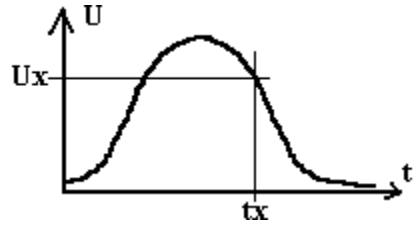
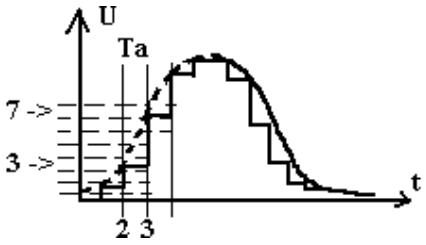
13. Digitaltechnik

Digitale Information kann aus einzelnen **Zeichen** bestehen, welche digital codiert sind. Z.B. Buchstaben, die in Form eines Bytes codiert und gespeichert werden. Oder es handelt sich um **Signale**, das heisst eine Sequenz (Abfolge) von physikalischen Grössen, z.B. um einen Temperaturverlauf oder ein Tonsignal, welches eine Folge von Amplitudenwerden darstellt, wobei jeder Wert als digitales Zeichen codiert und gespeichert wird.

13.1 Analoge und digitale Signale

	Bei der Darstellung oder Abbildung eines **analogen** Signals kann jedem beliebigen Zeitpunkt genau ein Signalwert zugeordnet werden. Die Signalwerte und Zeitwerte können beliebig genau angegeben werden und prinzipiell **jeden Wert** annehmen. Man sagt, das Signal sei **zeit- und wertekontinuierlich.** Bsp: Im Bild kann man zum Beispiel sagen, zur Zeit tx=1.68307s sei der Messwert Ux=7.39231 Volt.
	Bei der **digitalen** Abbildung eines kontinuierlichen Signals sind nur eine beschränkte Anzahl von **Signal**werten (U-Achse) und von **Zeit**werten (t-Achse) möglich. Dies ergibt die typische Treppenfunktion. Wenn ein digitales Signal aus einem analogen Signal entsteht, geschehen mehrere Schritte:

13.1.1 Die Abtastung:

In regelmässigen Zeitabständen (Abtastzeit Ta) wird eine Probe, eine Momentaufnahme (ein sample) des Signals genommen. Die Abtastfrequenz ist 1/Ta.
Damit kein Informationsverlust entsteht, muss die **Abtastfrequenz** (Häufigkeit) dieser **Abtastung *fa* mindestens doppelt** so gross sein wie die maximale Frequenz ***fmax*** des abgetasteten Signals.

Diese Regel heisst **Abtasttheorem**.

$$f_a \geqq 2 \cdot f_{max}$$

Je grösser die Abtast-Frequenz ist, desto besser ist die Auflösung des Signals (das digitale Signal ist näher beim analogen).
Die Abtastung bewirkt, dass es viele Zeitpunkte gibt, in denen der genaue Signalwert unbekannt bleibt. Man sagt nun, das Signal sei **zeitdiskret**.

Je **mehr Stützwerte** es gibt, desto **günstiger (höher)** ist die **Abtastrate** des Signals.

13.1.2 Quantisierung

Der Signalwert, der bei der Abtastung ermittelt wurde, wird nun einer **Messwerte-Stufe oder -Klasse zugeteilt**. Es ist nicht mehr jeder beliebige Signalwert denkbar, es gibt also nur eine beschränkte Anzahl möglicher Signalwerte.
Das Signal ist nun zusätzlich **wertediskret**.

Bsp: Im Bild oben heisst das, der zur Zeit 2s abgetastete Wert wird der Klasse 3 zugeordnet, der Wert bei t=3s wird der Klasse 7 zugeordnet, obwohl die Werte nicht genau 3 oder 7 waren. Dieser Vorgang heisst Quantisierung.
Auch hier gilt: Der ganze Amplitudenbereich muss in mindestens zwei Stufen eingeteilt werden. Besser sind natürlich wesentlich mehr.

Je **mehr Stufen** es gibt, desto **besser** ist die **Auflösung** des Signals.

Die Quantisierung bewirkt, dass man von einem bestimmten Signalwert nicht mehr genau sagen kann, zu welcher Zeit er exakt gegolten hat.

13.1.3 Codierung

Mit den ersten beiden Schritten ist die Digitalisierung eines analogen Signals im Prinzip abgeschlossen.
Unter Codierung versteht man eigentliche die **Numerierung** der Stufen im binären Zahlensystem. Im der obigen Grafik war geschah die Numerierung mit Dezimalzahlen.

Allgemein kann