ & 2 * 10^3 = 2000 \\
\hline
= & \mathbf{2597}
\end{array}
$$

Zur Erinnerung:
$10^0 = 1$
$10^1 = 10$
$10^2 = 100$

Hieraus wird ersichtlich, dass, wenn wir die Ziffern von rechts nach links durchnummerieren, die erste Ziffer die Wertigkeit 10^0, die zweite Ziffer die Wertigkeit 10^1, die dritte Ziffer die Wertigkeit 10^2 und die vierte Ziffer die Wertigkeit 10^3 besitzt.
Der Zusammenhang zwischen Wertigkeit der Ziffer und Zahlensystem besteht aus der Anzahl der verschiedenen Ziffern die das Zahlensystem verwendet.
Beim Dezimalsystem sind es 10 verschiedene Ziffern (0….9). Deshalb die Wertigkeiten 10^0, 10^1, 10^2, 10^3, 104……usw.

Das **Binärsystem** verwendet 2 verschiedene Ziffern (0 und 1).
Die Wertigkeiten der Ziffern sind also 2^0, 2^1, 2^2, 2^3, 2^4, 2^5…….usw.

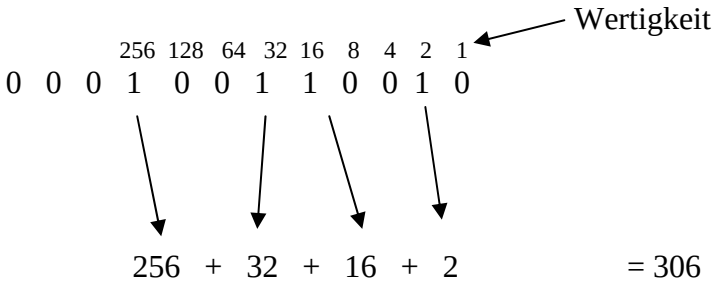
Schauen wir uns also eine Binärzahl genauer an. zB: **0 1 0 1**
Diese Zahl lässt sich auch folgendermaßen darstellen:

$$
\begin{array}{ll}
 & 1 * 2^0 = 1 \\
+ & 0 * 2^1 = 0 \\
+ & 1 * 2^2 = 4 \\
+ & 0 * 2^3 = 0 \\
\hline
= & 5 \quad \text{(dezimal)}
\end{array}
$$

Zur Erinnerung:
$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$

Tipp:
Wenn ich eine Binärzahl in eine Dezimalzahl umrechnen will, dann schreibe ich zuerst die Wertigkeiten über die Ziffern und zähle dann einfach die Wertigkeiten die mit Eins belegt sind zusammen.

Bsp:



256 + 32 + 16 + 2 = 306 (dezimal)

(am Schluss dieses Kapitels gehe ich noch mal genau auf die benötigten Umrechnungsarten ein)

Das **Hexadezimal**system verwendet 16 verschiedene Ziffern (0……F)
Die Wertigkeiten der Ziffern sind also 16^0, 16^1, 16^2, 16^3……..usw.
Zum Beispiel lässt sich die Zahl 312 (hex) folgendermaßen darstellen:

$16^0 = 1$
$16^1 = 16$
$16^2 = 256$
$16^3 = 4096$

$$
\begin{array}{ll}
 & 2 * 1 = 2 \\
+ & 1 * 16 = 16 \\
+ & 3 * 256 = 768 \\
\hline
= & 786 \quad \text{(dezimal)}
\end{array}
$$

Umrechnungen

Binär → Dezimal
Binär → Hex

Dezimal → Binär
Dezimal → Hex

Hex → Binär
Hex → Dezimal

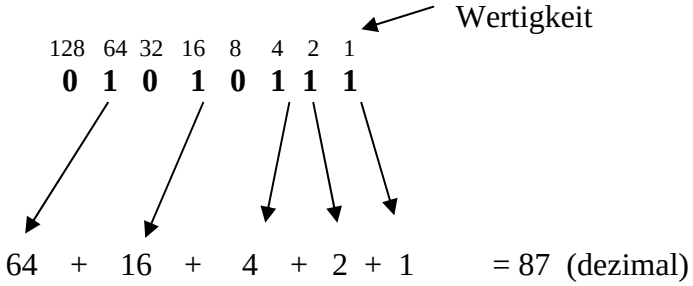
Wertigkeit der Stellen einer Zahl in den drei Zahlensystemen						
Stelle	6	5	4	3	2	1
Dezimal	100.000	10.000	1.000	100	10	1
Binär	32	16	8	4	2	1
Hex	…..	65.536	4.096	256	16	1

Bild 1.2

Binär → Dezimal

Diese Umrechnung erfolgt durch addieren der Wertigkeiten die mit Eins belegt sind.

Bsp: **0101 0111**



Binär → Hex

Diese Umrechnung erfolgt durch Bilden von Vierer-Blöcken der Binärziffern und Umwandlung dieser Blöcke in jeweils eine Hexzahl-Ziffer anhand einer Binär → Hex Tabelle

Bsp: 0101 0111

5 7 (hex)

Durch Bilden der Vierer-Blöcke kann man ganz einfach an der Tabelle ablesen, welchen Hex-Wert dieser Block hat. Mit dem Bilden der Blöcke fängt man von rechts an. Fehlende Stellen werden mit Nullen aufgefüllt.

Bsp:

Aus 1100101011 wird 0011 0010 1011

entspricht 3 2 B (hex)

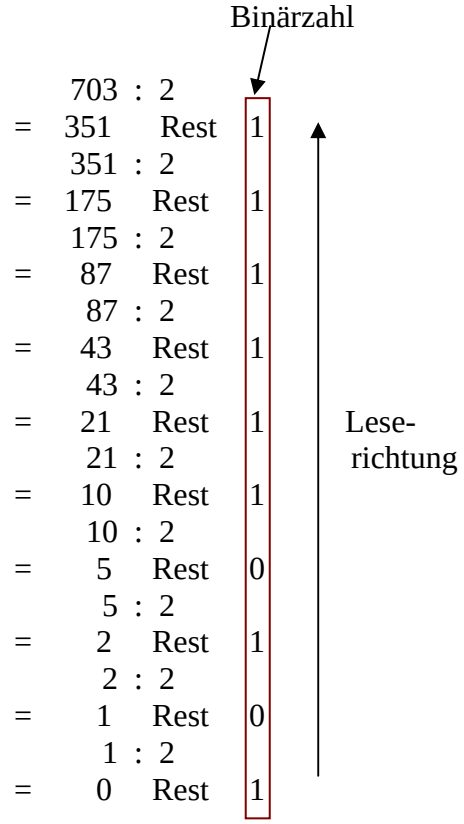
Binär	Hexadezimal
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	**A**
1 0 1 1	**B**
1 1 0 0	**C**
1 1 0 1	**D**
1 1 1 0	**E**
1 1 1 1	**F**

Bild 1.3

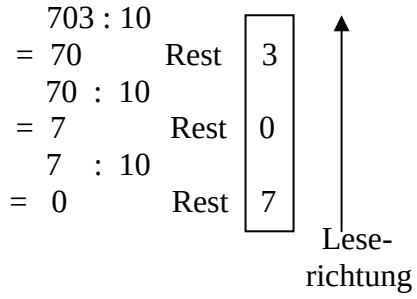
Dezimal → Binär

Ebenfalls sehr einfach ist die Umrechnung von Dezimal in Binär. Man teilt die umzuwandelnde Zahl durch 2 und verfährt mit dem Ergebnis genauso. Solange, bis man bei Null angelangt ist. Bei jedem Rechenschritt notiert man sich den verbleibenden Rest der Division. Der Rest entspricht nämlich genau der binären Zahl, die den Wert der Dezimalzahl hat.

Bsp: 703 (dezimal)



Die Leserichtung erklärt sich dadurch, dass wir bei jeder Division den ganzzahligen Rest der umzuwandelnden Zahl abtrennen. D.h., mit der Division setzen wir ein Komma und trennen die Stelle nach dem Komma ab. Mit dieser Methode können Sie eine Dezimalzahl in jedem beliebigen Zahlensystem darstellen. Am verständlichsten wird es, wenn man eine Dezimalzahl einfach auch in Dezimal darstellt.



703 (dezimal) entspricht also 10 1011 1111 (binär)

Dezimal → Hex

Die Umrechnung von Dezimal nach Hex erfolgt am einfachsten über das Binärsystem. D.h., zuerst wird die Dezimalzahl in eine Binärzahl gewandelt (wie oben besprochen), und dann werden die binären Vierer-Blöcke gebildet, aus denen man dann die Hex-Zahl aus der Binär → Hex –Tabelle ablesen kann.
(Gute Kopfrechner können auch gleich die Division mit dem Nenner 16 durchführen)

Bsp:

703 (dez) entspricht ja 10 1011 1111 (binär) also 2BF (hex)

Hex → Binär

Jeweils eine Hex – Ziffer bildet einen binären Vierer-Block. Diese Umrechnung ist dieselbe wie Binär nach Hex, nur in umgekehrter Reihenfolge. D.h. wir müssen wieder nur in der Tabelle nachschauen. (*mit ein wenig Übung wird die Tabelle bald überflüssig*).

Bsp:

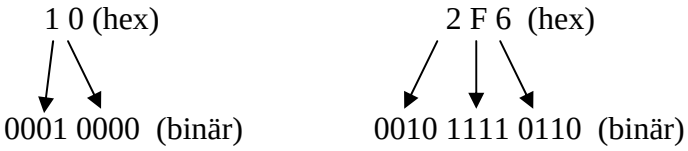


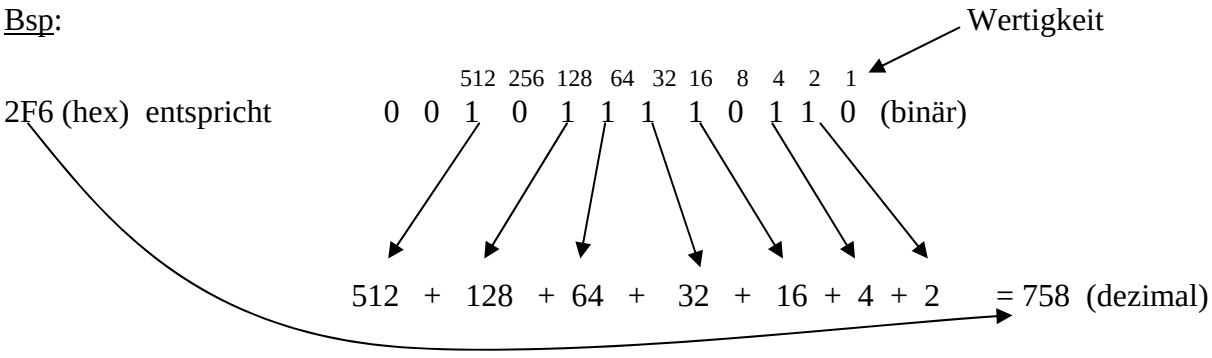
Bild 1.3

Binär	Hexadezimal
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	**A**
1 0 1 1	**B**
1 1 0 0	**C**
1 1 0 1	**D**
1 1 1 0	**E**
1 1 1 1	**F**

Hex → Dezimal

Diese Umrechnung erfolgt wieder am einfachsten über den Zwischenschritt Hex nach Binär. Dann kann ich diese Binärzahl in eine Dezimalzahl umrechnen, um zum Ergebnis zu kommen.

Bsp:



Addition binärer Zahlen

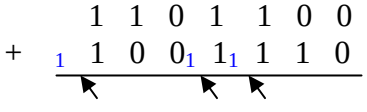
0 + 0 = 0

1 + 0 = 1

1 + 1 = 10 (= 0, Übertrag 1)

1 + 1 + 1 = 11 (= 1, Übertag 1)

Bsp:



Testen Sie Ihr Wissen

1. Umrechnen von Binär nach Dezimal

a) 0011 1100 ☐

b) 1110 0101 ☐

c) 1111 0001 ☐

2. Umrechnen von Binär nach Hex

a) 1010 1001 ☐

b) 0110 0011 ☐

c) 1000 1011 ☐

3. Umrechnen von Dezimal nach Binär

a) 254 ☐

b) 240 ☐

c) 170 ☐

4. Umrechnen von Dezimal nach Hex

a) 250 ☐

b) 175 ☐

c) 201 ☐

5. Umrechnen von Hex nach Binär

a) 81 ☐

b) 96 ☐

c) CF ☐

6. Umrechnen von Hex nach Dezimal

a) AB ☐

a) 7C ☐

c) F1 ☐

7. Addieren von zwei Binärzahlen

a) 1110 0101 + 1100 1100 ☐

b) 1100 1111 + 1111 1001 ☐

Grundgatter

Kapitel 2

Boolesche Algebra in der Praxis

Die Boolesche Algebra basiert auf logischen Funktionen. Diese Funktionen können nur zwei Werte annehmen und sind somit elektronisch gut darstellbar.
Um zu verstehen, wie wichtig diese Logikfunktionen im Bereich der Informatik sind, muss man sich vor Augen halten, dass sämtliche Operationen, die man durchführen will, darauf beruhen, dass man die Möglichkeit besitzt einen bestimmten Zustand herbeizuführen, der dann ein Ereignis auslösen soll.
z.B., wenn ich diesen Button anklicke, dann soll…. (Softwarelösung)
wenn ich diesen Schalter betätige, dann soll…. (Hardwarelösung)
Dies kann man mit den folgenden Funktionen realisieren.

Wir wenden uns zunächst der Hardwarelösung zu.
Anwenderprogramme realisieren diese Möglichkeit softwaremäßig, aber die zugrunde liegende Logik ist die gleiche. (Details dazu im 2. und 3. Semester)

Folgende Grundfunktionen sind definiert:

- AND
- OR
- NOT
- NAND
- NOR
- XOR
- XNOR

Was ist ein Gatter?

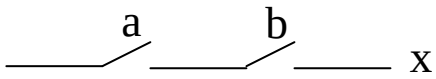
Ein Gatter ist ein Baustein, der mittels internen Schaltern (Transistoren) die Grundfunktionen elektrisch umsetzt. Im Computer sind die Arithmetik und Logik Unit (ALU), Schieberegister, Addierer, Flip-Flops, Speicher….etc. aus solchen Gattern aufgebaut. Zudem ermöglichen sie Rechenoperationen wie Addition, Subtraktion, Multiplikation…

Auf den folgenden Seiten sind die Grundgatter aufgeführt und erklärt. Zum einfacheren Verständnis ist jeder Baustein mit höchstens zwei Eingängen (a , b) und einem Ausgang (x) belegt. Die einzelnen Eingänge können entweder **log. 0** oder **log. 1** als Zustand führen. Der Ausgang ist abhängig von diesem Eingangszustand. Zur Darstellung dieser beiden logischen Zustände verwendet man i. Allg. die Spannungen **0 Volt** und **5 Volt** .
Um Fehler, die von abweichenden Spannungspegeln herrühren zu verringern, sind für beide Zustände abgegrenzte Spannungsbereiche festgelegt.

Von **0 Volt bis 0,8 Volt** → Spannungsbereich für **log. 0**
Von **2 Volt bis 5 Volt** → Spannungsbereich für **log. 1**

Zur Erklärung der Prinzipschaltung in den folgenden Tabellen, stellt man sich am Besten ein leitendes Medium vor (Kabel, etc.). Sobald das Kabel leitend ist, liegt am Ausgang x eine log. 1 an. Mit dem Anlegen von Spannung an den Eingängen a und b, verändere ich die interne Schalterstellung.

Bsp:

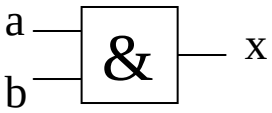
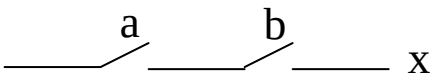
AND 

Der Anfangszustand (Eingang a=0 Volt, Eingang b=0Volt, Ausgang x = 0 Volt)
Erst wenn ich a und b verändert habe, ist das Kabel wirklich leitend.

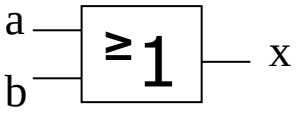
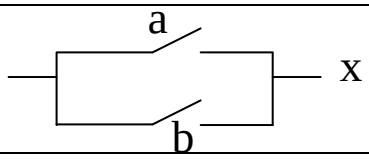
In den Funktionsgleichungen der folgenden Tabellen wird verwendet: AND → •
OR → +
NOT → —

(für NOT nicht das Minuszeichen, sondern ein Strich über der Variablen. z.B. $\overline{a}$)

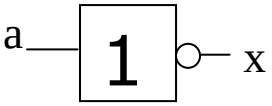
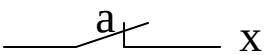
AND

Schaltzeichen	
Prinzipschaltung	
Funktionsgleichung	x = a • b
Beschreibung der Funktion	Der Ausgang x ist nur dann 1, wenn alle Eingänge 1 sind. Der Ausgang x ist 0, wenn mindestens ein Ausgang 0 ist.
Wahrheitstabelle	**a** | **b** | **x** 0 | 0 | **0** 0 | 1 | **0** 1 | 0 | **0** 1 | 1 | **1**

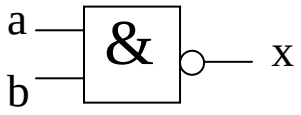
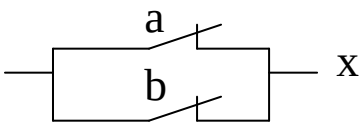
OR

Schaltzeichen	
Prinzipschaltung	
Funktionsgleichung	$x = a + b$
Beschreibung der Funktion	Der Ausgang x ist dann 1, wenn mindestens ein Eingang 1 ist. Der Ausgang x ist 0, wenn alle Eingänge 0 sind.
Wahrheitstabelle	**a** | **b** | **x** 0 | 0 | **0** 0 | 1 | **1** 1 | 0 | **1** 1 | 1 | **1**

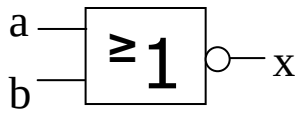
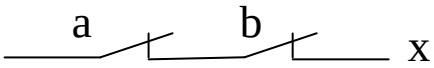
NOT

Schaltzeichen	
Prinzipschaltung	
Funktionsgleichung	$x = \overline{a}$
Beschreibung der Funktion	Der Ausgang x ist 1, wenn der Eingang 0 ist. Der Ausgang x ist 0, wenn der Eingang 1 ist.
Wahrheitstabelle	**a** | **x** 0 | **1** 1 | **0**

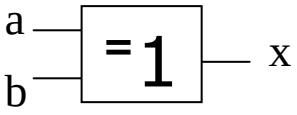
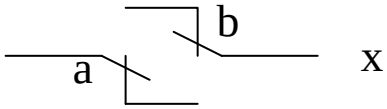
NAND (mit NAND lässt sich jede andere Funktion realisieren)

Schaltzeichen	
Prinzipschaltung	
Funktionsgleichung	$x = \overline{a \bullet b}$
Beschreibung der Funktion	Der Ausgang x ist 0, wenn alle Eingänge 1 sind. Der Ausgang x ist 1, wenn mindestens ein Eingang 0 ist.
Wahrheitstabelle	**a** | **b** | **x** 0 | 0 | **1** 0 | 1 | **1** 1 | 0 | **1** 1 | 1 | **0**

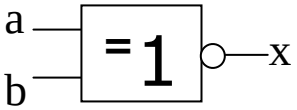
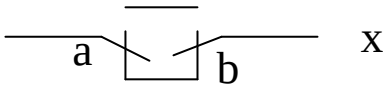
NOR (mit NOR lässt sich jede andere Funktion realisieren)

Schaltzeichen	
Prinzipschaltung	
Funktionsgleichung	$x = \overline{a + b}$
Beschreibung der Funktion	Der Ausgang x ist dann 0, wenn mindestens ein Eingang 1 ist. Der Ausgang x ist 1, wenn alle Eingänge 0 sind.
Wahrheitstabelle	**a** | **b** | **x** 0 | 0 | **1** 0 | 1 | **0** 1 | 0 | **0** 1 | 1 | **0**

XOR

Schaltzeichen	
Prinzipschaltung	
Funktionsgleichung	$x = (a \bullet \overline{b}) + (\overline{a} \bullet b)$
Beschreibung der Funktion	Der Ausgang x ist dann 1, wenn die Anzahl der Einsen an den Eingängen ungerade ist.
Wahrheitstabelle	**a** | **b** | **x** 0 | 0 | **0** 0 | 1 | **1** 1 | 0 | **1** 1 | 1 | **0**

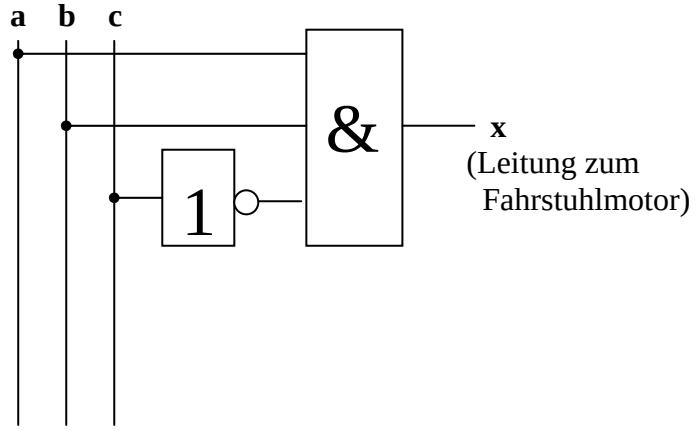
XNOR

Schaltzeichen	
Prinzipschaltung	
Funktionsgleichung	$x = (a \bullet b) + (\overline{a} \bullet \overline{b})$
Beschreibung der Funktion	Der Ausgang x ist dann 1, wenn die Anzahl der Nullen an den Eingängen gerade oder Null ist.
Wahrheitstabelle	**a** | **b** | **x** 0 | 0 | **1** 0 | 1 | **0** 1 | 0 | **0** 1 | 1 | **1**

Mit diesen Gattern lässt sich jede Schaltung realisieren, nicht nur im PC, sondern auch wenn es darum geht, Maschinen oder Sonstiges elektrisch zu steuern.

Beispiel:
Ein Fahrstuhl soll fahren, wenn drei Zustände erfüllt sind. Diese Zustände sind, dass der Knopf gedrückt ist, die Tür geschlossen ist und der Fahrstuhl nicht überladen ist.
Leitung a ist log. 1, wenn der Knopf gedrückt ist.
Leitung b ist log. 1, wenn die Tür geschlossen ist.
Leitung c ist log. 0, wenn der Fahrstuhl nicht überlastet ist.
Der Fahrstuhl fährt los, wenn der Ausgang x log. 1 ist. Der Ausgang x ist log. 1, wenn die Leitungen a und b log. 1 sind und wenn die Leitung c log. 0 ist.

(Strom/Spannung)



a	b	c	x
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**

Erstellen von Wahrheitstabellen

Zur Erstellung von Wahrheitstabellen muss man zuerst wissen, wie die Schaltung aufgebaut werden soll. Im obigen Fall soll die Funktion des Fahrstuhls über drei verschiedene Medien abgesichert werden. Über die Türe, über eine digitale Waage an der Aufhängung und über den Knopf. Das bestimmt dann auch die Anzahl der Leitungen oder Eingänge der Schaltung. Die Anzahl dieser Eingänge bestimmen die insgesamt möglichen Zustände.
Ein Eingang → zwei mögliche Zustände insgesamt
Zwei Eingänge → vier mögliche Zustände insgesamt
Drei Eingänge → acht mögliche Zustände insgesamt….usw.
Mit jedem zusätzlichen Eingang verdoppeln sich die insgesamt möglichen Zustände.

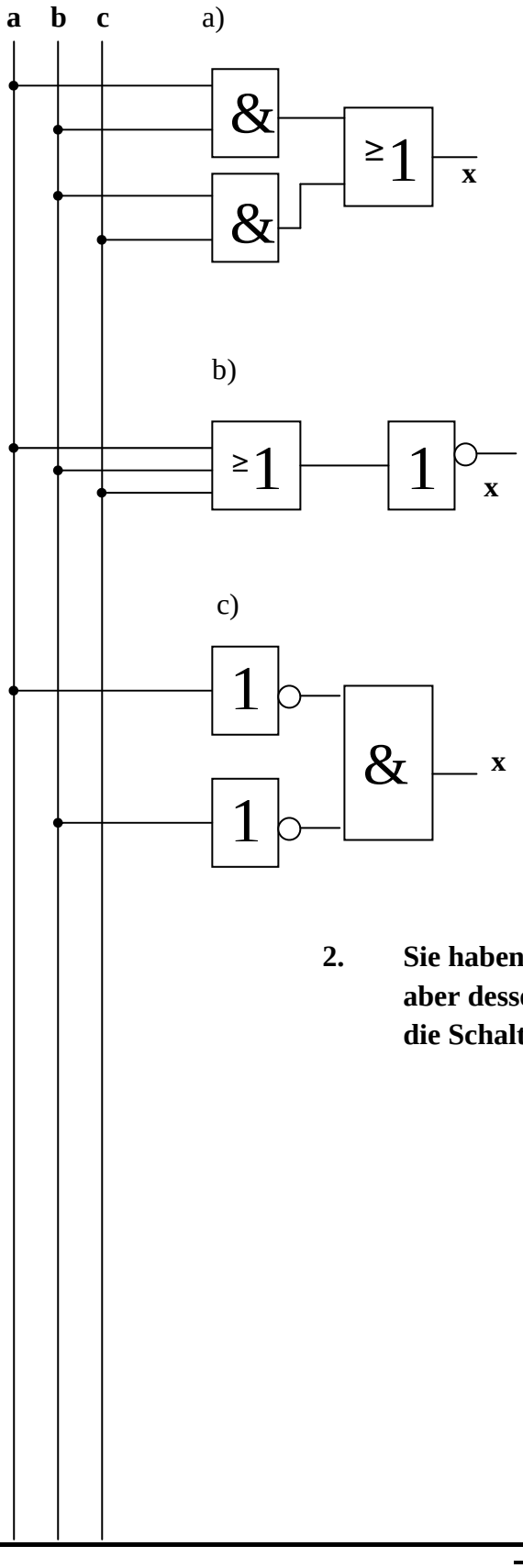
Stromversorgung

Damit in der Realität Gatter funktionieren, müssen die internen Transistoren mit 5 Volt Spannung versorgt werden.
Deshalb

Merke:
Offene Eingänge von Gattern verhalten sich, als lägen an ihnen 5 Volt an.

Testen Sie Ihr Wissen

1. Ergänzen Sie die Wahrheitstabellen für folgende Schaltungen.



a	b	c	x
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

a	b	c	x
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

a	b	x
0	0	
0	1	
1	0	
1	1	

2. Sie haben kein NOR-Gatter zur Hand, wollen aber dessen Funktion realisieren. Zeichnen Sie die Schaltung.

NOR - Tabelle

a	b	c	x
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

Schaltalgebra

Kapitel 3

Eine Einführung in die Digitaltechnik

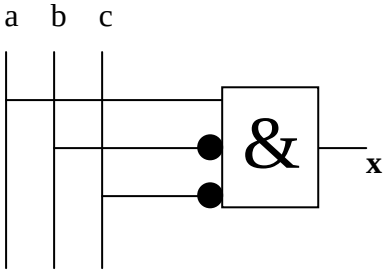
Schaltalgebra ist eigentlich eine Sonderform der Booleschen Algebra.
Die Basis der Booleschen Algebra ist das Prinzip der Zweiwertigkeit.
D.h., jede Aussage hat entweder den Wert „wahr“ oder den Wert „falsch“.
Dieses Prinzip kann man auf Schaltungen übertragen.

Die Aussage ist der Eingangszustand.
z.B. an Eingang **a** liegt 5 Volt an, an Eingang **b** liegt 0 Volt an, an Eingang **c** liegt 0 Volt an.
(* 0 Volt → log. 0, 5 Volt → log. 1)

Der Wert „wahr“ oder „falsch“ ist dabei der Ausgang.
z.B. an Ausgang x liegt 5 Volt an (=wahr).

Mathematisch dargestellt sieht das so aus: $x = a \bullet \overline{b} \bullet \overline{c}$

Die Schaltung dazu:



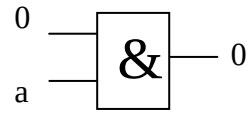
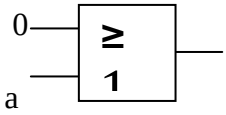
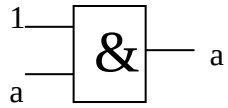
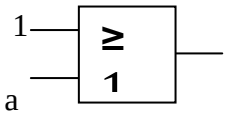
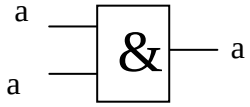
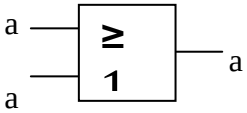
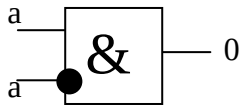
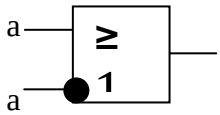
Da der NOT - Baustein relativ häufig gebraucht wird, zeichnet man diesen nicht jedes Mal mit seinem eigentlichen Schaltsymbol ein, sondern symbolisiert ihn mit einem schwarzen Kreis vor dem nächstfolgenden Gatter.

Wahrheitstabelle

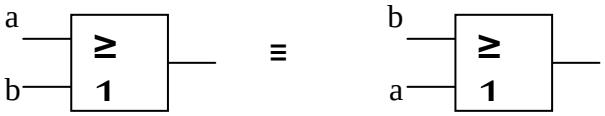
a	**b**	**c**	**x**
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

Da in der Praxis zum größten Teil sehr umfangreiche Schaltungen zu entwerfen sind, ist für uns eine Vertiefung in die Boolesche Algebra und dessen Regeln notwendig.

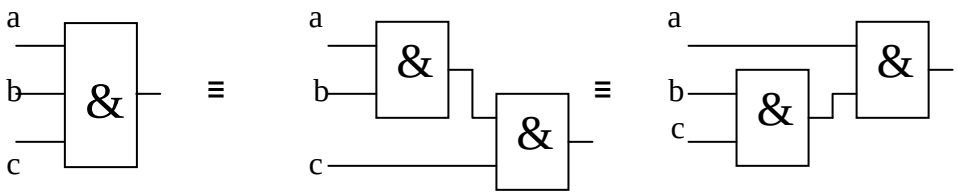
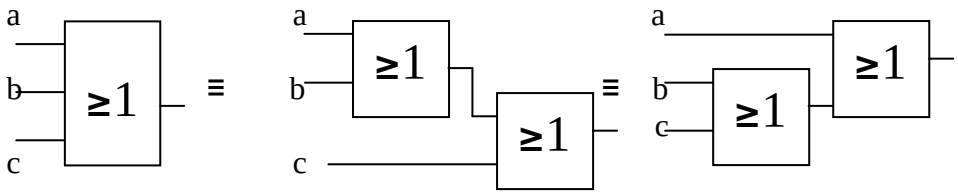
Regeln der Schaltalgebra

Theoreme (Aussagen) für Variablen		
Die Variable **a** steht für „entweder log. 0, oder log. 1“ **0** steht für log. 0 **1** steht für log. 1		
Regel	**Schaltung**	**Erklärung**
0 • a = 0		Wenn an einem Eingang einer AND – Verknüpfung eine log. 0 anliegt, dann sind die Zustände aller anderen Eingänge egal. Der Wert am Ausgang ist dann nämlich immer log. 0
0 + a = a		Wenn an einem Eingang einer OR – Verknüpfung eine log. 0 anliegt, nimmt der Ausgang den Wert der anderen Eingangs an.
1 • a = a		Wenn an einem Eingang einer AND – Verknüpfung eine log. 1 anliegt, nimmt der Ausgang den Wert der anderen Eingangs an.
1 + a = 1		Wenn an einem Eingang einer OR – Verknüpfung eine log. 1 anliegt, dann ist der Ausgang immer log.1
a • a = a		Wenn die Eingänge einer AND – Verknüpfung gleich sind, dann hat der Ausgang den selben Wert.
a + a = a		Wenn die Eingänge einer OR – Verknüpfung gleich sind, dann hat der Ausgang den selben Wert.
a • $\overline{a}$ = 0		Wenn bekannt ist, dass die Zustände der Eingänge einer AND-Verknüpfung ungleich sind, (es gibt ja nur 2 Zustände) dann muss mind. ein Eingang log. 0 sein. Und dann ist der Ausgang auch log. 0
a + $\overline{a}$ = 1		Wenn bekannt ist, dass die Zustände der Eingänge einer OR-Verknüpfung ungleich sind, dann muss mind. ein Eingang log. 1 sein. Und dann ist der Ausgang auch log. 1

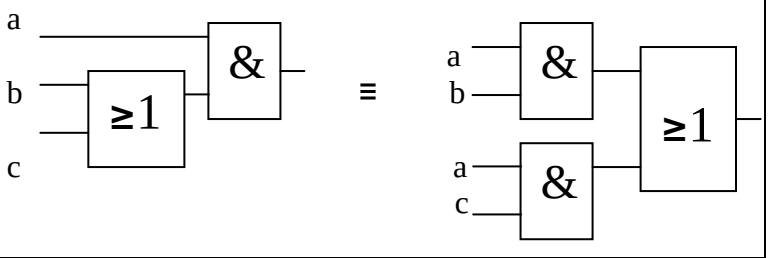
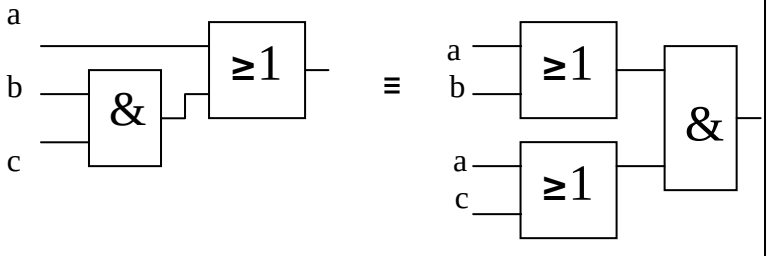
Kommutativgesetz (Vertauschungsgesetz)

Regel	Schaltung
a • b = b • a	
a + b = b + a	

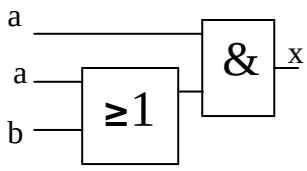
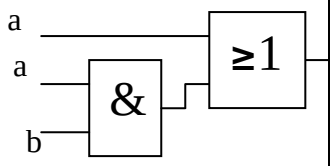
Assoziativgesetz (Zusammenfassen)

Regel	Schaltung
a • b • c = = (a • b) • c = = a • (b • c)	
a + b + c = = (a + b) + c = = a + (b + c)	

Distributivgesetz (Ausklammern)

Regel	Schaltung
a • (b + c) = (a • b) + (a • c)	
a + (b • c) = (a + b) • (a + c)	

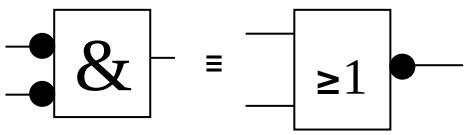
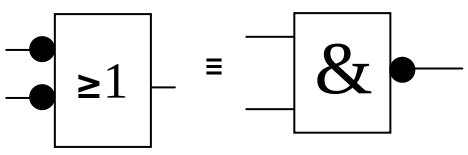
Sonderregeln

Regel	Schaltung	Erklärung
$a \bullet (a + b) = a$		Egal was an Eingang b anliegt, der Eingang a ist ausschlaggebend. Liegt an Eingang a eine log. 1 an, so ist auch der Ausgang x log. 1, liegt an Eingang a log.0 an, so ist auch der Ausgang x log. 0
$a + (a \bullet b) = a$		Auch hier ist der Eingang a entscheidend. Liegt an Eingang a eine log. 1 an, so ist der Ausgang x log. 1, liegt an Eingang a log.0 an, so ist der Ausgang x log. 0 . Eingang b kann also vernachlässigt werden.

Komplementbildung (doppelte Negation hebt sich auf)

Beispiele	Erklärung
$\overline{\overline{x}} = x$	Wenn man log. 1 negiert erhält man log. 0, wenn man noch mal negiert, hat man wieder log. 1
$\overline{\overline{x}} = \overline{x}$	
$\overline{\overline{a + b}} = a + b$	
$a + (\overline{\overline{b + \overline{c}}}) = a + (b + \overline{c})$	

De Morgansche Gesetze

Regel	Schaltung	Erklärung
aus $\bullet$ wird $+$ $\overline{a \bullet b} = \overline{a} + \overline{b}$		Am AND bekomme ich nur log. 1, wenn a und b log. 0 sind = Wahrheitstabelle von NOR
aus $+$ wird $\bullet$ $\overline{a + b} = \overline{a} \bullet \overline{b}$		Am OR bekomme ich nur log. 0, wenn a und b log. 1 sind = Wahrheitstabelle von NAND

Übersicht der Regeln

Theoreme

"•" = AND
"+" = OR
" "= NOT

AND-Verknüpfung

a • 0 = 0
a • 1 = a
a • a = a
a • $\overline{a}$ = 0

OR-Verknüpfung

a + 0 = a
a + 1 = 1
a + a = a
a + $\overline{a}$ = 1

NOT-Verknüpfung

a = $\overline{\overline{a}}$

Kommutativgesetz (Vertauschungsgesetz)

Das Kommutativgesetz besteht entweder nur aus OR oder aus AND Gliedern dessen Variablen beliebig vertauscht werden können.
x = a • b • c = c • b • a
x = a + b + c = c + b + a

Assoziativgesetz (Verbindungsgesetz bzw. Zuordnungsgesetz)

Das Assoziativgesetz ist dem Kommutativgesetz sehr ähnlich.
x = a • (b • c) = (a • b) • c
x = a + (b + c) = (a + b) + c

Distributivgesetz (Verteilungsgesetz)

Das Distributivgesetz besteht aus AND und OR Gliedern, es dient der Umformung der Gleichung (ausklammern und ausmultiplizieren), wodurch diese vereinfacht wird.
x = a • (b + c) = (a • b) + (a • c) **a.) a • (a + b) = a**
x = a + (b • c) = (a + b) • (a + c) **b.) a + (a • b) = a**

De Morgansche Gesetze

Die De Morgansche Gesetze sind sehr praktisch bei der Auflösung von negierten Ausdrücken, besonders für NAND- und NOR- Verknüpfungen.

$\overline{x = a • b} = \overline{a} + \overline{b}$ entspricht der Gleichung für NAND

$\overline{x = a + b} = \overline{a} • \overline{b}$ entspricht der Gleichung für NOR

(doppelte Negation ist nicht noch mal aufgeführt)

Vorrangregeln

Auch in der Schaltalgebra kann eine bestimmte Reihenfolge der durchzuführenden Operationen vereinbart werden. Diese lauten :

1. Negation (NOT - Verknüpfung)
2. Konjunktion (AND - Verknüpfung)
3. Disjunktion (OR - Verknüpfung)

Klammersetzung

NOR - Verknüpfung: **keine Klammern** erforderlich
AND - Verknüpfung: **keine Klammern** erforderlich
OR - Verknüpfung: **Klammern** unbedingt **erforderlich** (damit die Bedeutung der Gleichung erhalten bleibt).
Klammern sind vor allem dann zu setzen, wenn die De Morganschen Gesetze auf NAND- oder NOR- Verknüpfungen angewendet werden.

Allgemeines

Die Schaltalgebra befasst sich mit der formellen Beschreibung von digitalen Schaltnetzen. (Digitale Schaltnetze sind Schaltungen, die aus Gattern aufgebaut sind.)
Sie dient der Berechnung und Vereinfachung der jeweiligen Schaltungen.
z.B. zur Berechnung der Funktion einer Schaltung. (Wann ist ein Ausgang log.1, wann log.0)

In der Schaltalgebra existieren nur zwei Konstanten, **0** und **1**, sowie eine Reihe von Variablen(Eingänge/Ausgänge) denen man die Werte **log. 1** und **log. 0** zuordnen kann.

- Ist ein Schalter offen, so nimmt die Variable den Wert 0 an.
- Ist ein Schalter geschlossen, so nimmt die Variable den Wert 1 an.

Die Vereinfachungsregeln gleichen zum Teil denen der bekannten Algebra.

- Die Operationen "•" und "+" spielen in der Schaltalgebra eine ähnliche Rolle wie in der Zahlenalgebra. (**Punkt**rechnung, hier: AND, geht **vor Strich**rechnung, hier: OR).

Es lässt sich jede beliebige Schaltung aus NOT, AND und OR zusammensetzen.
Aus NAND oder NOR lässt sich ebenfalls jede beliebige Schaltung aufbauen.
Erklärung: wieso umformen nach NAND?

- Ein NAND-Gatter mit zwei Eingängen ist intern aus nur 4 Transistoren aufgebaut. Deshalb ist es billig im Einkauf, schnell in der Funktionsweise und bei vielen Firmen beliebt.

In der praktischen Schreibweise lässt man in der Ausdrücken für AND – Verknüpfungen das "•" weg!
Aus $a \bullet b$ wird ab

Zunächst schauen wir uns den Einsatz dieser Regeln an theoretischen Beispielen an. Danach folgt dann ein einfaches Beispiel, an dem man anhand der Schaltung und der Wahrheitstabelle den Spannungsverlauf nachvollziehen kann.

Beispiel 1:

Vereinfachen der folgenden Schaltung.	nach folgendem Gesetz
$x = a\,b\,\bar{c} + a\,\bar{b}\,c + a\,b\,c + a\,\bar{b}\,\bar{c}$	// $a\,b\,\bar{c} + a\,b\,c = a\,b\,(\bar{c} + c)$ // Distributivgesetz
$x = a\,b\,(\bar{c} + c) + a\,c\,(\bar{b} + b) + a\,\bar{b}\,(c + \bar{c})$	// $a\,b\,(\bar{c} + c) = a\,b\,(1) = a\,b$ //Theorem
$x = a\,b + a\,c + a\,\bar{b}$	//Distributivgesetz
$x = a\,c + a\,(b + \bar{b})$	//Theorem
$x = a + a\,c$	// Sonderregel $a + a\,c = a$
$x = a$	

Beispiel 2:

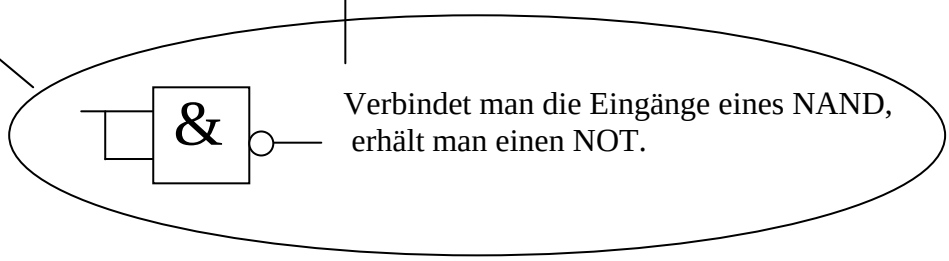
Schaltung mit einem OR, zwei AND und zwei NOT – Gattern ausschließlich mit NAND aufbauen!

nach folgendem Gesetz

$x = a\,\bar{b} + \bar{a}\,b$ → //Komplementbildung
//doppelte Negation

$x = \overline{\overline{a\,\bar{b}} + \overline{\bar{a}\,b}}$ → //De Morgansches Gesetz

$x = \overline{\overline{a\,\bar{b}} \bullet \overline{\bar{a}\,b}}$ → // fünf NAND – Gatter mit je 2 Eingängen



Beispiel mit Schaltung und Wahrheitstabelle:

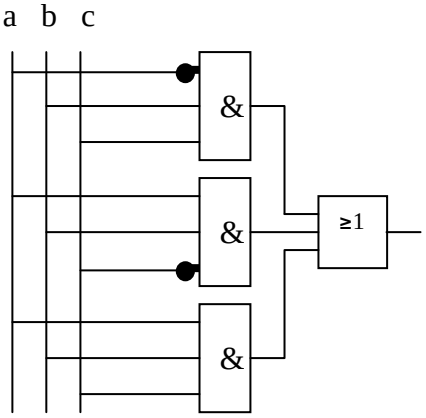
Folgende Schaltung zuerst vereinfachen und dann so umformen, dass sie ausschließlich mit NAND – Gattern aufgebaut ist.

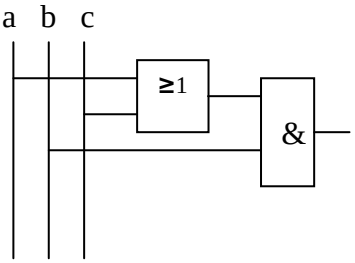
$x = \bar{a}\,b\,c + a\,b\,\bar{c} + a\,b\,c$

$x = a\,b\,(\bar{c} + c) + b\,c\,(\bar{a} + a)$

$x = a\,b + b\,c$

$x = b\,(a + c)$

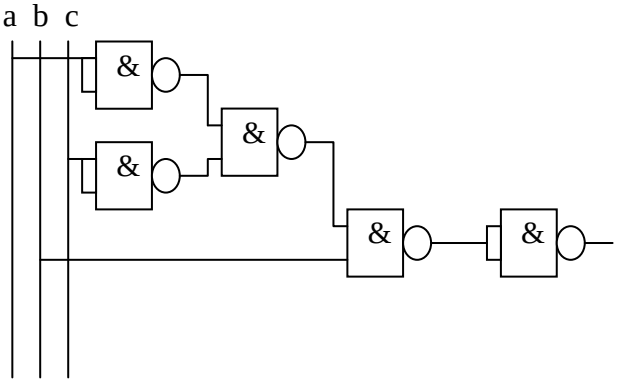




Umformen nach NAND

$x = \overline{\overline{b}\;\overline{(a + c)}}$

$x = \overline{\overline{b}\;\overline{(\bar{a} \bullet \bar{c})}}$



Wahrheitstabelle

a	**b**	**c**	**x**
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

Zusammenhänge verstehen

Um das bisher Gelernte besser zu verstehen und Gatter, Schaltalgebra, Vereinfachungen, Umformungen, Wahrheitstabelle etc. in einen verständlichen Zusammenhang zu bringen, schauen wir uns die Planung einer einfachen Schaltung an.

Planung einer Schaltung

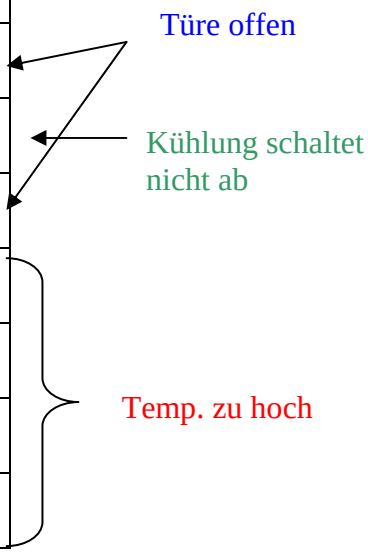
Zu einem Lagerraum für best. Lebensmittel soll eine Kontrollleuchte geschaltet werden. Sie ist verbunden mit einem Temperatursensor, der Kühlung und dem Türschloss.

Der Temperatursensor setzt eine Leitung (a) auf log. 1, wenn es wärmer als -5° C. wird.
Die Kühlung schaltet eine Leitung (b) auf log. 0, wenn sie abschaltet.
Das Türschloss ist mit einer Leitung (c) verbunden, die auf log.1 geht, wenn es auf „offen“ steht.

Die Kontrollleuchte (x) soll in jedem Fall anzeigen, wenn die Türe offen steht oder die Temperatur zu hoch ist. Sie muss aber auch leuchten, wenn die Temperatur unter -5° C. ist und die Kühlung nicht abschaltet.

Zuerst müssen wir alle Zustände bei denen die Kontrollleuchte anzeigen soll, also log. 1 führen soll, ausfindig machen. Dazu ist es hilfreich, eine Wahrheitstabelle zu erstellen und alle möglichen Kombinationen einzutragen. (in der Tabelle schwarz)

Temp. (a)	Kühl. (b)	Tür (c)	LED (x)	Term
0	0	0		
0	0	1	**1**	$\bar{a}\,\bar{b}\,c$
0	1	0	**1**	$\bar{a}\,b\,\bar{c}$
0	1	1	**1**	$\bar{a}\,b\,c$
1	0	0	**1**	$a\,\bar{b}\,\bar{c}$
1	0	1	**1**	$a\,\bar{b}\,c$
1	1	0	**1**	$a\,b\,\bar{c}$
1	1	1	**1**	$a\,b\,c$



Wir können nun im Voraus sagen, dass die LED leuchten soll, wenn die Temperatur zu hoch ist. In der Wahrheitstabelle wird das durch eine 1 bei Temp. (a) angezeigt.
Also, in jeder Zeile in der die Spalte Temp. eine 1 führt, muss auch bei LED eine 1 stehen. (in der Tabelle rot eingetragen)

„Türe offen“, wird in der Tabelle auch mit einer 1 in der Spalte „Tür (c)“ angezeigt. Nun auch in jeder Zeile in der die Spalte „Tür“ eine 1 führt bei LED eine 1 eintragen.
(in der Tabelle blau)
*Dort wo vom ersten Schritt schon eine 1 steht, muss natürlich nicht noch mal eine 1 eingetragen werden

Jetzt noch die 1 für die Aussage:
„Kühlung schaltet nicht ab, obwohl die Temp. unter -5° C. beträgt und die Tür geschlossen ist.“ (in der Tabelle grün)

Nun sind alle Zustände bei denen die Kontrollleuchte brennen soll definiert. Eine nützliche Methode für den Anfang ist es, danach die dazugehörenden Terme daneben einzutragen. So kann man sie direkt aus der Tabelle in die Gleichung übernehmen.
Das Durchnummerieren der Terme macht es einfacher den Überblick zu behalten.

Vereinfachen

$$x = \overline{a}\,\overline{b}\,c + \overline{a}\,b\,\overline{c} + \overline{a}\,b\,c + a\,\overline{b}\,\overline{c} + a\,\overline{b}\,c + a\,b\,\overline{c} + a\,b\,c$$

1 2 3 4 5 6 7

$$x = \overline{b}\,c\,(\overline{a} + a) + \overline{a}\,b\,(\overline{c} + c) + a\,\overline{b}\,(\overline{c} + c) + a\,b\,(\overline{c} + c)$$

1 / 5 2 / 3 4 / 5 6 / 7

$$x = \overline{b}\,c\,(1) + \overline{a}\,b\,(1) + a\,\overline{b}\,(1) + a\,b\,(1)$$

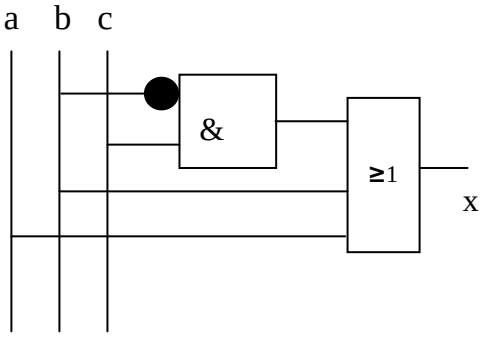
$$x = \overline{b}\,c + \overline{a}\,b + a\,\overline{b} + a\,b$$

1 2 3 4

$$x = a\,(\overline{b} + b) + b\,(\overline{a} + a) + b\,c$$

3 / 4 2 / 4

$x = a + b + \overline{b}\,c$

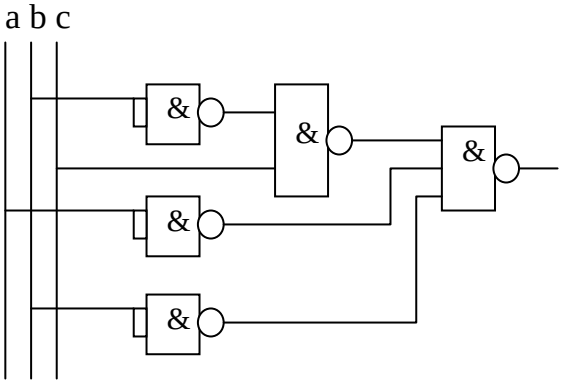


Die Firma verwendet nur NAND – Gatter. Die Gleichung muss also umgeformt werden.

$$x = \overline{\overline{a + b + \overline{b}\,c}}$$

Nach doppelter Negation auftrennen bei OR

$$x = \overline{\overline{a} \bullet \overline{b} \bullet \overline{\overline{b}\,c}}$$



Testen Sie Ihr Wissen

Zeichnen Sie auf, wie man aus einem NAND – Gatter ein NOT-Gatter macht!

1. Formen Sie folgenden schaltalgebraischen Ausdruck so um, dass er ausschließlich mit NAND aufgebaut werden kann.

$x = \overline{(a + b)} + (b\ c)$

$x =$

$x =$

$x =$

3. Zeichnen Sie die dazugehörende Schaltung!

a b c

4. Vereinfachen Sie folgende Schaltung!

$x = a\,b\,\overline{c} + \overline{a}\,b\,c + a\,b\,c + \overline{\overline{a}\,b\,c}$

$x =$

$x =$

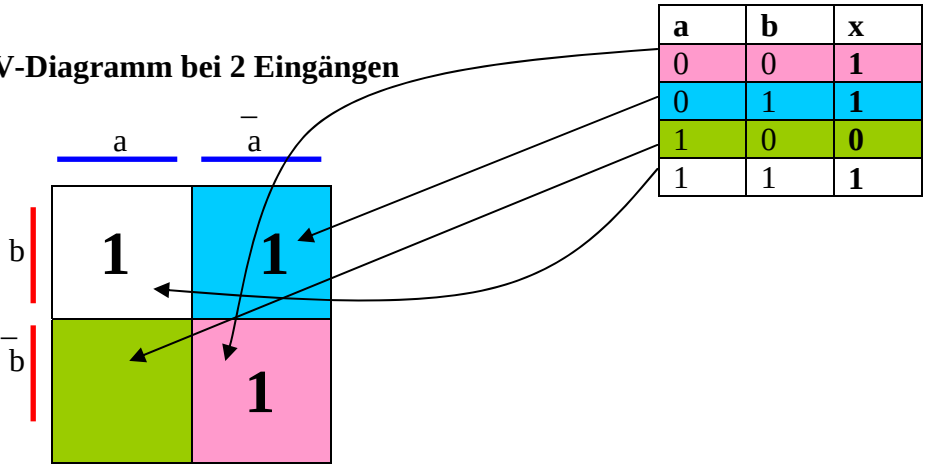
K V - Methode

Kapitel 4

Vereinfachen von Schaltungsgleichungen anhand des Karnaugh-Veitch Diagramms

KV-Diagramme dienen der übersichtlichen Darstellung und der Vereinfachung von Schaltungsgleichungen. Sie sind grafische Hilfen zur Reduzierung der Anzahl der Glieder im schaltalgebraischen Ausdruck. Das Grundprinzip eines KV-Diagramms ist die Darstellung der Wahrheitstabelle in einer Matrix. Diese besitzt so viele Felder wie die Tabelle Zeilen besitzt. D.h., wenn wir eine Schaltung mit 4 Eingängen beschreiben wollen, brauchen wir zur KV-Darstellung eine Matrix mit 4^2 (=16) Feldern. Nachteil dieser Methode ist, dass bei einer größeren Anzahl von Eingängen die Übersichtlichkeit verloren geht. Bei bis zu 4 Eingängen ist es aber die schnellste Methode eine Schaltung zu vereinfachen.

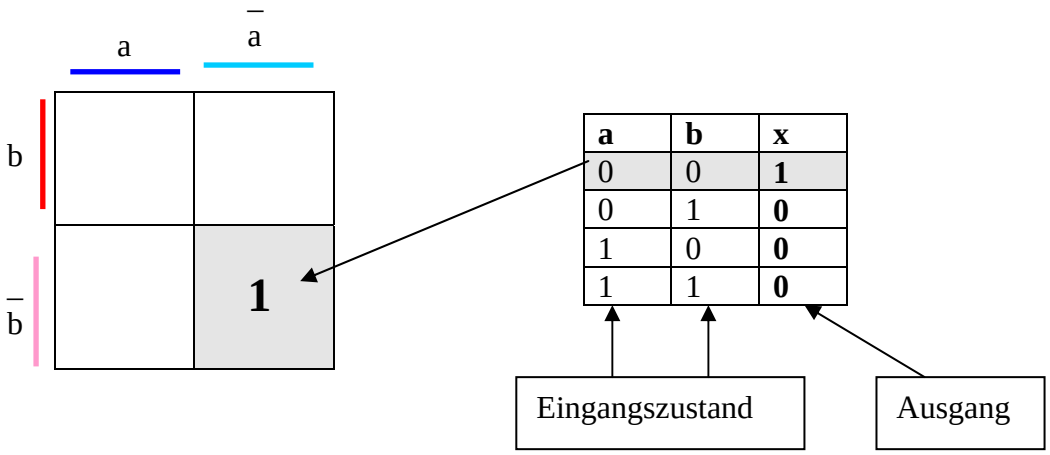
Beispiel: KV-Diagramm bei 2 Eingängen



a	b	x
0	0	**1**
0	1	**1**
1	0	**0**
1	1	**1**

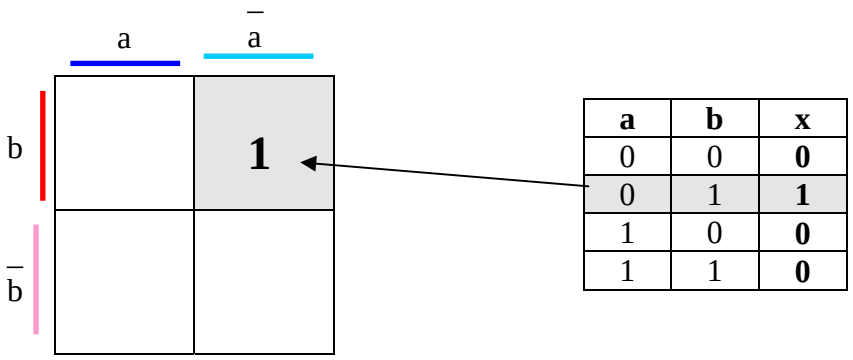
- Ein KV-Diagramm hat immer so viele Plätze, wie insgesamt mögliche Eingangszustände.
  (2 Variablen = 4 Plätze)
  (3 Variablen = 8 Plätze)
  (4 Variablen =16 Plätze)
- Jede vorkommende Variable wird in negierter und nicht negierter Form an den Rand des KV-Diagramms geschrieben. (Sobald man mit dieser Darstellungsart etwas vertraut ist, braucht man die negierte Variable nicht mehr zu kennzeichnen)
- Eine log. 1 am Ausgang wird durch eine 1 in das entsprechende Kästchen angegeben.
- Benachbarte Einsen können als Päckchen zusammengefasst werden.
- Eine log. 0 am Ausgang kann zur negierten Form der Vereinfachung herangezogen werden. (Alle Nullen werden zu Päckchen zusammengefasst)
- Die Päckcheninhalte werden mit OR verbunden und ergeben so die vereinfachte Gleichung.
  (oben: $\mathbf{x = a + b}$ oder in der negierten Form: $\mathbf{\bar{x} = \bar{a}\,\bar{b}}$

Zuordnung der KV-Diagrammbereiche für eine Schaltung mit 2 Eingängen



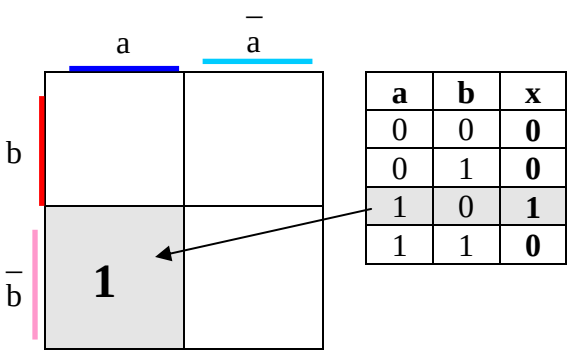
a	b	x
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Wahrheitstabelle und KV-Diagramm sagen dasselbe aus. Wenn also in der 1. Zeile der Wahrheitstabelle beim Ausgang x eine log. 1 steht, dann wird diese Eins in das untere rechte Kästchen des KV-Diagramms eingetragen. Denn der Eingangszustand für diese log. 1 ist **NOT a, NOT b**.
In der Schaltungsgleichung wäre diese Eins **x = $\bar{a}$ $\bar{b}$**
Ein Kästchen beschreibt also immer alle Eingänge und den Ausgang, genauso wie in der Wahrheitstabelle eine Zeile jeweils den Zustand aller Eingänge und den des Ausgangs beschreibt.



a	b	x
0	0	**0**
0	1	**1**
1	0	**0**
1	1	**0**

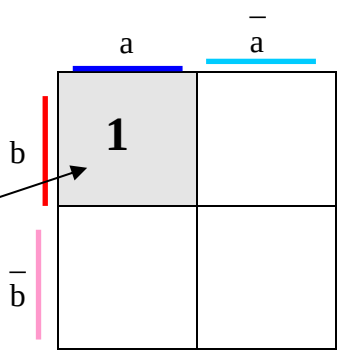
Das rechte obere Kästchen kennzeichnet den Bereich für **NOT a, b**. Eine Eins in diesem Kästchen sagt also aus: Wenn am Eingang a eine log. 0 und an Eingang b eine log. 1 anliegt, dann ist der Ausgang x log. 1
In der Schaltungsgleichung wäre diese Eins **x = $\bar{a}$ b**



a	b	x
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**0**

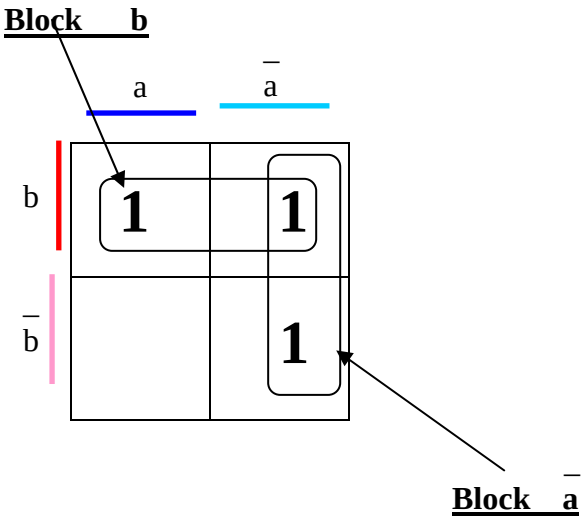
Schaltungsgleichung: **x = a $\bar{b}$**

a	b	x
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**



Schaltungsgleichung: **x = a b**

Päckchenbildung (Blockbildung)
Ziel ist es, im KV-Diagramm größtmögliche Blöcke zu bilden, die dann jeweils einem Glied in der vereinfachten Gleichung entsprechen. Die einzelnen Glieder werden in der Gleichung mit einer OR-Verknüpfung verbunden. Wenn wir uns das folgende Diagramm anschauen, sehen wir, dass man benachbarte Felder zu Blöcken zusammenfassen kann, um die Schaltung zu vereinfachen.



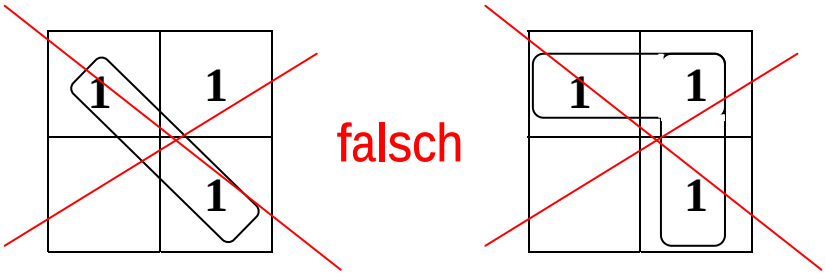
a	b	x
0	0	**1**
0	1	**1**
1	0	**0**
1	1	**1**

Aus der Tabelle ist erkennbar, dass immer wenn am Eingang **a** eine **log. 0** anliegt, der Ausgang x eine log. 1 führt, **oder** wenn an Eingang **b** eine **log. 1** anliegt, führt der Ausgang x auch eine log. 1
In der Schaltungsgleichung ist das: $\mathbf{x = \bar{a} + b}$

Gleichung $\mathbf{x = \bar{a} + b}$

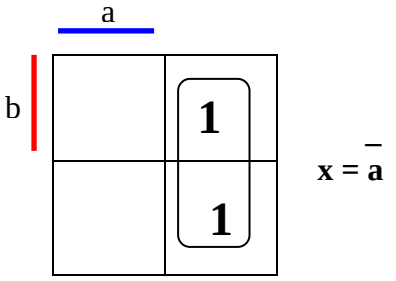
Allg. Hinweise zur Blockbildung in KV-Diagrammen:

Um eine größtmögliche Vereinfachung zu erhalten, sollten die Päckchen so groß wie möglich sein. Die Blöcke dürfen nur senkrecht oder waagrecht gebildet werden, nicht diagonal und nicht übers Eck. **Aber, Päckchen dürfen sich überlappen** (siehe oben).

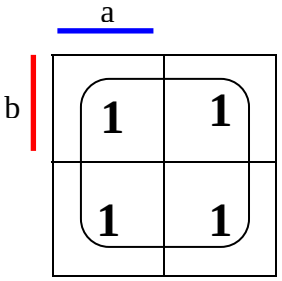


Hinweise zur Blockbildung in KV-Diagrammen mit 2 Variablen:

Ein Päckchen darf 2 oder 4 benachbarte Felder haben.



$\mathbf{x = \bar{a}}$



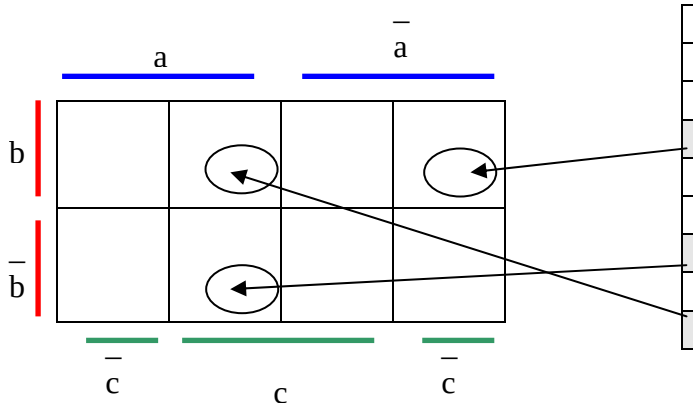
$\mathbf{x = a\,\bar{a}\;b\;\bar{b}}$

entspricht

x = 1

KV-Diagramme mit 3 Variablen:

Das Prinzip des KV-Diagramms bleibt bei drei Eingängen dasselbe wie bei einer Schaltung mit zwei Eingängen. Man braucht nur mehr Platz, um alle Zeilen der Wahrheitstabelle dazustellen. Eine Wahrheitstabelle für eine Schaltung mit drei Eingängen besitzt 2^3 Zeilen, also benötigt man im KV-Diagramm 2^3 Kästchen. Die dritte Variable erhält die untere Diagrammseite (hier c)



a	b	c	x
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

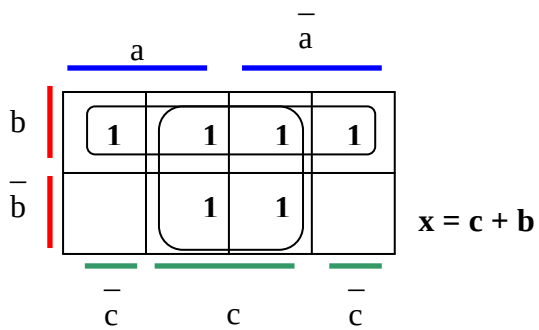
An drei Zeilen ist die Zellzuordnung demonstriert.

Der untere Pfeil zeigt auf die Zelle: a, b, c → a b c
Der mittlere Pfeil zeigt auf die Zelle: a, NOT b, c → a $\bar{b}$ c
Der oberste Pfeil zeigt auf die Zelle: NOT a, b, NOT c → $\bar{a}$ b $\bar{c}$

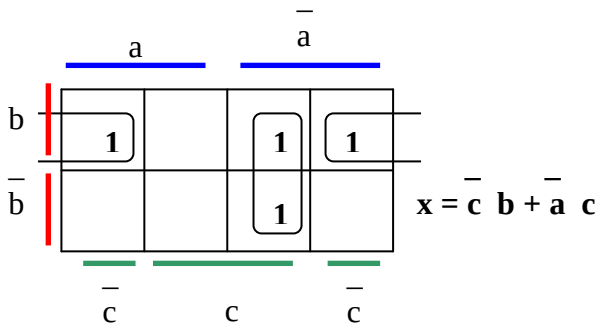
Hinweise zur Blockbildung in KV-Diagrammen mit 3 Variablen:

Zusätzlich zu den Regeln der Blockbildung bei zwei Eingängen können Einsen, die am rechten und linken Rand sind, ebenfalls zu Päckchen zusammengefasst werden.

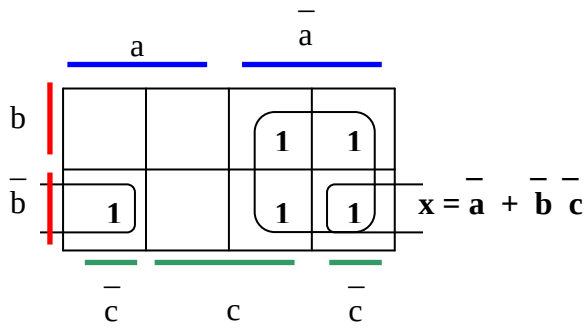
Beispiele zur Blockbildung:



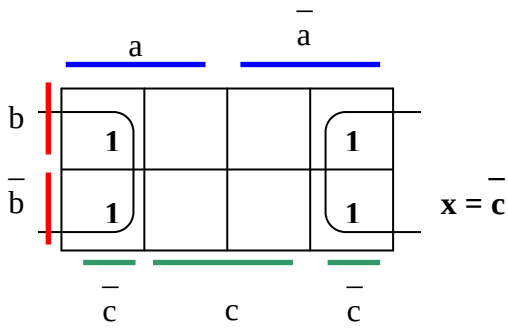
x = c + b



x = $\bar{c}$ b + $\bar{a}$ c



x = $\bar{a}$ + $\bar{b}$ $\bar{c}$



x = $\bar{c}$

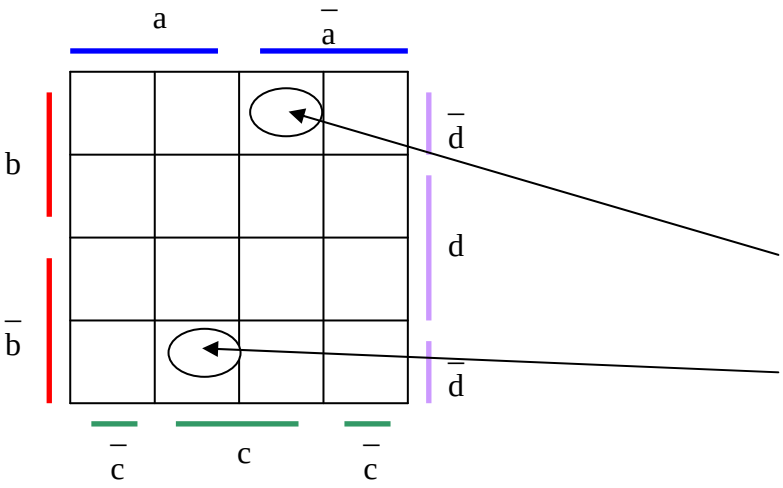
An den Beispielen wird deutlich, je größer ein Block ist, desto mehr Variablen entfallen in der Schaltungsgleichung.

Wir haben die Möglichkeit 2er, 4er oder 8er Blöcke zu bilden.
Dazu folgender Merksatz:

2er Block → eine Variable entfällt
4er Block → zwei Variablen entfallen
8er Block → drei Variablen entfallen

KV-Diagramme mit 4 Variablen:

Das Prinzip des KV-Diagramms ist immer das Gleiche. Nun brauchen wir noch zusätzlich Platz für die vierte Variable. Eine Wahrheitstabelle für eine Schaltung mit vier Eingängen besitzt 2^4 Zeilen, so benötigt man im KV-Diagramm 2^4 Kästchen. Die vierte Variable erhält die rechte Diagrammseite (hier d)



a	b	c	d	x
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

An zwei Zeilen sehen wir wieder die Zellzuordnung.

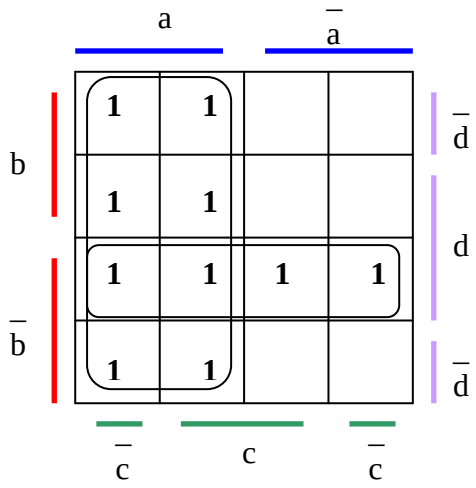
Hinweise zur Blockbildung in KV-Diagrammen mit 4 Variablen:

Zusätzlich zu den bisherigen Möglichkeiten der Blockbildung können Einsen, die am oberen und unteren Rand sind, ebenfalls zu Päckchen zusammengefasst werden. Es können auch die vier Ecken zu einem Block zusammengefasst werden.

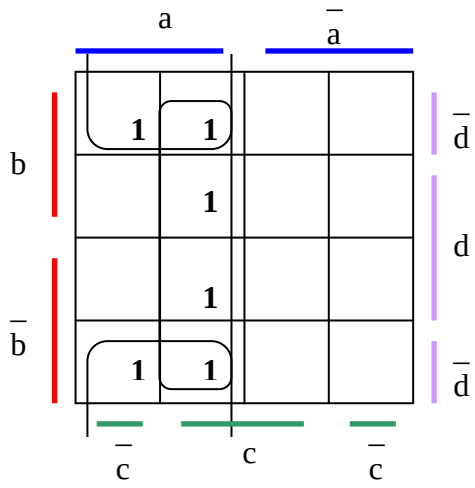
Beispiele zur Blockbildung:

Schauen Sie sich nun die Beispiele auf der nächsten Seite an und versuchen Sie die Aufgabe zum „Selbst lösen“. Sollten Sie diese Aufgabe nicht lösen können, dann hilft Ihnen sicherlich das darauf folgende Aufgabenbeispiel weiter. Arbeiten Sie dieses durch und versuchen Sie sich danach noch mal an dem Beispiel zum „Selbst lösen“.

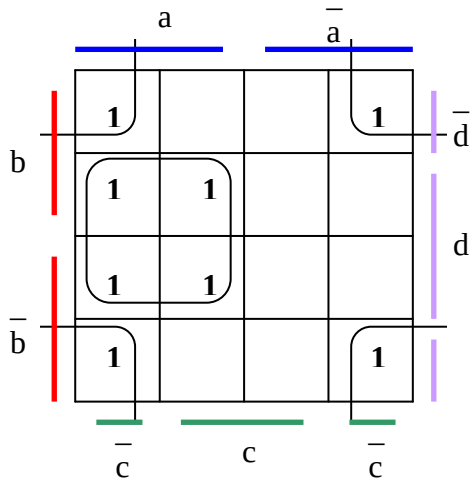
Beispiele zur Blockbildung:



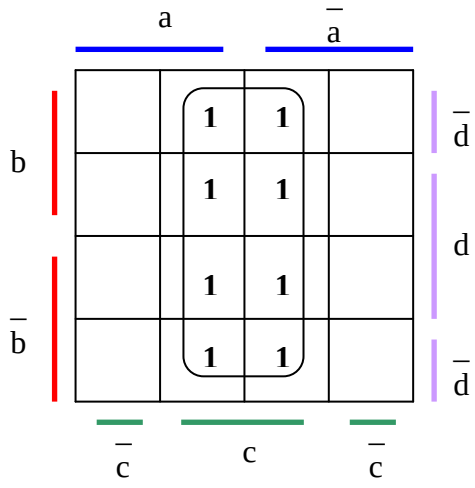
$x = a + \bar{b}\ d$



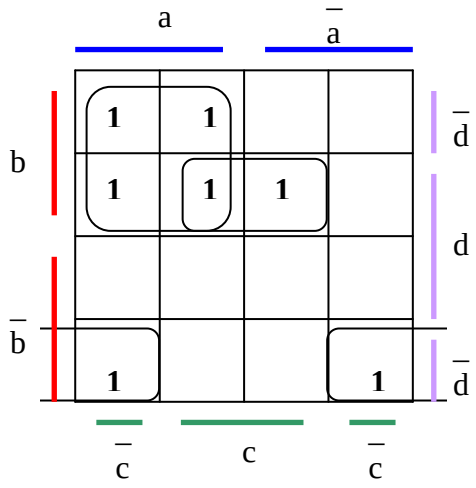
$x = a\ c + a\ \bar{d}$



$x = \bar{c}\ \bar{d} + a\ d$

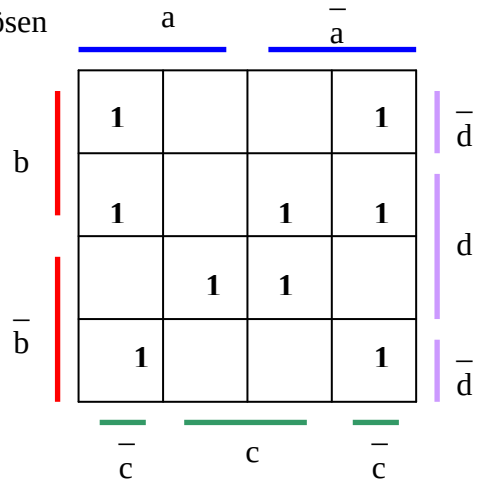


$x = c$



$x = a\ b + b\ c\ d + \bar{b}\ \bar{c}\ \bar{d}$

Selbst lösen

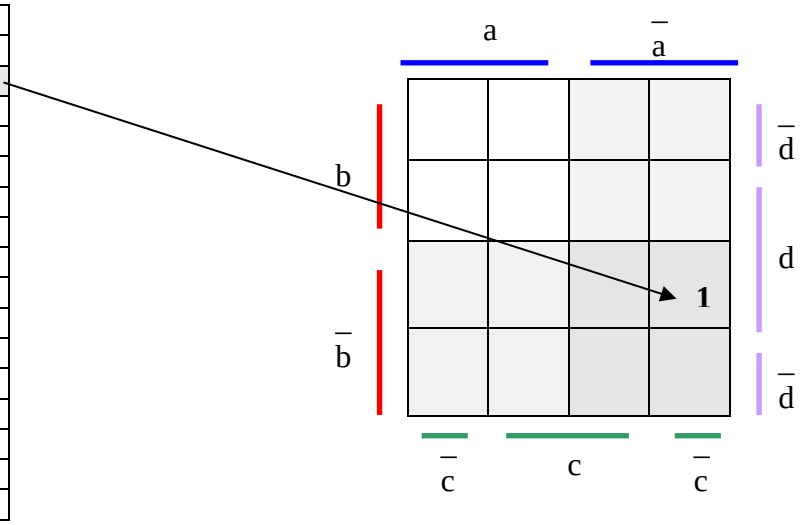


$x =$

Aufgabenbeispiel:

Sie sind im Begriff eine Schaltung zu erstellen und haben folgende Wahrheitstabelle ausgearbeitet:

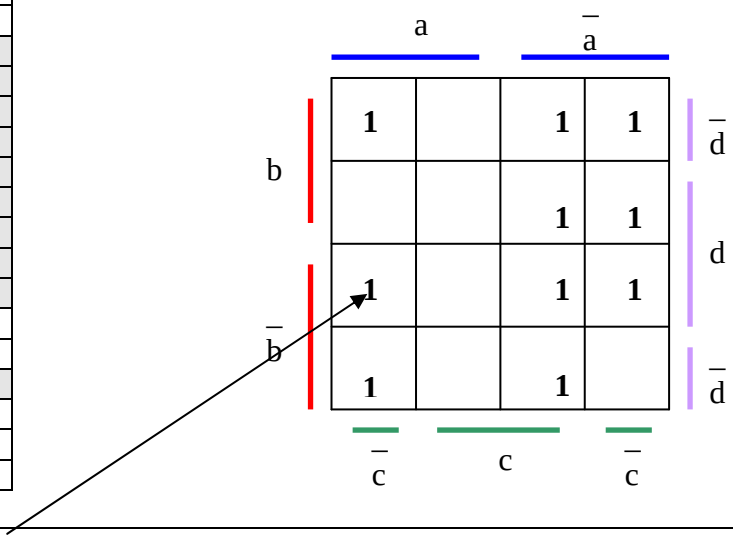
a	b	c	d	x
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Alle Zeilen, in denen der Ausgang x eine log.1 führt, sind für uns wichtig. Diese Einsen übertragen wir ins KV-Diagramm. Wenn wir in der Wahrheitstabelle von oben anfangen, dann steht die erste zu übertragende Eins in der Zeile:
a = 0, b = 0, c = 0, d = 1
Weiter oben im Kapitel haben wir ja gelernt, dass jedes Kästchen des KV-Diagramms eine Zeile der dazugehörenden Wahrheitstabelle beschreibt.
Nun müssen wir das dazugehörende Kästchen finden.
„a = 0“ bedeutet im KV-Diagramm der Bereich „NOT a“ → $\bar{a}$
„b = 0“ bedeutet im KV-Diagramm der Bereich „NOT b“ → $\bar{b}$
„c = 0“ bedeutet im KV-Diagramm der Bereich „NOT c“ → $\bar{c}$
„d = 1“ bedeutet im KV-Diagramm der Bereich „d“ → d

Das dazugehörende Kästchen ist also die Schnittmenge dieser 4 Bereiche. Es kann genau definiert werden. Markieren wir den Bereich „NOT a“ und „NOT b“, so erhalten wir eine Schnittmenge von 4 Kästchen. Innerhalb dieser Schnittmenge ist das Kästchen „NOT c“, „ d“ leicht auszumachen. Dort tragen wir unsere erste Eins ein und so verfahren wir mit allen Zeilen, deren Ausgang x eine log. 1 führt.

a	b	c	d	x
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Blockbildung (Vereinfachung der Schaltung)

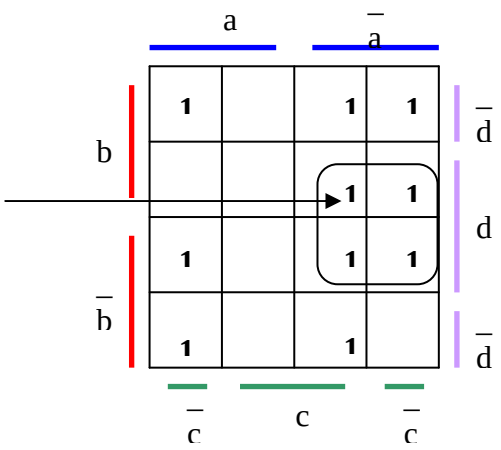
Zur Verdeutlichung, dass die KV-Methode wirklich etwas bringt, schreibe ich die „Monstergleichung“ hin, die anhand von Schaltalgebra zu vereinfachen wäre.

$$x = \bar{a}\,\bar{b}\,\bar{c}\,\bar{d} + \bar{a}\,\bar{b}\,c\,\bar{d} + \bar{a}\,b\,\bar{c}\,\bar{d} + \bar{a}\,b\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,\bar{d} + a\,\bar{b}\,c\,\bar{d} + a\,b\,\bar{c}\,\bar{d} + a\,b\,c\,\bar{d} + a\,\bar{b}\,\bar{c}\,\bar{d} + a\,\bar{b}\,c\,\bar{d} + a\,b\,\bar{c}\,\bar{d}$$

Wenden wir uns nun aber der Vereinfachung mittels KV-Diagramm zu. Wir arbeiten uns immer von den größtmöglichen Blöcken zu den kleinstmöglichen oder einzelnen Einsen vor. Die gefundenen Blöcke tragen wir immer sofort in die Gleichung ein.
Der größtmögliche Block bei einem KV-Diagramm für 4 Eingänge ist ein 8er Block.
(*von einem 16er Block mal abgesehen. Aber dann wäre die Gleichung: x = 1
Und dann bräuchten wir auch keine Schaltung, sondern könnten den Ausgang x an eine Leitung legen, die log. 1 führt)

8er Block → kein 8er Block möglich
4er Block → zwei 4er Blöcke möglich
aber nur einer notwendig

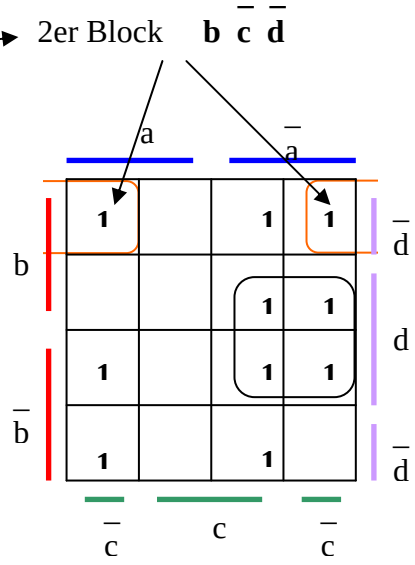
$x = \bar{a}\,d$ → 4er Block



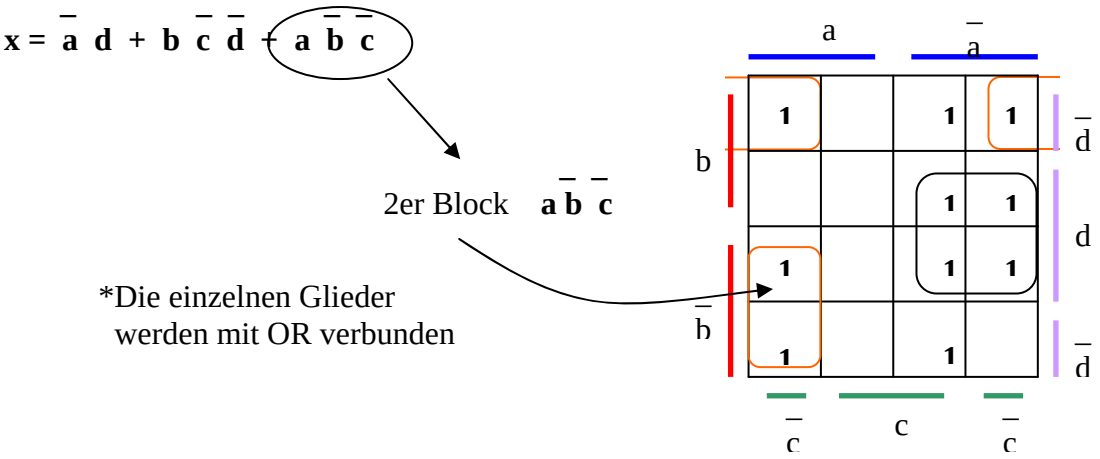
Normalerweise arbeitet man natürlich mit **einem** KV-Diagramm und **einer** Gleichung. Der besseren Übersichtlichkeit wegen, zerlegen wir hier die Schritte und erstellen die Blöcke mit verschiedenen Farben. (4er Blöcke in grau, 2er Blöcke in orange)

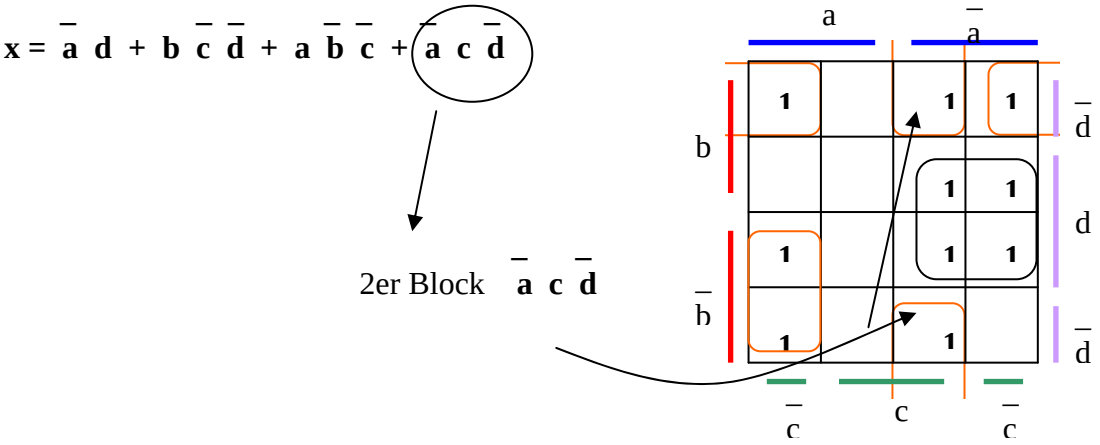
2er Block → drei 2er Blöcke möglich

$x = \bar{a}\,d + b\,\bar{c}\,\bar{d}$ → 2er Block $b\,\bar{c}\,\bar{d}$



*Schauen Sie sich auch noch mal den Merksatz weiter oben im Kapitel an.
z.B. 2er Block → eine Variable entfällt





Die vollständig vereinfachte Schaltungsgleichung in der nicht negierten Form lautet:

$$x = \bar{a}\,d + b\,\bar{c}\,\bar{d} + a\,\bar{b}\,\bar{c} + \bar{a}\,c\,\bar{d}$$

Ist Ihnen aufgefallen, dass es nicht immer vorteilhaft ist, alle größtmöglichen Blöcke zu bilden? Zunächst sah es oben nämlich so aus, als ob man am besten zwei, sich überlappende 4er Blöcke bildet. Das wäre aber nicht die bestmögliche Vereinfachung gewesen.

Da wir zur Erstellung der Schaltung immerhin 12 Gatter benötigen, überlegen wir uns, ob es nicht sinnvoller ist, die Schaltung in der negierten Form zu vereinfachen. D.h., die Funktion des Ausgang x auf Null zu erstellen. (Ausgang x = 0, wenn ……)
Dazu füllen wir die bisher leer stehenden Kästchen mit Nullen und bilden damit die Blöcke.

Die vereinfachte Schaltungsgleichung in der negierten Form lautet:

$$\bar{x} = a\,c + a\,b\,d + \bar{a}\,\bar{b}\,\bar{c}\,\bar{d}$$

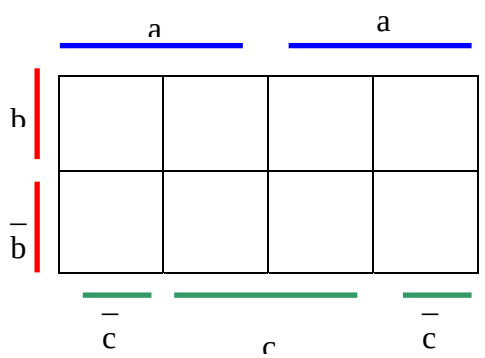
Die negierte Form ist hier eindeutig günstiger, da wir nun mit 9 Gattern auskommen.

Testen Sie Ihr Wissen

Ermitteln Sie zu jeder Gleichung oder Tabelle die vereinfachte Gleichung.

1.)

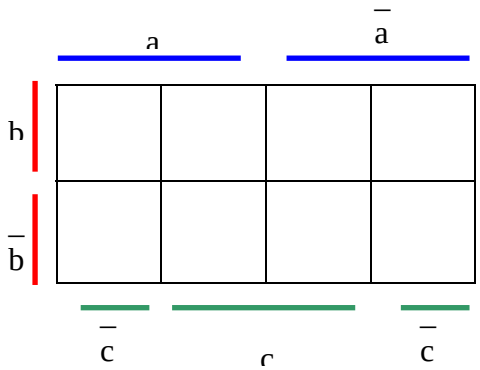
a	b	c	x
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**



x =

2.)

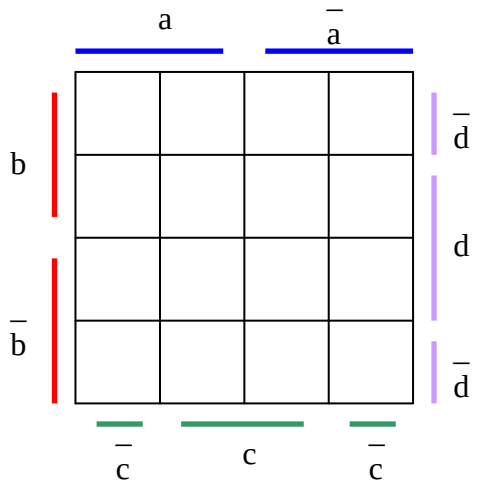
$$x = \bar{a}\,\bar{b}\,c + a\,b\,\bar{c} + \bar{a}\,b\,c + a\,\bar{b}\,\bar{c} + a\,\bar{b}\,c + a\,b\,\bar{c} + a\,b\,c$$



x =

3.)

a	b	c	d	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



x =

Digi Trace

Kapitel

5

Simulationssoftware
für logische Schaltungen

In diesem Kapitel geht es darum, sich mit dem Simulationsprogramm Digi Trace vertraut zu machen. Es folgt zunächst ein Überblick über die Programmoberfläche, den wichtigsten Toolbar-Funktionen und Arbeitsoberflächen und danach ein Auszug aus der Hilfe-Funktion des Programms.

Hilfe-Funktionen von Programmen sind häufig vergleichbar mit einer Bedienungsanleitung. Lesen Sie also die folgenden Seiten noch heute durch. Auch wenn Sie vielleicht nicht alles auf Anhieb verstehen, so kennen Sie danach doch die Funktionsweise und den Umfang von DigiTrace. Und da wir unsere Schaltungen für eine geraume Zeit mit dieser Software entwickeln und simulieren, sparen Sie sich somit viel Zeit.

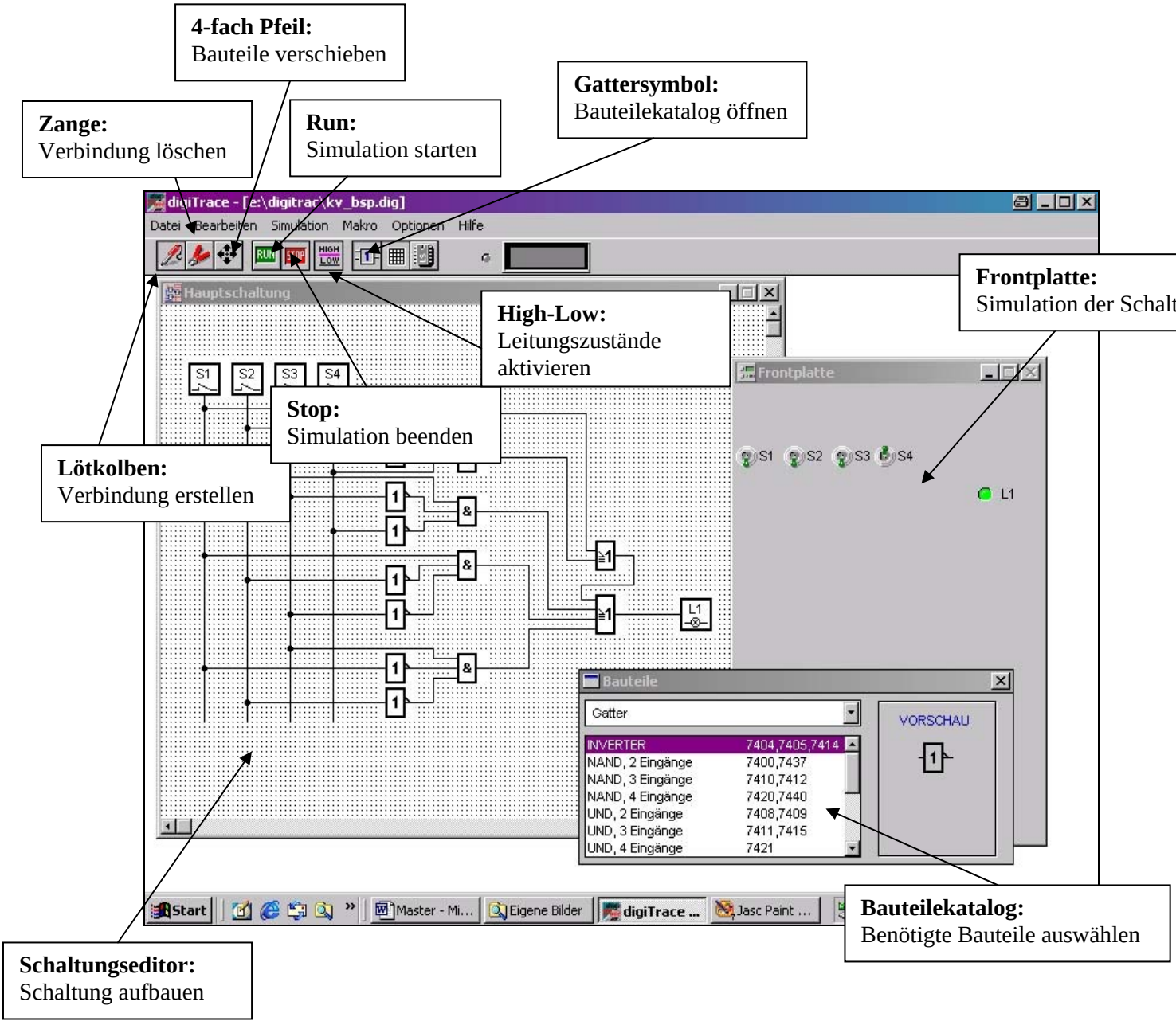
Am Schluss des Kapitels stelle ich noch den Aufbau einer sehr einfachen Schaltung, Schritt für Schritt, vor. Es handelt sich dabei um die Funktionsweise oder das „Innenleben“ eines RS-Latch. Viele Funktionen im Computer werden von solchen Speicherbausteinen erledigt. Mit dieser Schaltung starten wir im nächsten Praktikum.

(*da ich den Auszug aus der Hilfe auf den Rahmen unserer Ausbildung zugeschnitten habe, fehlen einige Abschnitte die im Orginal vorhanden sind. Um aber dessen Funktion als Nachschlagewerk nicht zu verlieren, sind auch diverse Abschnitte dabei, die wir zur Ausarbeitung noch folgender Praktika nicht brauchen. Vor allem im Bereich „Bauteile“.)

Ansicht der Programmoberfläche 'Digi Trace'

Mit DigiTrace werden logische Schaltungen am PC entworfen und simuliert. Das Programm wird aus der Windows – Umgebung gestartet. (Doppelklick auf dieses Symbol)

Schon auf den ersten Blick sieht man, ein sehr einfach zu bedienendes Programm.
Die Toolbar präsentiert in übersichtlicher Form die am häufigsten verwendeten Funktionen.



Auf den nächsten Seiten folgt ein Auszug der für uns wichtigen Hilfe – Seiten.
(*nicht vollständig)

Digi Trace – Logiksimulator Hilfe

Inhalt

Grundlagen

Mit DigiTrace können digitale Schaltungen auf dem PC entworfen und simuliert werden. Die wesentlichen Merkmale von digiTrace sind:

- Digital -Simulator zur Simulation von Schaltungen aus logischen Bauteilen
- Schaltungseditor zur schematischen Eingabe der Schaltung
- Makrofunktionen zum Zusammenfassen von eigenen Schaltungen
- Frontplatteneditor zur ansprechenden optischen Gestaltung und einfachen Bedienung der Schaltung
- Viele Bedienungs- und Anzeigeelemente für die Frontplatte
- Hardware-Ein- und Ausgänge zur Anbindung externer Schaltungen über LPT-Druckerport(s)
- TTL-Katalog mit einer großen Auswahl oft benötigter Logik-IC´s
- Soundblaster-Unterstützung zur Ausgabe akustischer Signale
- Berichtsfunktion zur Dokumentation der Schaltung

Bitte beachten Sie die mit dem Programm gelieferten Beispiele, die Ihnen einen schnellen Überblick über die Leistungsfähigkeit von DigiTrace ermöglichen und die Ihnen wertvolle Tipps und Anregungen für eigene Projekte liefern können.

Die Dateiendungen haben in digiTrace folgende Bedeutungen:

*.DIG	DigiTrace - Projektdateien
*.MKR	Makro-Dateien
*.BMP	Bitmap-Dateien für die Frontplatte (16-Farben)
*.WAV	Audio-Dateien für die Lautsprecher
*.HLP	Hilfe- und Infodateien
*.BTF	Programm-Dateien für die Bauteile
*.INF	Programm-Dateien für den TTL-Katalog

Hardware

Eines der wesentlichen Merkmale von DigiTrace ist die Möglichkeit, 'echte' Schaltungen mit den Schaltungen von DigiTrace zu kombinieren.
Dies geschieht über die parallelen Schnittstellen LPT1, LPT2 und LPT3 oder über geeignete Schnittstellenkarten.

Verwendung der LPT-Ports

Jeder LPT-Port stellt acht Ausgänge und fünf Eingänge zur Verfügung, die mit DigiTrace angesprochen werden können. Somit steht bei der Verwendung von 3 Schnittstellen eine maximale Anzahl von 24 Ausgängen und 15 Eingängen zur Verfügung.
Bitte achten Sie darauf, dass Sie keine Eingangsspannungen >5V an die Eingänge legen und den maximalen Ausgangsstrom der Ausgänge nicht überschreiten. Geeignete Anschlussschaltungen finden Sie in großer Zahl in der einschlägigen Fachliteratur.
Im Folgenden ist die Anschlussbelegung des LPT-Ports (SubMin D 25) wiedergegeben, soweit die Anschlüsse für die Verwendung mit DigiTrace von Bedeutung sind:

Ausgänge:

PIN 2...9	Datenausgänge D0...D7

Eingänge:

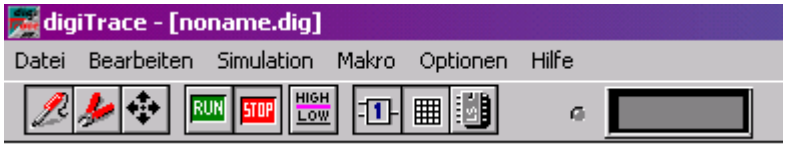
PIN 10	ACK
PIN 1	BUSY
PIN 12	PE
PIN 13	SLCT
PIN 15	ERROR

Bedienung

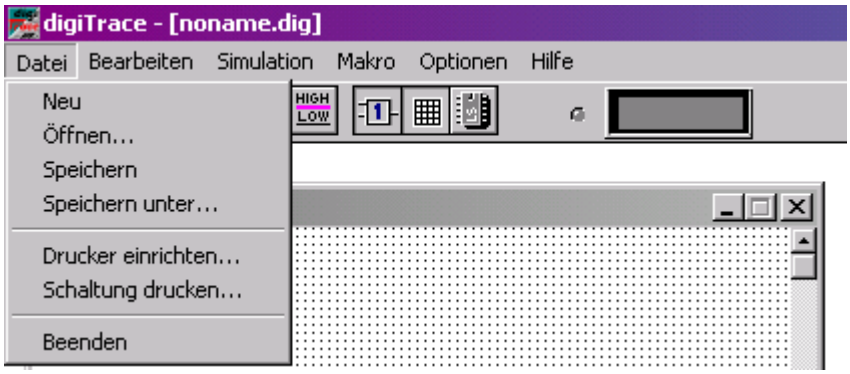
Starten der Software

Nach der Installation steht Ihnen in der Programm-Gruppe 'DigiTrace' das Programmsymbol der DigiTrace-Software zu Verfügung
Starten Sie das Programm wie gewohnt. Nach dem Programmstart erscheint das Hauptmenü des Programms. Wenn im Arbeitsverzeichnis des Programms einen Datei mit dem Namen NONAME.DIG existiert, so wird diese Schaltung bei Programmstart automatisch geladen.

Das Menü

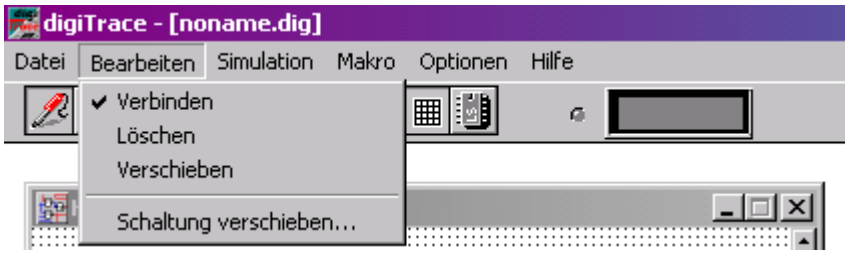


Datei



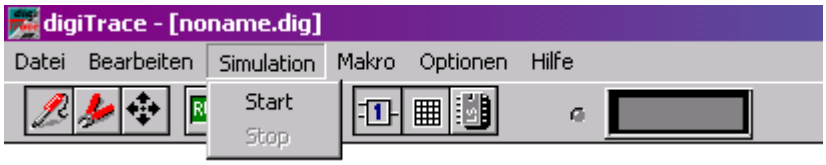
Neu	→	löscht nach einer Sicherheitsabfrage die aktuelle Schaltung.
Öffnen…	→	lädt eine gespeicherte Schaltung.
Speichern	→	speichert die aktuelle Schaltung.
Speichern unter...	→	speichert die aktuelle Schaltung unter einem neuen Namen.
Drucker einrichten...	→	wählt einen Drucker aus.
Schaltung drucken…	→	druckt die Schaltung.
Beenden	→	beendet DigiTrace.

Bearbeiten



Verbinden → schaltet den Schaltungseditor in den Verbinden-Modus.
Löschen → schaltet den Schaltungseditor in den Lösch-Modus.
Verschieben → schaltet den Schaltungseditor in den Verschieben-Modus.
Schaltung verschieben… → öffnet ein Dialogfenster zum Verschieben der Schaltung im Schaltungseditor.

Simulation

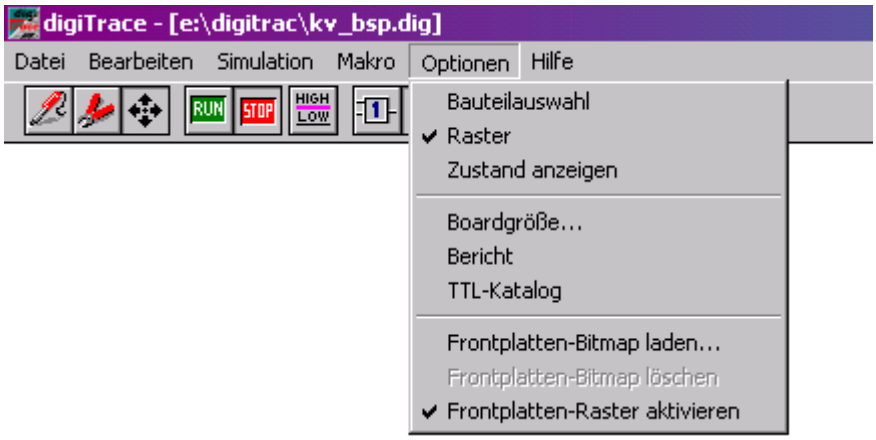


Start → startet die Simulation. Alle anderen Menüeinträge sind bei laufender Simulation deaktiviert.
Stop → stoppt die Simulation und schaltet in den Editiermodus.

Makro

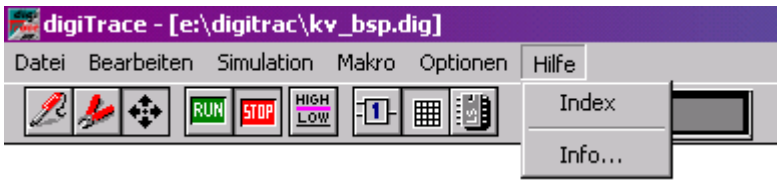
Hinzufügen… → öffnet eine Dateiauswahlbox aus der Sie das Makro wählen können, welches Sie zu der aktuellen Schaltung hinzufügen möchten.
Neues Makro → schaltet den Editor in den Makro-Modus. Sie können hier ein neues Makro erstellen.
Makro bearbeiten → schaltet den Editor in den Makro-Modus und öffnet gleichzeitig das Makro, welches Sie zuletzt bearbeitet haben.
Hauptschaltung bearbeiten → schaltet den Editor vom Makro-Modus zurück in den normalen Editiermodus der Hauptschaltung.

Optionen



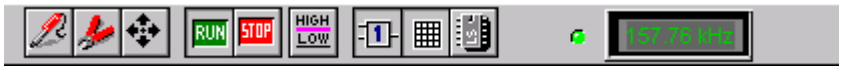
Bauteilauswahl → öffnet und schließt das Bauteil-Auswahlfenster.
Raster → schaltet das Raster des Schaltungseditors ein bzw. aus.
Zustand anzeigen → schaltet die Zustandsanzeige bei laufender Simulation ein bzw. aus. Ist diese Option gewählt, so werden bei laufender Simulation die aktuellen High- und Low-Zustände aller Drähte im Schaltungseditor angezeigt. Bitte beachten Sie, dass hierdurch die Simulationsgeschwindigkeit herabgesetzt wird.
Boardgröße… → stellt die Arbeitsfläche des Schaltungseditors ein.
Bericht → öffnet ein Textfenster mit Angaben zu Ihrer Schaltung. Der Bericht steht Ihnen als Datei mit dem Namen BERICHT.TXT zur Verfügung.
TTL-Katalog → öffnet ein Fenster mit einer Auswahl oft benötigter TTL-ICs und zeigt deren Anschlussbelegung an. Die Typen sind nach verschiedenen Gruppen sortiert. Diese ICs können nicht zur Schaltung hinzugefügt werden. Sie sollen bei der Auswahl geeigneter Bauelemente zur hardwaremäßigen Realisierung dienen.
Frontplatten-Bitmap laden… → lädt eine Bitmap-Datei in den Frontplatteneditor.
Frontplatten-Bitmap löschen → löscht die Frontplatten-Bitmap aus dem Frontplatteneditor.
Frontplatten-Raster aktivieren → schaltet den Rastermodus für die Positionierung von Bauteilen auf der Frontplatte ein bzw. aus.

Hilfe



Index → aktiviert die DigiTrace-Hilfe.
Info… → öffnet eine Box mit Versionsinformation zur DigiTrace-Software.

Die Toolbar



Über die Toolbar sind oft benötigte Menüpunkte schnell erreichbar.
Rechts von den Toolbarschaltern befindet sich die Frequenzanzeige, die bei laufender Simulation die Simulationsfrequenz anzeigt.

Verbinden schaltet den Schaltungseditor in den Verbinden-Modus.

Löschen schaltet den Schaltungseditor in den Löschen-Modus.

Verschieben schaltet den Schaltungseditor in den Verschieben-Modus.

RUN startet die Simulation.

STOP stoppt die Simulation und schaltet in den Editiermodus.

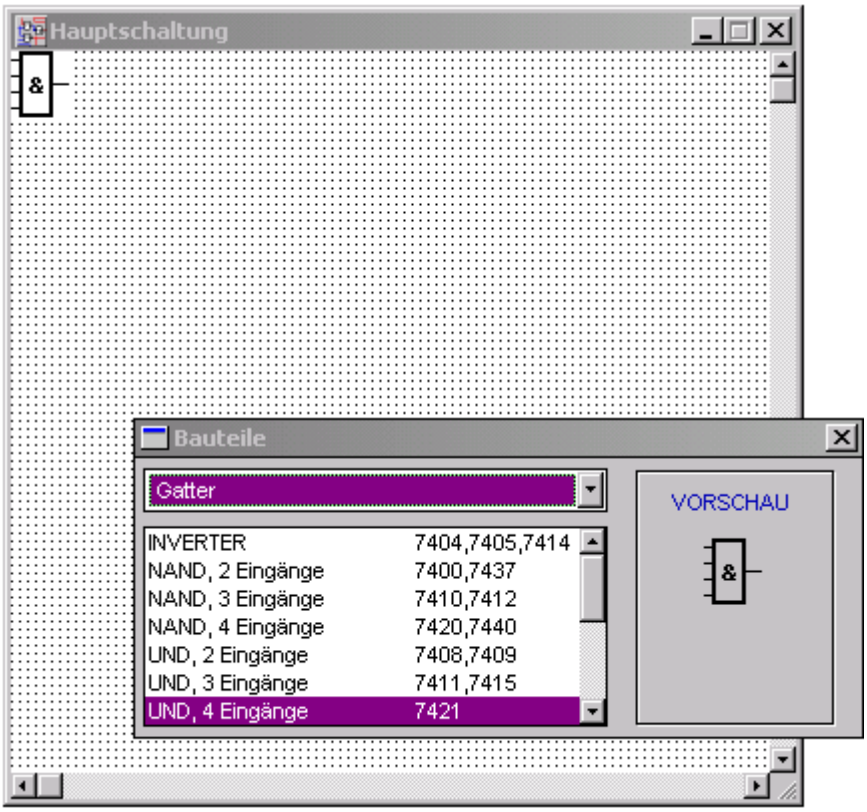
Zustand anzeigen schaltet die Zustandsanzeige bei laufender Simulation ein bzw. aus.

Bauteilauswahl öffnet und schließt das Bauteil-Auswahlfenster.

Raster schaltet das Raster des Schaltungseditors ein und aus.

TTL-Katalog öffnet ein Fenster mit einer Auswahl oft benötigter Logik-ICs und zeigt deren Anschlussbelegung an.

Schaltungseditor und Bauteilekatalog



Bauteile zur Schaltung hinzufügen
Aktivieren Sie das Bauteilauswahlfenster über den Menüpunkt *OPTIONEN->BAUTEILAUSWAHL* oder klicken Sie auf den entsprechenden Toolbarschalter.

Im Bauteilauswahlfenster erscheinen die zur Verfügung stehenden Bauteile nach Gruppen sortiert. Wählen Sie zunächst eine Bauteilgruppe wie z.B. Gatter. In der Bauteilliste erscheint daraufhin eine Auswahl von Gattern aus denen Sie das gewünschte Gatter durch einen Doppelklick auswählen können. Jeder Doppelklick fügt ein neues Bauteil in Ihre Schaltung ein. Neue Bauteile werden immer oben links im Schaltungs- bzw. Frontplatteneditor eingefügt.

Handelt es sich um ein Bauteil aus der Gruppe ' Bedienungselemente', so wird zusätzlich das entsprechende Bedienungselement in die Frontplatte eingefügt. Die Bezeichnung im Schaltungseditor wird automatisch vergeben. Das Bauteil der Frontplatte erhält die gleiche Bezeichnung, um eine eindeutige Zuordnung zu ermöglichen.

Bauteile verschieben
Wählen Sie den Menüpunkt *BEARBEITEN ->VERSCHIEBEN* oder betätigen Sie den entsprechenden Toolbarschalter.

Klicken Sie mit der Maus auf ein Bauteil und verschieben Sie es an die gewünschte Stelle, während Sie die linke Maustaste gedrückt halten. An der gewünschten Position lassen Sie das Bauteil los. Auf die gleiche Weise verfahren Sie mit den Bedienungselementen der Frontplatte.
Bitte beachten Sie, dass Sie Bauteile nicht auf Drähten oder anderen Bauteilen ablegen können. In diesem Fall springt das Bauteil wieder an seine ursprüngliche Position zurück. Die Lage der Anschlüsse von Gattern und kleinen Bauteilen lassen sich im Verschieben-Modus mit einem Klick mit der rechten Maustaste verändern. Jeder Klick dreht das Bauteil um 90 Grad.

Bauteile verbinden
Wählen Sie den Menüpunkt *BEARBEITEN ->VERBINDEN* oder betätigen Sie den entsprechenden Toolbarschalter. Der Mauszeiger im Schaltungseditor besteht aus einem Fadenkreuz. Das Ziehen von Drähten erfolgt stets in senkrechter oder waagerechter Richtung!

Betätigen Sie die linke Maustaste an der Stelle, an der der Draht beginnen soll. Bewegen Sie das Fadenkreuz zum nächsten Eckpunkt und betätigen Sie erneut die linke Maustaste. Damit ist das erste Drahtstück entstanden. Von dieser Stelle aus können Sie nun den Draht in eine andere Richtung fortsetzen. Bewegen Sie das Fadenkreuz auf die nächste Position und klicken Sie erneut. Um abzusetzen und an einer anderen Stelle weiterzumachen, betätigen Sie die rechte Maustaste und beginnen erneut wie zuvor beschrieben. Um das Drahtziehen zu beenden, betätigen Sie ebenfalls die rechte Maustaste.

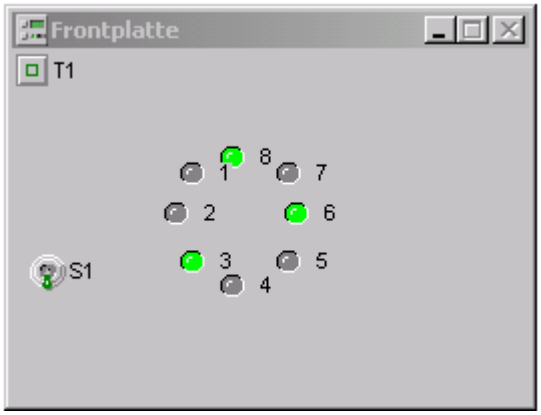
Treffen Sie beim Drahtziehen auf den Anschluss eines Bauteils oder auf einen anderen Draht, so erscheint im Fadenkreuz ein kleines Quadrat. Dies ist das Zeichen dafür, dass Sie die Verbindung zu diesem Bauteil bzw. Draht herstellen können. Wenn Drähte miteinander verbunden sind, so werden die 'Punkte' für Abzweigungen automatisch gesetzt.

Bitte beachten Sie, dass Sie Drähte nicht DURCH, ÜBER oder UNTER Bauteile zeichnen können, und dass sich Drähte nicht überlappen, jedoch kreuzen können.

Bauteile und Drähte löschen
Wählen Sie den Menüpunkt *BEARBEITEN ->LÖSCHEN* oder betätigen Sie den entsprechenden Toolbarschalter.
Um einzelne Elemente zu löschen, bewegen Sie das Zielkreuz auf das zu löschende Bauteil oder Drahtstück und drücken Sie dann die linke Maustaste.
Zum Löschen von mehreren nebeneinander liegenden Elementen, fahren Sie mit dem Zielkreuz, bei gedrückter linker Maustaste, über die zu löschenden Schaltungsteile.
Handelt es sich bei einem gelöschten Bauteil um ein Bedienungselement, so wird dieses auch aus der Frontplatte gelöscht.
Das Löschen von Bauteilen muss zusätzlich bestätigt werden.

Schaltung verschieben
Wählen Sie den Menüpunkt *BEARBEITEN->SCHALTUNG VERSCHIEBEN* .
Es erscheint ein Fenster in dem die Ausmaße Ihrer Schaltung auf dem Board maßstabsgetreu wiedergegeben sind. In diesem Fenster können Sie nun Ihre gesamte Schaltung beliebig auf dem Board platzieren. Wählen Sie OK, um das Verschieben zu beenden.

Der Frontplatteneditor



Bedienungselemente zur Frontplatte hinzufügen
Frontplattenelemente werden durch den Schaltungseditor verwaltet. Fügen Sie der Schaltung ein Bauteil aus der Gruppe 'Bedienungselemente' hinzu, so erhalten Sie dieses Bauteil automatisch auch auf der Frontplatte. Die Bezeichnung des Bedienungselementes wird mit der Bezeichnung des entsprechenden Bauteils im Schaltungseditor vorbelegt.

Bedienungselemente auf der Frontplatte anordnen
Bewegen Sie den Mauszeiger auf ein Bedienungselement, drücken Sie die linke Maustaste und schieben Sie es, bei gedrückter linker Maustaste, an die gewünschte Stelle.
Sie können im Menü OPTIONEN eine Rasterung für die Frontplatte ein- bzw. ausschalten. Bei eingeschaltetem Raster ist es leichter möglich, die einzelnen Elemente auszurichten.

Bedienungselemente von der Frontplatte löschen
Löschen Sie das entsprechende Bauteil aus der Schaltung, so wird es auch von der Frontplatte entfernt. Die Bezeichnung für die übrigen Bauteile der Schaltung werden im Schaltungseditor neu vergeben. Die gewählte Bezeichnung auf der Frontplatte bleibt aber erhalten.

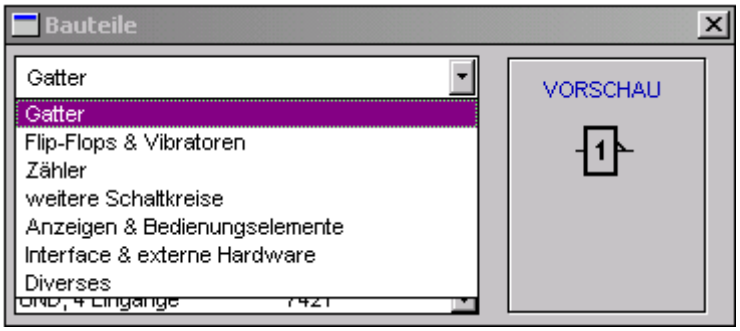
Beschriftung von Frontplattenbauteilen ändern
Bewegen Sie den Mauszeiger auf das gewünschte Bauteil und betätigen Sie die rechte Maustaste. Es erscheint eine Dialogbox, in der Sie die Beschriftung für das Frontplattenbauteil ändern können.
Die Bauteilbezeichnung stellt den Bezug zum Schaltungseditor her.

Bauteile

DigiTrace stellt Ihnen eine große Auswahl an digitalen Bauteilen zur Verfügung. Die Auswahl erfolgt im Bauteilauswahlfenster.
Konfigurierbare Bauteile wie z.B. der Taktgenerator oder der Lautsprecher, können Sie mit einem Doppelklick auf das entsprechende Bauteil konfigurieren.
Einige einfache Bauteile wie Gatter, Schalter, Taster, Taktgeneratoren u. ä. können mit der rechten Maustaste um 90° gedreht werden.

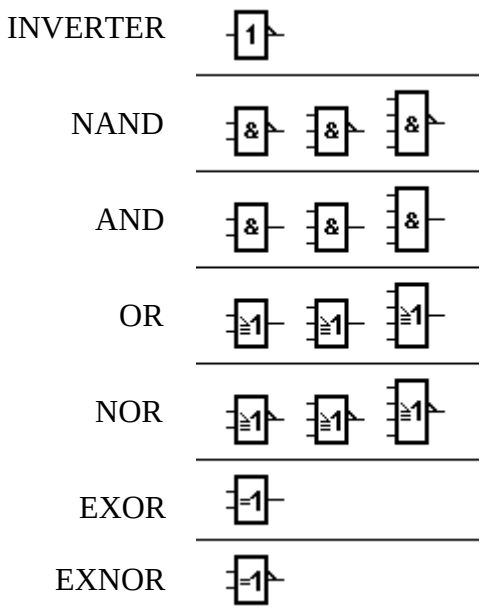
Die Bauteile sind im Bauteilauswahlfenster nach folgenden Gruppen gegliedert:

- **Gatter**
- **Flip-Flops & Vibratoren**
- **Zähler**
- **weitere Schaltkreise**
- **Anzeigen & Bedienungselemente**
- **Interface & externe Hardware**
- **Diverses**



Gatter

Die Gruppe Gatter enthält folgende Bauteile:



Bis auf die EXOR und EXNOR ist jedes Gatter mit 2, 3 oder 4 Eingängen vorhanden.

(*INVERTER = NOT)

Flip-Flops & Vibratoren

Taktgenerator

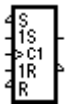
Es gibt wohl kaum eine Digitalschaltung, die ohne Taktsignal auskommt. Hierfür gibt es Taktgeneratoren, deren Pulsbreiten in Vielfachen von 50 ms einstellbar sind. Ein Doppelklick auf den Taktgenerator im Schaltplan öffnet die entsprechende Dialogbox.
Da die Taktimpulse aus den Windows-Timern gewonnen werden, kann man von diesen aber keine besonders hohe Genauigkeit erwarten. Dies ist für die Anwendung in der Simulation aber auch in aller Regel nicht von entscheidender Bedeutung. Allerdings sind schnellere Taktraten manchmal wünschenswert. In diesem Fall kann man den Ausgang eines Inverters auf seinen Eingang zurückkoppeln. Die dabei entstehende Schwingung entspricht der halben Simulationsfrequenz und kann gut als schneller Takt genutzt werden. Man nimmt dann allerdings in Kauf, dass die Taktrate von der Rechenleistung des PCs abhängt. Alternativ besteht noch die Möglichkeit, Taktsignale von externer Hardware einzuspeisen.

RS-Kippstufe, ungetaktet

Die ungetaktete RS-Kippstufe verfügt über zwei Eingänge zum Setzen (Set) und Rücksetzen (Reset) des Ausgangs Q. Der Ausgang /Q ist der invertierte Q-Ausgang. Der Ausgangszustand für R=1 und S=1 (gleichzeitiges Setzen und Rücksetzen) ist logisch nicht definiert und sollte vermieden werden.

RS-Latch, zustandsgesteuert

Dieses Latch unterscheidet sich von der ungetakteten RS-Kippstufe durch einen weiteren Eingang C (Clock). Die Eingänge der ursprünglichen RS-Kippstufe gehen aus der AND-Verknüpfung der SET- und RESET- Eingänge mit dem Clock-Eingang hervor. Für C=1 verhält sich dieses Latch wie eine RS-Kippstufe ohne Takt und für C=0 bleiben alle Änderungen an den Eingängen unberücksichtigt.



RS-Flip-Flop, flankengesteuert

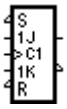
Dieses Flip-Flop besteht aus zwei hintereinander geschalteten RS-Latches, von denen bei C1=1 das erste Latch (Master) und bei C1=0 das zweite Latch (Slave) freigegeben wird. Bei aktivem Clock kann also mit dem Master-Latch die gewünschte Funktion (1S=SET oder 1R=RESET) gewählt werden, die dann mit der fallenden Taktflanke durch das Slave-Latch ausgeführt wird.

D-Latch, zustandsgesteuert

Das D-Latch gibt die Information an seinem D-Eingang (DATA) bei aktivem Clock (C=1) an den Ausgang weiter. Für C=0 bleibt der Ausgang unverändert, d.h. das Latch speichert. Das D-Latch geht aus dem RS-Latch hervor, wenn man dessen SET-Eingang als DATA-Eingang betrachtet und den RESET-Eingang mit dem invertierten SET-Signal beschaltet (RESET= NOT SET).

D-Flip-Flop, flankengesteuert

Ein flankengesteuertes D-Flip-Flop erhält man, wenn man zwei zustandsgesteuerte D-Latches in Reihe schaltet und den Takt des Master-Latch invertiert. Während CLK=0 wird der Zustand des DATA-Signals in das Master-Latch übernommen. Mit CLK=1 wird das Master-Latch gesperrt und es gibt seinen gespeicherten Wert an das nun freigegebene Slave-Latch weiter. Zusätzlich verfügt das flankengesteuerte D-Flip-Flop über zwei asynchrone Eingänge /S und /R zum taktunabhängigen Setzen der Ausgänge.



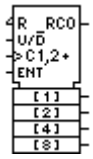
JK-Flip-Flop

Das JK-Flip-Flop ist wohl der universellste Vertreter seiner Art, da seine Ausgänge auf die Eingänge zurückgeführt sind und so die gewünschte Funktion über die Eingänge J und K bestimmt werden kann.

J=0 ,K=0	Q=Q-1	Speichern
J=1, K=0	Q=1	Setzen
J=0, K=1	Q=0	Rücksetzen
J=1, K=1	Q=/Q	Toggle (Q ändert sich bei jedem Takt)

Bei der steigenden Taktflanke wird die gewünschte Funktion in das Master-Flip-Flop eingelesen die bei fallender Taktflanke vom Slave-Flip-Flop ausgeführt wird.
Für die taktunabhänige Betätigung der Ausgänge stehen die asynchronen Eingänge /S und /R zur Verfügung.

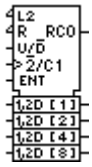
Zähler



Dual-Zähler, 4-bit

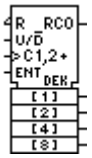
Ein synchroner Dualzähler besteht prinzipiell aus hintereinander geschalteten Toggle-Flip-Flops. Alle Flip-Flops erhalten einen gemeinsamen Taktimpuls. Das Umkippen (Togglen) einer Zählstufe erfolgt synchron mit der fallenden Taktflanke.

Der Dualzähler verfügt außerdem über die Möglichkeit die Zählrichtung umzuschalten (UP = /DOWN). Der Eingang ENT gibt den Takteingang frei oder sperrt diesen.
Das Setzen des Übertragsflags RCO erfolgt in Abhängigkeit von der Zählrichtung. Für U/D=1 (aufwärts) erscheint RCO bei einem Zählerstand von 16 (&h0F), für U/D=0 (abwärts) bei einem Zählerstand von 0 (&h00).
Mit dem Übertragsflag RCO können mehrere Zähler kaskadiert (hintereinander geschaltet) werden. Mit dem Reset-Eingang /R kann man den Zählerstand &h00 (U/D=1) bzw. &h0F (U/D=0) wiederherstellen. Der Reset-Eingang arbeitet taktunabhängig.



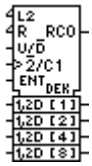
Dual-Zähler, 4-bit, ladbar

Der Unterschied zum einfachen Dualzähler besteht in den vier Ladeeingängen dieses Zählers, mit denen der Zähler mit einem bestimmten Zählerstand vorbelegt werden kann. Das Laden erfolgt synchron mit der fallenden Taktflanke, wenn der Eingang /L2=0 ist.



Dekaden-Zähler

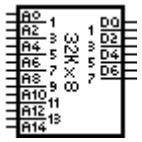
Die Dekadenzähler verhalten sich identisch zu den Dualzählern, mit dem Unterschied, dass ihr Zähler-Endstand 9 (&h09) ist. Diese Zähler eignen sich damit besonders zum Aufbau dezimal-basierter Schaltungen, wie Anzeigen im 10-er Zahlensystem, als Teiler durch 10, usw.



Dekaden-Zähler, ladbar

Der Unterschied zum einfachen Dekaden-Zähler besteht in den vier Ladeeingängen dieses Zählers, mit denen der Zähler mit einem bestimmten Zählerstand vorbelegt werden kann. Das Laden erfolgt synchron mit der fallenden Taktflanke, wenn der Eingang /L2=0 ist.

Weitere Schaltkreise



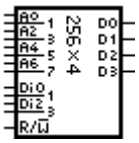
EPROM

Das EPROM erlaubt die dauerhafte Speicherung von 32kByte Daten mit 8-Bit Breite. Man kann sich das EPROM als eine Tabelle vorstellen, in der zeilenweise 8-Bit-Werte (also Bytes) eingetragen sind. Die Zeilennummern - beginnend mit 0 - stellen die Adressen dar, an denen ein Byte abgelegt ist. Legt man eine Adresse (Zeilennummer) in binärer Form an die Adresseingänge des EPROMs an, so wird der an dieser Adresse abgelegte Bytewert - ebenfalls in binärer Form - über die Datenausgänge des EPROMs ausgegeben.
Durch einen Doppelklick auf das EPROM öffnen sie ein Fenster in dem Sie den Inhalt betrachten und ändern können.

Sie können den EPROM-Inhalt auch aus einer Text-Datei laden. Die Text-Datei sollte folgenden Aufbau haben:

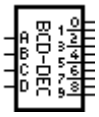
```
; Semikolon leitet einen Kommentar ein ( der Rest der Zeile wird ignoriert )
06 05 F8 37    ; der Inhalt wird in Form von HEX-Zahlen angegeben
03,07,69,56    ; zwischen den HEX-Zahlen können beliebige Trennzeichen stehen
$56, $FF       ; es können beliebig viele Trennzeichen verwendet werden
&FFFF , $77    ; HEX-Zahlen >$FF werden ignoriert! ( hier $FFFF )
```

es können beliebig viele Zahlen in einer Zeile stehen (max. Zeilenlänge 255 Zeichen !!!)
Alle gefunden Bytes (HEX-Zahlen) werden ab Adresse 0 in das EPROM geladen.



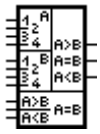
RAM 256 x 4

Das RAM dient der temporären Speicherung von Daten. Über die Eingänge Di0..Di2 können 4 Datenbits parallel in das RAM eingelesen werden. Die Speicherung erfolgt an die Adresse, die über die Adressleitungen A0.A6 angelegt wird. Es stehen die 256 Adressen &h0000..&h00FF zur Verfügung. Solange das Signal R/W=0 (schreiben) ist, sind die Dateneingänge aktiv. Bei R/W=1 (lesen) werden die Daten an der aktuellen Adresse über die Ausgänge D0..D3 ausgegeben.



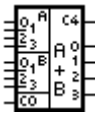
BCD nach Dezimal Decoder

Der BCD-Decoder erhält über die Eingänge eine Dezimalzahl, die im BCD-Format kodiert ist und setzt die dazugehörige Ausgangsleitung (0..9). Legt man an die Eingänge eine höhere (Dual-)Zahl von 10..16 - was praktisch möglich ist - so ist das Ausgangssignal undefiniert.



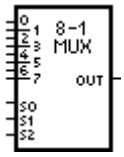
4-bit Vergleicher

Mit einem Vergleicher können zwei Dualzahlen A und B (jeweils 4 Bit) miteinander in Relation gesetzt werden. Seine Ausgänge zeigen die Zustände A= B, A>B und A<B an. Um auch größer Zahlen (z.B. Bytes=8 Bit) miteinander vergleichen zu können, verfügt jeder Vergleicher über Kaskadeneingänge, die mit den Ausgängen des jeweils niederwertigeren Vergleichers verbunden werden. Jeder Vergleicher übernimmt ein Halbbyte (Nibble) des gesamten Zahlenwertes. Die nicht benötigten Kaskadeneingänge des niederwertigsten Vergleichers müssen A=B signalisiert bekommen. Hierzu legt man die Eingänge A<B und A>B auf Masse.



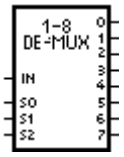
4-bit-Volladdierer

Mit einem Addierer lassen sich zwei Dualzahlen summieren. Volladdierer zeichnen sich gegenüber sog. Halbaddierern durch einen Übertragseingang C0 und Übertragsausgang C4 aus, über die sich diese Addierer kaskadieren lassen. Dazu verbindet man jeweils den Ausgang C4 mit dem Eingang C0 des nächsten, höherwertigeren Addierers. Der nicht benötigte C0-Eingang des niederwertigsten Addierers muss mit 0 beschaltet werden, da andernfalls eine 1 in der untersten Stufe addiert wird.



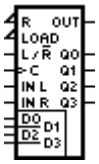
8-Kanal-Multiplexer

Der Multiplexer kann als steuerbarer Umschalter betrachtet werden, der einen von acht möglichen Eingangskanälen auswählt und die Daten dieses Kanals an den Ausgang weitergibt. Welcher Kanal selektiert wird, wird über die Eingänge S0...S2 bestimmt.



8-Kanal-Demultiplexer

Der Demultiplexer ist das Gegenstück zum Multiplexer. Hier bestimmen die Eingänge S0..S2 an welchen Ausgang 0..7 die Daten des Eingangs weitergegeben werden.



4-bit-Schieberegister

Schieberegister sind hintereinander geschaltete Flip-Flops, die ihre Information schrittweise an ihren Nachfolger weitergeben. Am Anfang der Reihe können Daten in das erste Flip-Flop geschrieben werden, die dann, nachdem Sie die Reihe der Flip-Flops durchlaufen haben, zeitverzögert am Ausgang erscheinen.
Die Schieberichtung des Schieberegisters kann umgekehrt werden (L/R). Während beim Linksschieben (L/R=1) der Eingang INL aktiv ist, der seine Daten an das erste Flip-Flop (Q0) weitergibt, werden beim Rechtsschieben (L/R=0) die Daten vom Eingang INR in das höchstwertige Flip-Flop (Q3) übernommen. Der Ausgang OUT entspricht je nach Schieberichtung dem Ausgang Q3 (Linksschieben) bzw. dem Ausgang Q0 (Rechtsschieben).

Das Schieberegister kann außerdem parallel geladen werden, d.h. die Daten an den Eingängen D0..D3 werden mit /LOAD=0 und einer fallenden Taktflanke direkt in die zugehörigen Flip-Flop-Stufen übernommen.

Das Rücksetzen aller Stufen über den /RESET - Eingang ist taktunabhängig (asynchron).

Anzeigen & Bedienungselemente

Leuchtdiode

Leuchtdioden verfügen lediglich über einen Signaleingang. Eine ‘1’ am Eingang lässt die Leuchtdiode aufleuchten.

Schalter

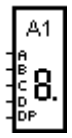
Schalter werden als Kippschalter dargestellt. Sie verfügen über einen Ausgang der unmittelbar an Logikbauteile angeschlossen werden kann. Welche Schalterstellung einer ‘1’ am Ausgang entspricht (Polarität), kann über einen Dialog eingestellt werden. Der Dialog erscheint, wenn man im Bearbeitungsmodus einen Doppelklick auf das Bauteil im Schaltplan ausführt.

Taster

Taster erscheinen als Drucktaster auf der Frontplatte. Sie verfügen über einen Ausgang der unmittelbar an Logikbauteile angeschlossen werden kann. Welche Tasterzustand einer ‘1’ am Ausgang entspricht (Polarität), kann über einen Dialog eingestellt werden. Der Dialog erscheint, wenn man im Bearbeitungsmodus einen Doppelklick auf das Bauteil im Schaltplan ausführt.

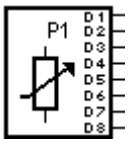
Hexadezimal-Eingabe

Die Hex-Eingabe gibt den eingestellten Wert (&h00..&h0F) als Dualzahl an den Ausgängen ab. Die Einstellung erfolgt bei laufender Simulation mit dem integrierten Taster. Die Hex-Eingabe kann mit der linken (Erhöhen) und rechten (Verringern) Maustaste betätigt werden.



7-Segment-Anzeige

Während in der Realität BCD-7-Segment-Decoder nötig sind, um die einzelnen LED-Segmente anzusteuern, besitzt dieses Bauteil bereits die Eingänge A..D, an die man direkt eine 4-Bit-Dualzahl &h00..&h0F anlegen kann, um die entsprechende Anzeige zu erhalten. Der Decoder ist bereits integriert. Der Eingang DP steuert in gewohnter Weise den Dezimalpunkt an.



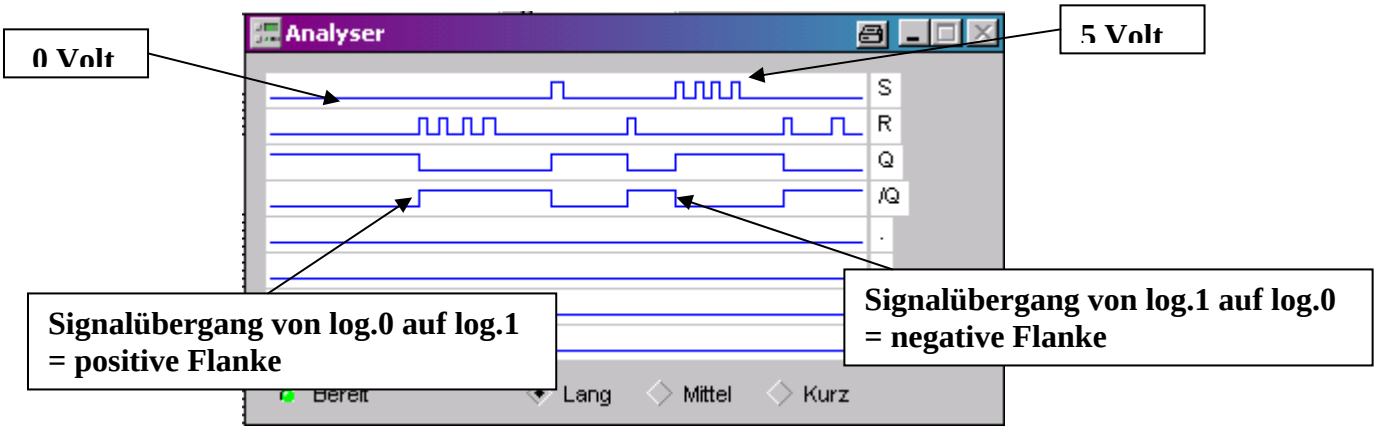
Potentiometer

Mit dem Potentiometer kann ein Wert von &h00..&hFF eingestellt werden, der über die Ausgänge D1..D8 als Dualzahl abgegeben wird. Das Einstellen des Potentiometers kann durch einzelne Mausklicks mit rechter und linker Maustaste oder durch ‘Ziehen’ eingestellt werden.

8-Kanal-Logikanalyser

Ein besonders leistungsfähiges Bauteil stellt der Logikanalyser dar. Mit ihm lassen sich bis zu acht Signalverläufe über einen gewissen Zeitraum aufzeichnen und darstellen. Er dient somit in erster Linie als Hilfsmittel bei der Schaltungsentwicklung mit dem man z.B. die Funktionsweise einer Schaltung überprüfen kann. Fügt man den Logikanalyser in eine Schaltung ein, so erscheint auf dem Bildschirm ein neues Fenster, das die Signalverläufe und den Gerätestatus anzeigt. In jedem Projekt kann man einen Analyser verwenden.
Die Aufnahmedauer des Analysers kann in 3 Stufen umgeschaltet werden. Die Aufzeichnung wird über den TRIGGER-Eingang ausgelöst. Mit einer 1 am Trigger-Eingang startet die Aufnahme und endet nach der eingestellten Aufnahmedauer. Danach werden die aufgenommen Signalverläufe der acht Eingangskanäle angezeigt. Solange am Trigger-Eingang eine 1 erscheint, wird unmittelbar der nächste Aufnahmezyklus ausgelöst, also auch bei offenem Trigger-Eingang. Ist der Trigger-Eingang hingegen 0, so geht der Analyser nach der Aufnahme in den Wartezustand (Bereit), bis er erneut ausgelöst wird.

Das Bild zeigt den Signalverlauf einer RS-Kippstufe, aufgenommen vom Logikanalyser.



Die Tiefen der Signale werden von 0Volt, die Höhen von 5Volt erzeugt. Aus diesem Signal ist ersichtlich, wie die Ausgänge Q und /Q von den Eingängen S und R abhängig sind. Die 5Volt-Signale der Eingänge stammen von Tastern. Man sieht, dass das Drücken der Taste R den Ausgang Q auf 0Volt zieht. Ein erneutes drücken von R führt zu keiner Änderung. Erst durch Drücken der Taste S geht der Ausgang Q wieder auf 5Volt und zwar solange, bis erneut R gedrückt wird. Man kann also sagen, mit S setze ich Q auf 5Volt (Set) und mit R setze ich Q auf 0Volt zurück. (Reset)

Protokoll-Meldung

Protokolle eignen sich z.B. um Benutzereingaben aufzuzeichnen oder Schaltungszustände zu überwachen und Veränderungen als Klartextmeldung auszugeben. Dabei kann zwischen verschiedenen Ausgabegeräten gewählt werden. Mögliche Ausgabegeräte sind Drucker, Dateien oder eines von zehn möglichen Bildschirm-Fenstern. Um eine Textausgabe auf einem dieser Ausgabegeräte zu erzeugen, benutzt man eine sog. Protokoll-Meldung. Eine Protokoll-Meldung besitzt einen Eingang PRINT, an den das zu überwachende Signal angeschlossen wird.
Mit einem Doppelklick auf das Bauteil öffnet sich ein Konfigurationsdialog.
Im Konfigurationsdialog wird der gewünschte Meldungstext eingegeben, der wahlweise bei steigender oder fallender Flanke am PRINT-Eingang ausgegeben wird. Standardmäßig wird die Ausgabe zunächst unterdrückt. Erst wenn ein Ausgabegerät gewählt wird, erhält man eine Ausgabe. Der Meldungstext wird im ASCII-Format zeilenweise in ein Fenster, die gewählte Datei oder einen auf einen Drucker ausgegeben.

Es können mehrere Protokoll-Meldungen definiert werden, die auf dasselbe Ausgabegerät zugreifen.

Mit dem /RESET-Eingang kann das jeweilige Ausgabegerät zurückgesetzt werden. Bei Fenstern und Dateien wird so deren bisheriger Inhalt gelöscht, während der Drucker einen Seitenvorschub ausführt.

Lautsprecher / Klangwiedergabe

Wer eine Soundkarte mit Windows-Treiber besitzt, kann mit diesem Bauteil Klänge (Wave-Dateien; *.WAV) wiedergeben. Mit einem Doppelklick auf das Bauteil wird die Verbindung zur gewünschten Klangdatei hergestellt. Die Wiedergabe des Klangs wird mit der steigenden Flanke am Eingang des Bauteils ausgelöst.

Interface & externe Hardware

Digital-Ausgang

Mit diesem Bauteil lassen sich die Ausgänge Ihrer LPT-Ports bzw. Schnittstellenkarten ansteuern. Mit einem Doppelklick auf das Schaltsymbol öffnen Sie eine Dialogbox, in der Sie den gewünschten Anschluss einstellen.

Wählen Sie zunächst das Gerät, über das die Ausgabe erfolgen soll.
Stellen Sie dann die Basisadresse für dieses Gerät ein. Das Ändern der Basisadresse für ein Gerät passt automatisch alle Ein- und Ausgänge, die dieses Gerät benutzen, an die neue Basisadresse an. So ist es leicht möglich, eine Schaltung für andere Portadressen zu konfigurieren.
Schließlich wählen Sie den gewünschten Pin, über den Sie das Signal nach außen führen möchten.
Im Feld Kommentar können Sie den Pin beschriften.

Bitte beachten Sie, dass Sie einen Ausgangspin nur einmal in einer Schaltung verwenden können. Andernfalls werden Sie auf die Doppelbelegung hingewiesen.

Digital-Eingang

Mit diesem Bauteil können Sie 'echte' Signale über Ihre Schnittstellen in die Simulation einspeisen. Mit einem Doppelklick auf das Schaltsymbol öffnen Sie eine Dialogbox, in der Sie den gewünschten Anschluss einstellen.
Wählen Sie zunächst das Gerät, über das die Eingabe erfolgen soll.
Stellen Sie dann die Basisadresse für dieses Gerät ein. Das Ändern der Basisadresse für ein Gerät passt automatisch alle Ein- und Ausgänge, die dieses Gerät benutzen, an die neue Basisadresse an. So ist es leicht möglich, eine Schaltung für andere Portadressen zu konfigurieren.
Schließlich wählen Sie den gewünschten Pin, über den Sie das Signal einspeisen möchten.
Im Feld Kommentar können Sie den Pin 'beschriften'.

Diverses

Masse

Um die Eingänge von Bauteilen mit definierten Zuständen zu versehen, können diese mit Masse (logisch 0) oder Plus (logisch 1) beschaltet werden. Für die Spannungsversorgung der Bauteile braucht man nicht zu sorgen.

Plus

Um die Eingänge von Bauteilen mit definierten Zuständen zu versehen, können diese mit Masse (logisch 0) oder Plus (logisch 1) beschaltet werden. Für die Spannungsversorgung der Bauteile braucht man nicht zu sorgen.

Beschriftung **Beschriftung**

Dieses 'Bauteil' dient dazu einen Text in Ihren Schaltplan einzufügen. Mit einem Doppelklick auf das Schaltsymbol öffnen Sie eine Dialogbox, in der Sie den Beschriftungstext eingeben können.

TTL-Katalog

Über den Menüpunkt OPTIONEN->TTL-KATALOG oder über den entsprechenden Toolbar-Schalter, kann eine Übersicht über oft benötigte Standard-TTL-Bausteine und deren Anschlussbelegung eingeblendet werden. Diese Programmfunktion ist besonders bei der hardwaremäßigen Realisation von Schaltungen hilfreich. Aus dem TTL-Katalog können jedoch keine Bauteile in die simulierte Schaltung eingefügt werden. Sie sollen bei der Auswahl geeigneter Bauelemente zur hardwaremäßigen Realisierung dienen.

Drucken

Wählen Sie aus dem Menü DATEI zunächst den Menüpunkt DRUCKER EINRICHTEN aus. Es erscheint eine Dialogbox, in der Sie den gewünschten Drucker auswählen können. Über den SETUP-Knopf können Sie Ihren gewählten Drucker konfigurieren. Sie können hier z.B. die Druckerauflösung und das Druckformat (Hoch- oder Querformat) einstellen. Die Einstellungsmöglichkeiten hängen von Ihrem Drucker, bzw. vom entsprechenden Treiber ab.

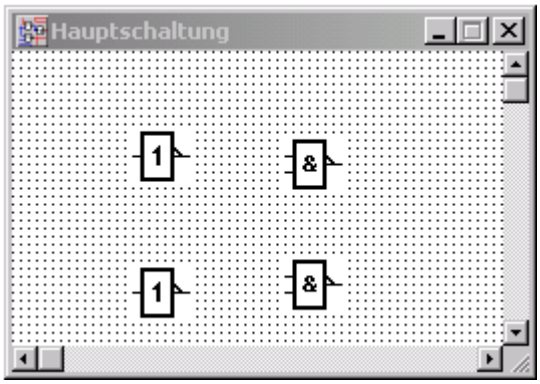
Wählen Sie dann aus dem Menü DATEI den Menüpunkt SCHALTUNG DRUCKEN aus. Es erscheint eine Dialogbox zum Skalieren der Druckerausgabe. Sie können die Skalierung direkt in Prozent angeben. Dabei bedeutet eine Einstellung von 100 %, dass die Schaltung 1:1 auf den Drucker ausgegeben wird. Die Größe dieser Druckausgabe hängt dabei von der gewählten Druckerauflösung ab. Erhöhen Sie die prozentuale Skalierung, um den Ausdruck zu vergrößern.
Mit dem Schaltfeld 'auf Seite anpassen' haben Sie die Möglichkeit, die Druckausgabe automatisch so zu skalieren, dass das gesamte Schaltungsboard in maximaler Größe auf dem Papier ausgegeben wird. Ist die Boardgröße an die Schaltung angepasst, erhalten Sie Ihre gesamte Schaltung auf dem Papier in maximaler Größe.

Übung: Schaltung mit DigiTrace erstellen

Ungetaktete RS-Kippstufe aus NAND-Gattern aufgebaut

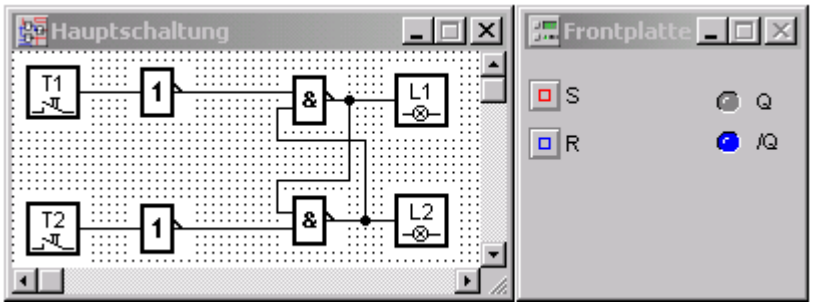
Starten Sie DigiTrace und ziehen Sie den Schaltungseditor, die Frontplatte und den Bauteilekatalog auf die gewünschte Größe und Position.

Mit einem Klick auf den Bauteile-Button  öffnen Sie den Bauteilekatalog.
Dort wählen Sie die Bauteilgruppe „Gatter“ und fügen jeweils mit Doppelklick zwei „**INVERTER**“ (=NOT) und zwei “**NAND, 2 Eingänge**“ dem Schaltungseditor hinzu. Diese sind nun, übereinander liegend, in der linken oberen Ecke positioniert. Während dem Hinzufügen von Bauteilen befinden Sie sich automatisch im „Verschieben-Modus“. Ziehen Sie die bisher eingefügten Bauteile mit gedrückter Maustaste in die gewünschte Position.



Wählen Sie nun im Bauteilekatalog die Bauteilgruppe „Anzeigen & Bedienungselemente“ und fügen Sie zwei „**Taster**“ und zwei „**Leuchtdioden**“ der Schaltung hinzu. Anzeigen und Bedienungselemente werden automatisch zusätzlich der Frontplatte hinzugefügt.

Nachdem Sie diese Bauteile an eine geeignete Position gezogen haben, schalten Sie mit einem Mausklick auf den ersten Toolbar-Button  (der Lötkolben), der das Programm in den „Verbinden-Modus“ setzt. Der Mauszeiger verwandelt sich nun über dem Schaltungseditor in ein blaues Fadenkreuz. Mit einem Linksklick starten Sie eine Drahtverbindung, mit jedem weiteren Linksklick leiten Sie einen 90° Winkel ein, und mit einem Links-Rechtsklick schließen Sie die Verbindung ab. Um Bauteile miteinander zu verbinden, müssen Sie das Fadenkreuz über dem gewünschten Ein- oder Ausgang so positionieren, dass dort ein kleines schwarzes Quadrat erscheint. Verbinden Sie nun die Bauteile wie auf dem Bild unten und gewöhnen Sie sich am besten von Anfang die richtige Bennennung der Elemente auf der Frontplatte an.



Starten Sie die Simulation mit Klick auf den RUN-Button  .

Testen Sie Ihr Wissen

(*Alle Fragen beziehen sich auf die Simulationssoftware DigiTrace)

1.) Beschreiben Sie, wie man Bauteile miteinander verbindet.

2.) Wie ändert man die Beschriftung von Anzeigen und Bedienungselementen auf der Frontplatte?

3.) Sie wollen während einer Simulation den Stromverlauf anzeigen. Wie gehen Sie vor?

4.) Mit dem Logikanalyser kann man Schaltungen analysieren, indem man Signalverläufe aufzeichnet. Wie würden Sie dieses Bauteil an eine RS-Kippstufe anschließen, um ein aussagekräftiges Impulsdiagramm zu erhalten?

Addierer

Kapitel 6

Ein Addierer ist nichts anderes, als ein Schaltwerk aus Gattern, das zwei binäre Zahlen miteinander addiert. Es werden zwei Arten von Addierern unterschieden. Die einfachen Halbaddierer ohne Übertragseingang und die etwas komplexeren Volladdierer mit Übertragseingang.

In diesem Kapitel lernen wir, aus welchen Gattern ein Halbaddierer aufgebaut ist, wie man aus Halbaddierern einen Volladdierer baut und wie man aus 4-Bit Volladdierern einen n-Bit Volladdierer macht.

Halbaddierer

Um die Funktionsweise von Addierern zu verstehen, schauen wir uns das Bild 6.1 an. Es ist ein 1-Bit Halbaddierer, aufgebaut aus einem XOR zur Addition und einem AND, der den Übertrag erzeugt.

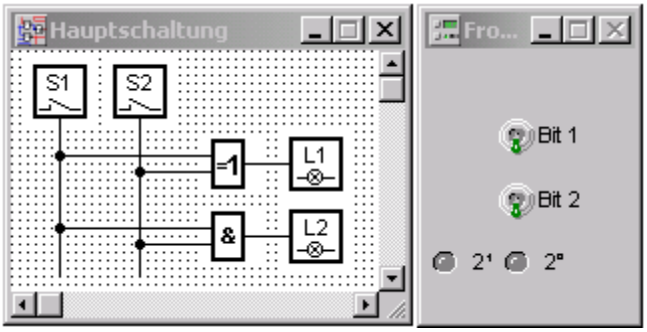


Bild 6.1 1-Bit-Halbaddierer

XOR

a	b	x
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

AND

a	b	x
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Die zu addierenden zwei Binärzahlen wurden an Schalter gelegt. Die Summe und der Übertrag sind hier mit LEDs dargestellt. Beide Bits und die Summe-LED (L1) haben die Stellenwertigkeit 2^0. Die Übertrags-LED (L2) auf der Frontplatte ist mit der Stellenwertigkeit 2^1 bezeichnet.

An den Wahrheitstabellen können wir nachvollziehen, wie diese Schaltung arbeitet. Das XOR-Gatter bildet die Summe, das AND-Gatter ist für den Übertrag zuständig. Zuerst erinnern wir uns aber noch mal, wie das mit der Addition binärer Zahlen war.

Addition binärer Zahlen

0 + 0 = 0

1 + 0 = 1

0 + 1 = 1

1 + 1 = 0 Übertrag 1

Wahrheitstabelle 1-Bit Halbaddierer

S1	S2	L1 (Summe)	L2 (Übertrag)
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

Volladdierer

Ein Volladdierer besitzt einen Übertragseingang. Den benötigen wir, um mehrere Addierer hintereinander zu schalten (kaskadieren) und so größere Zahlen miteinander addieren zu können. Ein Volladdierer lässt sich aus zwei Halbaddierern und einem OR aufbauen.

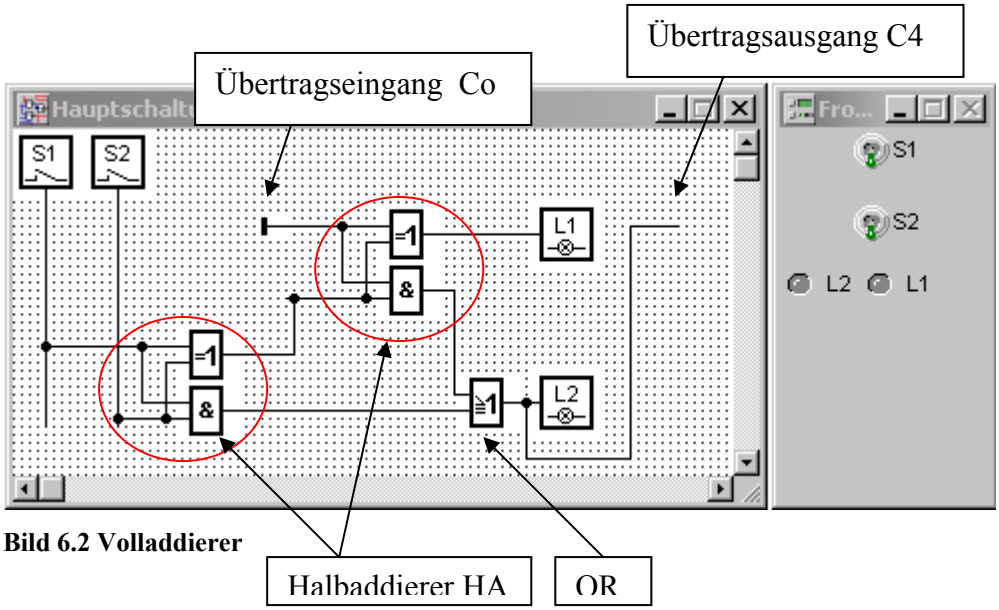
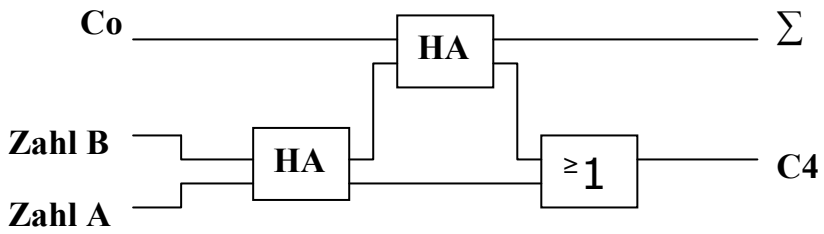


Bild 6.2 Volladdierer

Da diese Bausteine drei Eingänge besitzen (zwei Zahleneingänge und ein Übertragseingang), muss der erste Übertragseingang auf Masse gelegt werden. Ein offener Übertragseingang bedeutet Übertrag = 1. Das würde das Ergebnis verfälschen. Man könnte sich jetzt fragen, wieso man nicht einfach für die Addition der niederste Stellenwertigkeit einen Halbaddierer benutzt. Das könnte man schon machen, aber dann ist dieser Volladdierer beim Cascading (stufenweises hintereinander schalten) nur für die unteren Bits zu gebrauchen. Den eventuellen Übertrag eines davor geschalteten Addierers könnte er nicht weitergeben.

Vereinfacht dargestellt sieht der Volladdierer aus Bild 6.2 so aus



Übertragseingang = Co
Übertragsausgang = C4
Summe = ∑

Bild 6.2 zeigt einen 1-Bit Volladdierer. Durch Cascading kann man beliebig große Addierer bauen. Geläufige Addierer sind z.B. 4-Bit Volladdierer.

Mehrstufige Volladdierer

Um größere Zahlen zu summieren, können mehrere Volladdierer hintereinander geschaltet werden**. Dabei ist zu beachten, dass der Übertrag jeweils von $\sum_0$ an $\sum_1$ an $\sum_2$ an $\sum_3$ und dann erst an C4 weitergereicht wird.**
Wenn also zwei 2-Bit Zahlen mit einem 4-Bit Volladdierer summiert werden, liegt der Übertrag an $\sum_2$.
Schauen wir uns nun an einem Beispiel eine Kaskadierung mit 4-Bit-Volladdierern an. Wir verwenden für das unten aufgeführte Exempel das Schaltsymbol aus Bild 6.3 als Addierer.

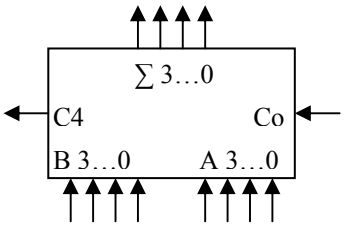
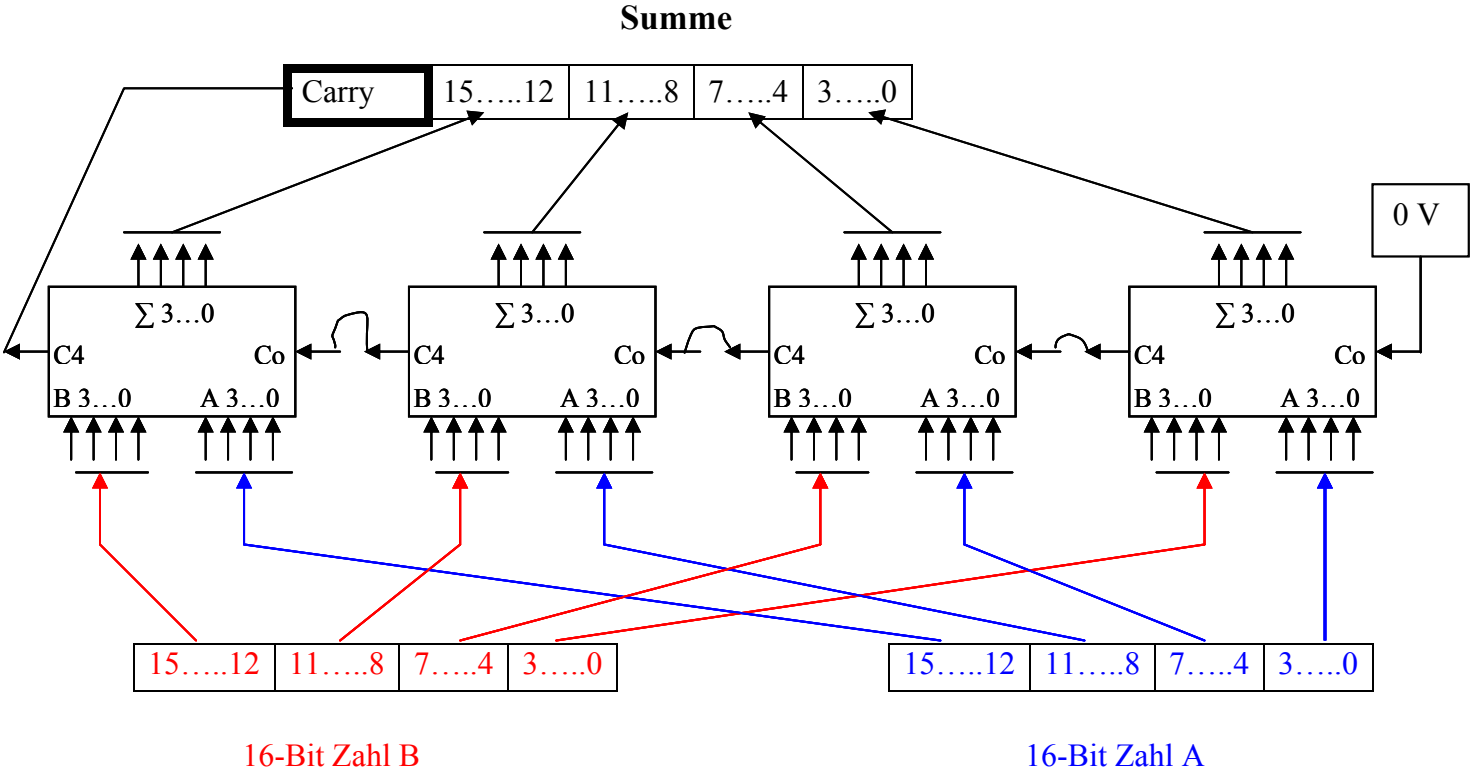


Bild 6.3 4-Bit Volladdierer

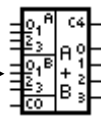
16-Bit Addierer aufgebaut aus 4-Bit Volladdierern

Es wird jeweils die 4-Bit Zahl A zur 4-Bit Zahl B summiert.
Das 4-Bit Ergebnis wird am Ausgang Summe (∑ 3…0) ausgegeben. Der Co des Addierers mit den untersten 4-Bit wird auf Masse gelegt. Die Übertragseingänge der stellenwertig höheren Addierer werden jeweils mit den Übertragsausgängen der vorigen Bausteine verbunden.



von links nach rechts wird eine Addition nach der anderen abgearbeitet.

Im DigiTrace-Bausteinkatalog ist ein 4-Bit Addierer verfügbar. Dieser kann zum modularen Aufbau solcher Schaltungen verwendet werden.

4-Bit Volladdierer in DigiTrace 

Vergleicher

Ein Vergleicher ist eine Schaltung, die zwei Zahlen miteinander vergleicht und das Ergebnis größer (>), kleiner (<), oder gleich (=) ausgibt.
Aus beispielsweise 1mal NAND-, 2mal AND-, und einem NOR-Gatter, lässt sich eine solche Schaltung für zwei 1-Bit Zahlen generieren.

Schauen wir uns dazu folgende Funktionstabelle an.

Funktionstabelle 1-Bit Vergleicher

A	B	A > B	A < B	A = B
0	0	**0**	**0**	**1**
0	1	**0**	**1**	**0**
1	0	**1**	**0**	**0**
1	1	**0**	**0**	**1**

Eine häufig eingesetzte Variante zur Realisierung der nebenstehenden Funktionstabelle ist in Bild 6.4 dargestellt.

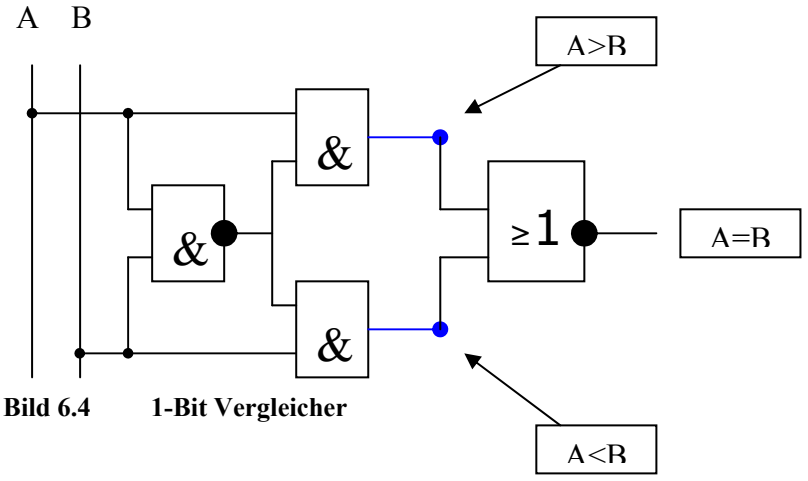


Bild 6.4 1-Bit Vergleicher

Um Schaltungen zu bauen, die größere Zahlen miteinander vergleichen, hat man zwei Möglichkeiten. Entweder man erstellt Funktionstabellen und die dazugehörenden Schaltungen für alle benötigten Größen, oder man verwendet für jedes zu vergleichende Bit einen 1-Bit Vergleicher und erweitert dann diese Module mit einer anderen Schaltung.

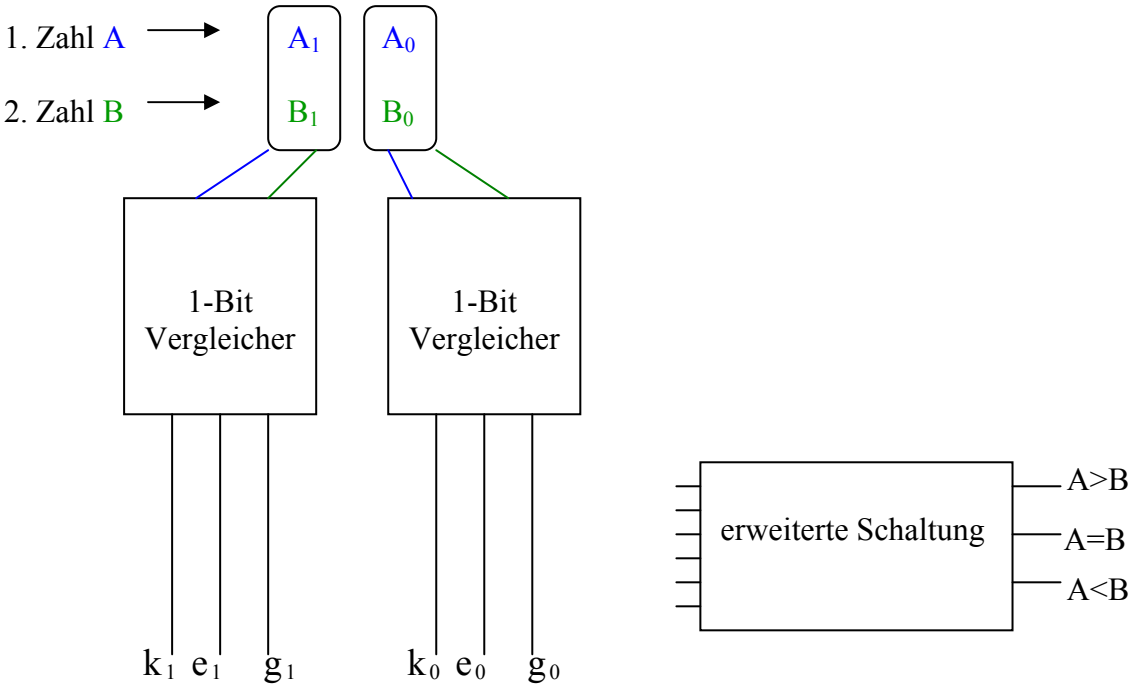
Die Funktionstabelle für einen 1-Bit Vergleicher ist 4 Zeilen groß. Für einen 2-Bit Vergleicher ist sie 16 Zeilen groß. Und wollen wir zwei Zahlen mit jeweils 3 Bit miteinander vergleichen, so benötigen wir immerhin schon eine Tabelle mit 64 Zeilen. Wir sehen also, dass man mit dieser Versionsmöglichkeit eine Vergleicherschaltung aufzubauen, sehr schnell an die Grenzen des Machbaren stößt.

Üblicherweise werden Vergleicher deshalb, und auch der besseren Erweiterungsfähigkeit wegen, modular aufgebaut. Schauen uns die Entwicklung eines modular aufgebauten Vergleicher an einem Beispiel an.

Modular aufgebauter Vergleicher.

Für einen Vergleich mit zwei 1-Bit Zahlen, kann ich das Schaltungsmodul aus Bild 6.4 direkt und ohne Erweiterungsschaltung einsetzen. Die Schaltung hat zwei Eingänge (zum Einlesen der Zahl A und der Zahl B) und drei Ausgänge (zum Zurückgeben der Ergebnisse A>B, A<B und A=B).
Will man nun zwei 2-Bit Zahlen miteinander vergleichen, so verwendet man diese Ausgänge als Eingänge der erweiterten Schaltung.

Als Beispiel vergleichen wir die Zahl A (z.B. 10_b) mit der Zahl B (z.B. 11_b).
Für die einzelnen Stellen verwenden wir einen Index.



(*die Ergebnisse der 1-Bit Vergleicher, hier mit **g** für **größer**, **e** für **gleich** und **k** für **kleiner**)

Uns geht es nun darum, wie die erweiterte Schaltung, die die Endergebnisse des Vergleichs bringen soll, zu erstellen und anzuschließen ist.

Da wir drei Endergebnisse erhalten sollen, stellen wir drei Gleichungen auf, die wir am einfachsten zunächst in Worte fassen.
Wir beginnen mit dem Vergleich immer an der höchstwertigen Stelle der Zahlen.

1. Gleichung für A>B

Die Zahl A ist größer als die Zahl B, wenn A_1 größer ist als B_1 oder, wenn A_1 gleich groß ist wie B_1 und A_0 größer ist als B_0.

Mathematisch dargestellt: $A>B = g_1 \overline{k_1} + e_1 g_0 \overline{k_0}$

2. Gleichung für A<B

Die Zahl A ist kleiner als die Zahl B, wenn A1 kleiner ist als B_1, oder, wenn A_1 gleich groß ist wie B_1 und A_0 kleiner ist als B_0.

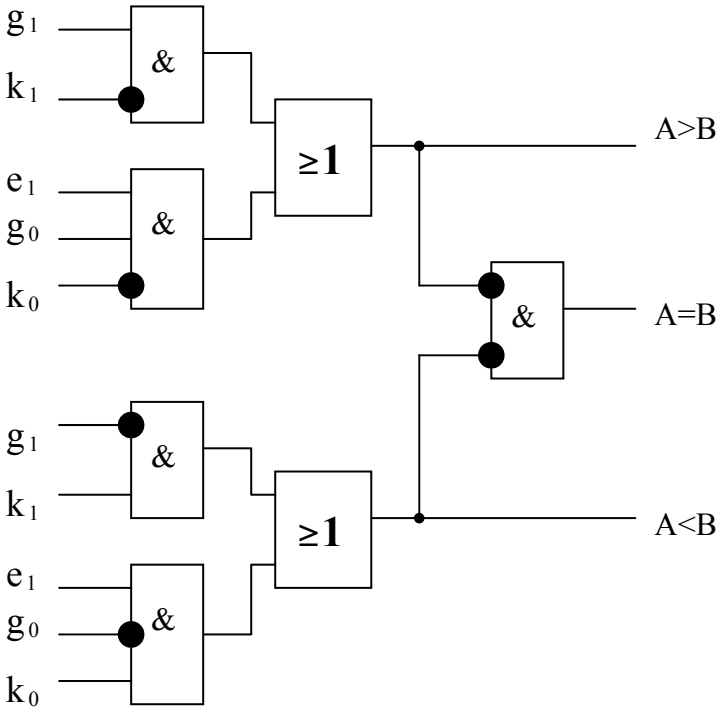
$A<B = \overline{g_1}\ k_1 + e_1\ \overline{g_0}\ k_0$

3. Gleichung für A=B

Da wir nun beim letzten der drei Ergebnisse sind, können wir hier der Einfachheit halber sagen, wenn die Zahl A nicht größer ist als die Zahl B und die Zahl A nicht kleiner ist als die Zahl B, dann sind die Zahlen gleich.

$A=B = \overline{A>B}\ \ \overline{A<B}$

Aus diesen drei Gleichungen erstellen wir nun die erweiterte Schaltung.



Bei 3-Bit Zahlen verfährt man genauso.
Die in Worte gefasste Gleichung für A>B bei zwei 3-Bit Zahlen lautet:

Die Zahl A ist größer als die Zahl B, wenn A_2 größer ist als B_2 oder, wenn A_2 gleich groß ist wie B_2 und A_1 größer ist als B_1 oder, , wenn A_1 gleich groß ist wie B_1 und A_0 größer ist als B_0.

Mathematisch dargestellt: $A>B = g_2\ \overline{k_2} + e_2\ g_1\ \overline{k_1} + e_1\ g_0\ \overline{k_0}$

Auf diese Art könnte man theoretisch unendlich erweitern. Üblich sind aber 4-Bit Vergleicher und 8-Bit Vergleicher.

Testen Sie Ihr Wissen

1.) Sie wollen aus zwei 4-Bit Volladdierern einen 6-Bit Volladdierer realiesieren. Wo liegt der Übertrag bei der fertigen Schaltung?

2.) Worin besteht der grundlegende Unterschied zwischen Halb- und Volladdierern?

3.) Wie kann man die letzte zu erstellenden Gleichung für die erweiterte Schaltung eines modular aufgebauten Vergleichers extrem kürzen?

4.) Sie wollen einen 3-Bit Vergleicher modular aufbauen. Fassen Sie Gleichung für A<B der erweiterten Schaltung in Worte!

Bistabile Kippstufen

Kapitel 7

Speichernde Schaltwerke

Bei den bisher besprochenen Schaltungen haben sich Änderungen am Eingang sofort auf den Ausgang ausgewirkt. Solche Schaltungen bezeichnet man allg. als ***Schaltnetze***. ***Schaltwerke*** haben eine speichernde Funktion. D.h. , Änderungen am Eingang wirken sich nicht sofort, sondern erst bei einem best. Eingangszustand auf den Ausgang aus, z.B. wenn der Rücksetzeingang log.0 führt, oder der Takteingang log.1 führt etc.

Im PC sind Speicher, Register, Zähler, Schieberegister…etc. bis hin zur Tastatur (entprellte Schalter) mit bistabilen Kippstufen wie Flip-Fops bzw. Latches organisiert.

Hinweis zu den Begriffen Latch / Flip-Flop:

Speichernde Schaltwerke lassen sich zur Groborientierung in zwei Ober- und zwei Unterklassen aufteilen:

- **Speicherschaltwerke ohne Taktsteuerung**
- **Speicherschaltwerke mit Taktsteuerung**
  - Zustandsgesteuerte
  - Flankengesteuerte

Ein Latch ist pegel-, oder zustandsgesteuert .
Ein Flip-Flop ist flankengesteuert.

Aufgrund der Funktionsweise von Grundgattern kann man sich vorstellen, dass man, um einen Zustand zu speichern, den Ausgangspegel zum Eingang zurückführen muss. Man braucht dazu also eine sog. ***Rückkoppelung***.

Ein einfaches Beispiel eines Schaltwerkes sehen wir in Bild 7.1.

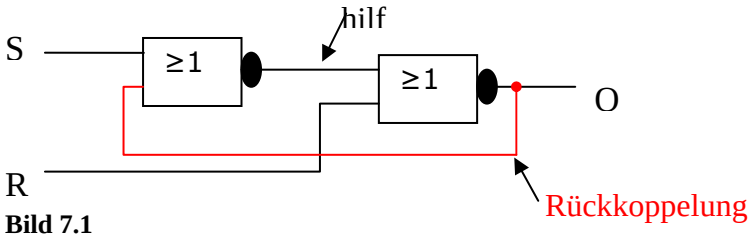


Bild 7.1

NOR

a	b	x
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Ist S=0 und Rückkoppelung=0, dann ist hilf=1.
Ist hilf=1, dann ist Q=0.

Ist S=1, dann ist hilf=0.
Ist hilf=0 und R=0, dann ist Q=1 und somit ist auch Rückkoppelung=1.

Die Rückkoppelung=1 bewirkt, dass hilf auf Null bleibt und der Ausgang Q auf Eins bleibt, solange R auch Null führt. Führt S wieder einen Null-Pegel, so hat das keine Auswirkung auf den Ausgang Q.

Es gibt 16 klassifizierte Standard-Kippstufenarten. Sechs davon schauen wir uns in diesem Kapitel näher an.

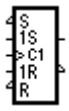
Schaltzeichen aus DigiTrace

RS-Kippstufe (ungetaktet)

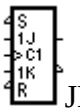
D-Latch (taktzustandsgesteuert)

RS-Latch (taktzustandsgesteuert)

D-Flip-Flop (taktflankengesteuert)



RS-Flip-Flop (taktflankengesteuert)



JK-Flip-Flop (taktflankengesteuert)

RS-Kippstufe (ungetaktet)

Ungetaktet (=nicht taktgesteuert). D.h., dass das zu speichernde Signal oder dessen Reset sofort am Ausgang übernommen werden. Die meisten Anwendungen benötigen jedoch Kippstufen, die die Signale zu bestimmten Zeitpunkten speichern. Realisiert wird dies über eine Taktsteuerung.

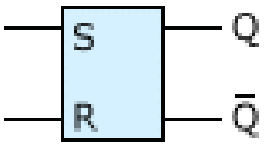


Bild 7.1 zeigt eine RS-Kippstufe. Gewöhnlich besitzt ein solches Schaltwerk einen zweiten Ausgang $\overline{Q}$, welcher den Pegelverlauf von „hilf“(siehe Bild 7.1) führt, wie in Bild 7.2.
Es gilt, wenn Q=1, dann ist $\overline{Q}$=0. Und umgekehrt, wenn Q=0, dann ist $\overline{Q}$=1.

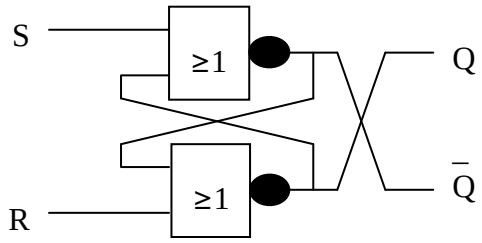


Bild 7. 2 RS-Kippstufe aus NOR

Häufig verwendetes Schaltwerk zur Realisierung einer RS-Kippstufe. (NOR)

Die ungetaktete RS-Kippstufe wird selten alleine verwendet, sondern dient eher zum Aufbau von komplexeren Schaltwerken. Es ist v.a. der Grundbaustein für alle anderen Latches und Flip-Flops.

Hinweise zum Gebrauch der Begriffe:
In der deutschen Literatur werden die Begriffe Latch und Flip-Flop vermischt. Ich achte in diesem Kapitel darauf, die Begriffe, wie in der englischen Literatur, zu differenzieren.

In der TTL-Technik wird die ungetaktete RS-Kippstufe mit NAND-Gattern realisiert.

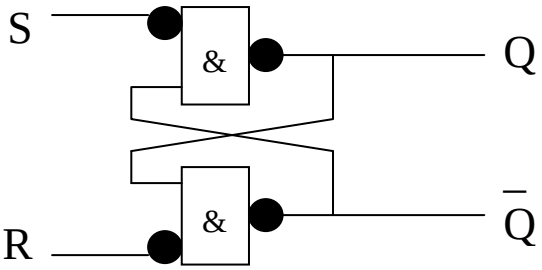


Bild 7.3 RS-Kippstufe (NAND)

Dieses Schaltwerk kommt Ihnen sicherlich bekannt vor.
In „Kapitel 3“ war dies eine Aufgabe des Praktikums. Was damals nur zur Übung mit Grundgattern galt, wollen wir heute näher untersuchen. Vor allem die Wahrheitstabelle und den vielleicht damals „unverständlichen“ verbotenen Zustand.

S	R	Q
0	0	**der alte Zustand wird gehalten**
0	1	**0**
1	0	**1**
1	1	**„verbotener Zustand“ (undefiniert)**

Funktionstabelle RS-Latch

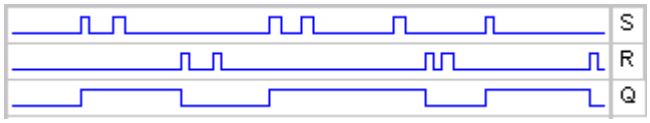
Setze ich **S** auf 1, dann ist mindestens ein Eingang des oberen NAND 0. Somit ist Q=1.
Wenn Q=1 und R=0, dann sind beide _Eingänge des unteren NAND auf 1. Das hat zur Folge, das Q=0 und somit der untere Eingang des oberen NAND 0 ist. Q=1 wird also gehalten, auch wenn ich S auf 0 setzte.
Erst wenn ich dann **R** auf 1 **R**ücksetze, nimmt der Ausgang Q den Zustand 0 wieder an.

NAND

a	b	x
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

Die Funktion von NAND ist 1, wenn mindestens ein Eingang 0 ist.

Impulsdiagramm RS-Kippstufe



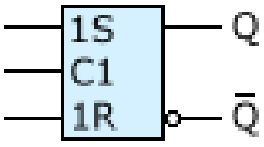
Am Impulsdiagramm, das mit Tastern am Eingang erzeugt wurde, sehen wir, sobald S auf log.1 geht, speichert der Ausgang Q den Wert von S. Mit R=1 führen wir einen Reset aus und löschen den gespeicherten Wert wieder.

Der **„verbotene Zustand“**
Die Erklärung für den verbotenen Zustand ist am einfachsten über die Logik zu verstehen. Es ist unsinnig zur gleichen Zeit ein Signal zu setzen und dasselbe Signal zurückzusetzen.

Eine zusätzliche Erklärung dafür ist, dass die vorhandenen Funktionen der davon betroffenen Kippstufen ausreichend sind. Eine Vermeidung oder Änderung des verbotenen Zustands lässt sich nicht ohne Geschwindigkeitsverlust oder Verändern der anderen drei Funktionen realisieren.

Deshalb werden solche Schaltwerke nahezu symetrisch angelegt. Kleinste Abweichungen in der Symetrie, wie Leitfähigkeit oder Länge der Verbindungen, Störfelder….etc. lassen im verbotenen Zustand auf keinen bestimmten Ausgangszustand schließen. Wenn man also vom Speicherzustand (S=0,R=0) direkt auf den verbotenen Zustand (S=1,R=1) schaltet, kann nicht im Voraus definiert werden, welcher der beiden Ausgänge log.1 annimmt und welcher log.0 annimmt. Dies bestimmt die winzige Asymetrie des Schaltwerks.
Bei Bedarf wird diese Asymetrie mit einem kleinen Trick umgangen. (siehe D-Latch)

RS-Latch (taktzustandsgesteuert)



Der Unterschied zum “einfachen” RS-Latch besteht darin, dass das zustandsgesteuerte RS-Latch einen zusätzlichen Takteingang C (Clock) besitzt und sich Änderungen am Eingang nur dann auf den Ausgang Q auswirken, wenn C einen log.1-Pegel führt.

NAND

a	b	x
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

Aufbau eines RS-Latch

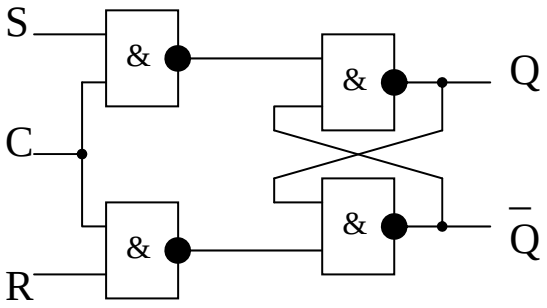
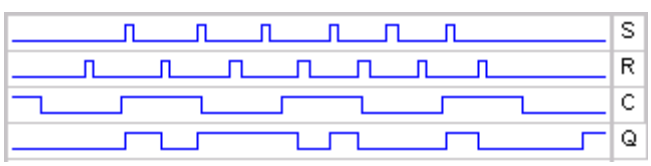


Bild 7.4 RS-Latch

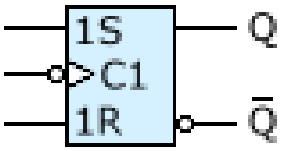
Die Taktzustandssteuerung hat gegenüber der ungetakteten RS-Kippstufe den Vorteil, dass Eingangswerte nicht zu jedem Zeitpunkt, sondern nur zu bestimmten Momenten gespeichert werden.

Anhand des Aufbaus und des Impulsdiagramms kann man nachvollziehen, dass sich Änderungen am Eingang nur dann auf den Ausgang auswirken, wenn C einen log.1-Pegel führt.

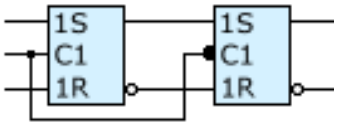
Impulsdiagramm RS-Latch



RS-Flip-Flop (taktflankengesteuert) (getriggert auf negative Flanke)



Das RS-Flip-Flop besteht aus zwei hintereinander geschalteten, RS-Latches. Mit der negativen Taktflanke wird der vorher gesetzte Wert am Ausgang übernommen.



Aufbau eines RS-Flip-Flop (getriggert auf negative Flanke)

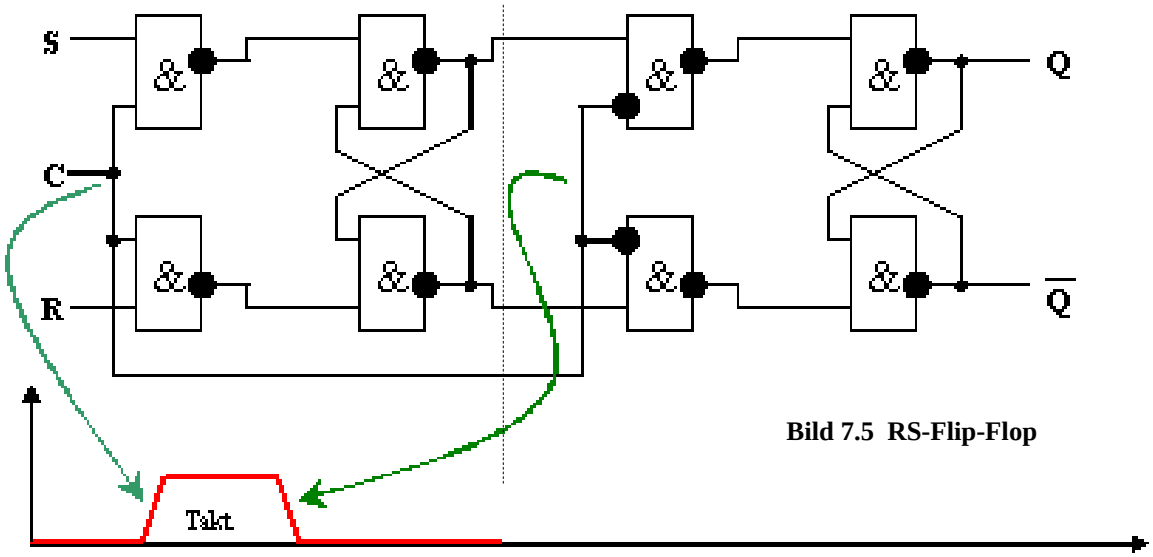
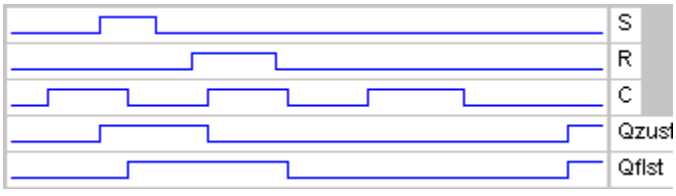


Bild 7.5 RS-Flip-Flop

- Mit C=1 wird das an S oder R anliegende Signal in die erste Stufe übernommen.
- Unterdessen liegt am Ausgang Q noch der alte Wert an.
- Mit der ***fallenden Flanke*** von C wird die erstes Stufe sozusagen „eingefroren“ und das gesetzte Signal der ersten Stufe in die zweite Stufe übernommen.

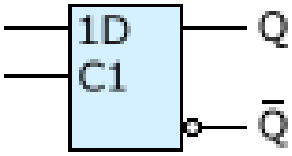
Taktzustandsgesteuert v.s. taktflankengesteuert



Das Impulsdiagramm zeigt das unterschiedliche Verhalten der Ausgänge Q eines RS-Latch und eines RS-Flip-Flop (negative Flanke). Der Ausgang des Latch übernimmt das Signal S sofort, wenn der Zustand von C log.1 ist. Der Ausgang des Flip-Flop (getriggert auf negative Flanke) übernimmt das Signal von S erst, wenn der Taktzustand von log.1 nach log.0 wechselt.

Flip-Flops bestehen intern grundsätzlich aus zwei Latch-Stufen. Die erste Stufe speichert ein Signal und die zweite Stufe übernimmt bei einem bestimmten Eingangszustand der zweiten Stufe das Signal.
Wenn ein Flip-Flop auf negative Flanke getriggert ist, ist die zweite interne Stufe ein zustandsgesteuertes Latch, das Änderungen übernimmt, wenn C log.0 ist (siehe Bild 7.5).

D-Latch (taktzustandsgesteuert)



Bei einem RS-Latch hatten wir das Problem des „verbotenen Zustands“. Bei einem D-Latch wird dieser Zustand mit einem einfachen Trick umgangen.
Man verbindet die Eingänge S und R und schaltet vor den R-Eingang einen Inverter.

Aufbau eines D-Latch, taktzustandgesteuert

Bei C=1 wird der an D anliegende Wert am Ausgang Q übernommen. Der einzige Eingang wird als „Data“ bezeichnet.
Dieses Latch ist u.a. das Grundelement für Schreib-Lese-
-Speicher.

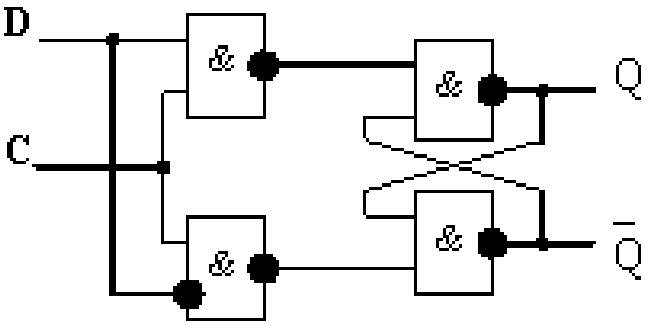
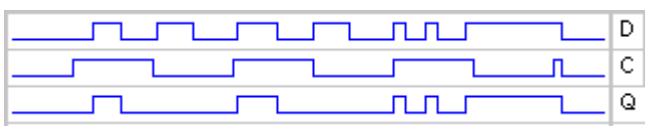


Bild 7.6 D-Latch

D	C	Q
0	0	**Latch nicht aktiv, der alte Zustand wird gehalten, kein Signal**
0	1	**0 (Reset, Signal löschen)**
1	0	**Latch nicht aktiv, der alte Zustand wird gehalten, Signal gesetzt**
1	1	**1 (Set, Signal übernehmen)**

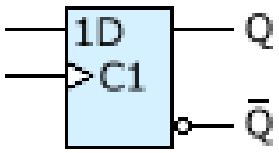
Funktionstabelle D-Latch

Impulsdiagramm D-Latch

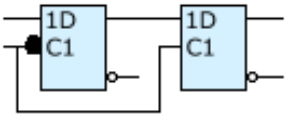


Wir sehen, nur wenn C log.1 führt, werden Signale an D am Ausgang Q übernommen.
Liegt an C log.0 an, dann wird am Ausgang zwar der alte Zustand gehalten, Änderungen am Eingang aber nicht übernommen.
Allg. arbeitet das D-Latch genauso wie ein RS-Latch, nur mit dem Unterschied, dass für S und für R nur ein Eingang vorhanden ist.
Führt dieser Eingang (D) eine log.1, so wäre das bei einem RS-Latch S=1, R=0.
Führt dieser Eingang (D) eine log.0, so wäre das bei einem RS-Latch S=0, R=1.

D-Flip-Flop (taktflankengesteuert) (getriggert auf positive Flanke)



Ein D-Flip-Flop besteht aus zwei hintereinander geschalteten D-Latches. Um dieses Flip-Flop auf positive Flanke zu triggern muss vor den Takteingang der Masterstufe ein Inverter geschaltet werden. (Um auf negative Flanke zu triggern müsste der Inverter vor den Takteingang der Slavestufe.)



Aufbau eines D-Flip-Flop (getriggert auf positive Flanke)

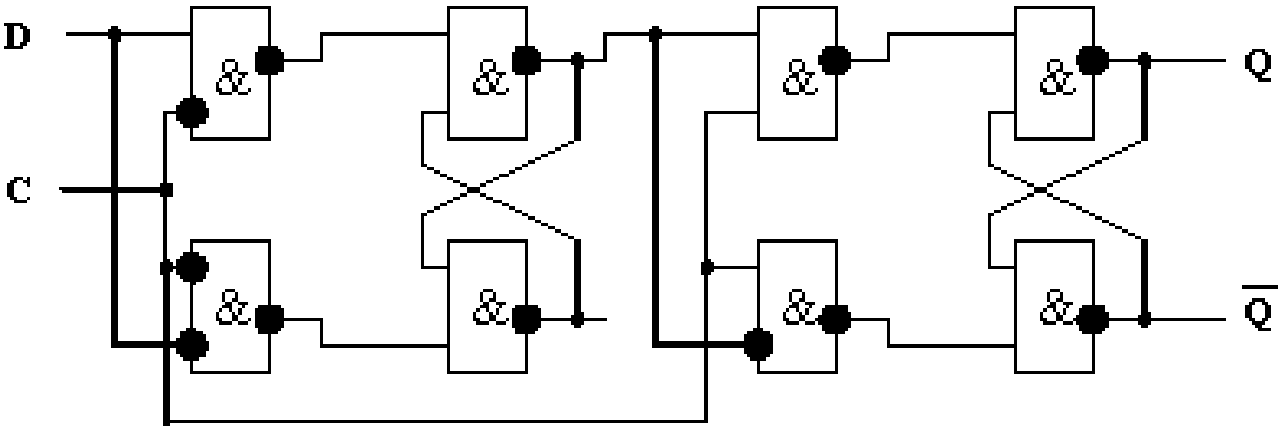
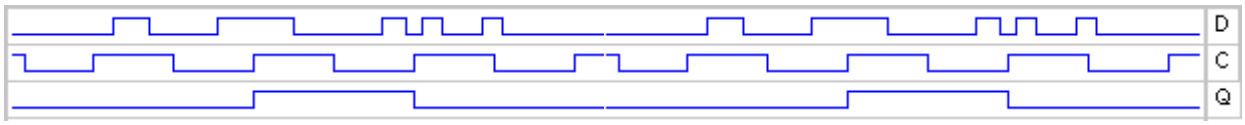


Bild 7.7 D-Flip-Flop

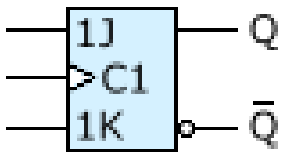
- Mit C=„ positive Flanke“, wird das an D anliegende Signal am Ausgang Q übernommen.
- Mit C=„ negative Flanke“, log.0 oder log.1 erfolgt keine Änderung an Q.

Impulsdiagramm D-FF (getriggert auf positive Flanke)



Am Impulsdiagramm ist gut zu erkennen, dass bei einem D-FF wirklich nur der kurze Moment entscheidend ist, an dem der Takt C von log.0 auf log.1 wechselt. Bei allen anderen Taktzuständen erfolgt keine Änderung an Q.

JK-Flip-Flop (taktflankengesteuert) (getriggert auf negative Flanke)



Aufbau eines JK-Flip-Flop (getriggert auf negative Flanke)

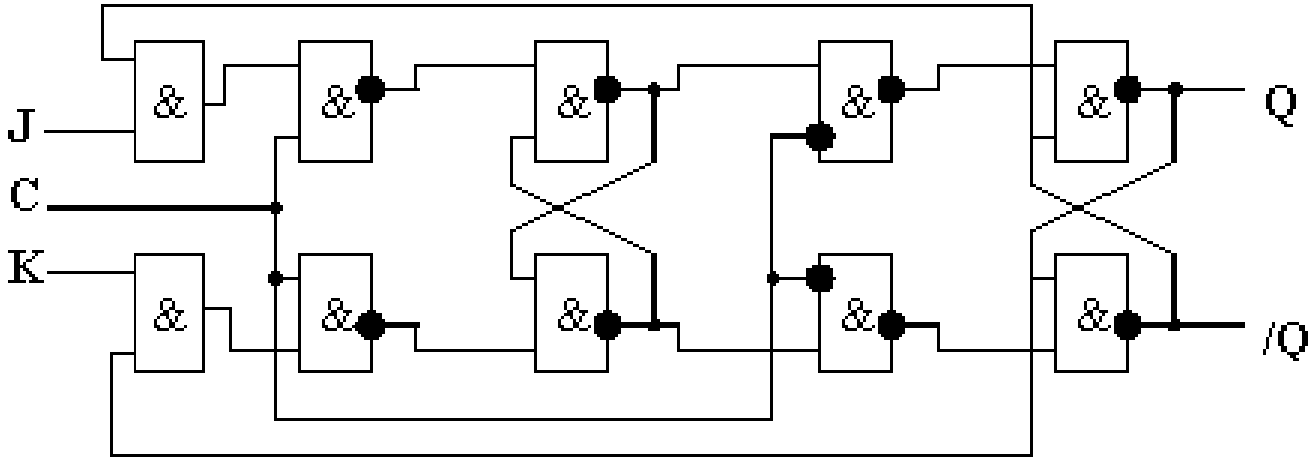
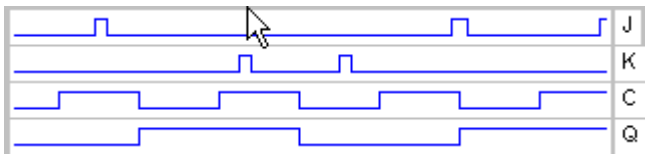
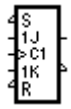


Bild 7.5 JK-Flip-Flop

Impulsdiagramm JK-FF (getriggert auf negative Flanke)



Das Impulsdiagramm veranschaulicht, dass die jeweiligen Signale mit der negativen Taktflanke ins Slave-FF übernommen werden.

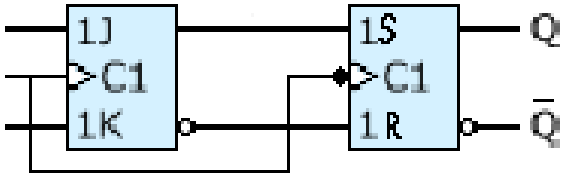
JK-Flip-Flop (zweiflankengesteuert) 

Eines der universellsten Vertreter seiner Art.

J=0 ,K=0	Q=Q-1	Speichern
J=1, K=0	Q=1	Setzen
J=0, K=1	Q=0	Rücksetzen
J=1, K=1	Q=/Q	Toggle (Q ändert sich bei jedem Takt)

Bei der steigenden Taktflanke wird die gewünschte Funktion in das Master-Flip-Flop eingelesen die bei fallender Taktflanke vom Slave-Flip-Flop ausgeführt wird.

Prinzipaufbau



Ein JK-Flip-Flop wechselt bei Anlegen eines Taktimpulses seinen Ausgangszustand, wenn an beiden Eingängen H-Pegel anliegen. Dieses Verhalten wird als Toggeln(kippen) bezeichnet. Wenn ein JK-Flip-Flop RS-Eingänge hat, so lässt es sich taktunabhängig Steuern.
Bei diesem Flip-Flop ist der unbestimmte Zustand ausgeschlossen.

Wahrheitstabelle

K	J	C1	Q	/Q	Funktion
0	1	0	n	n	Speichern
0	0	0	n	n	Speichern
1	0	0	n	n	Speichern
1	1	0	n	n	Speichern
0	1	1	1	0	Setzen
0	0	1	n	n	Speichern
1	0	1	0	1	Rücksetzen
1	1	1	X	X	Wechseln(Toggeln)

- Liegt kein High-Pegel am Takteingang, so wird der an den Ausgängen anstehende Pegel gespeichert.
- Liegt am Setzeingang(J) und am Takteingang(C) ein High-Pegel, so wird das Flip-Flop gesetzt.
- Liegt am Rücksetzeingang(K) und am Takteingang ein High-Pegel, so wird das Flip-Flop zurückgesetzt.
- Liegt an beiden Steuereingängen ein High-Pegel, so wird der gespeicherte Wert gewechselt, d.h. aus High wird Low, aus Low wird High.

Testen Sie Ihr Wissen

1.) Beschreiben Sie die Begriffe Kippstufe, Latch, Flipflop!

2.) Was versteht man unter dem Begriff „negative Flanke“ ?

3.) Beschreiben Sie die Besonderheiten eines D-Latch!

4.) Im Bauteilekatalog von DigiTrace wird ein JK-FF angeboten. Beschreiben Sie in Stichpunkten die Funktionen!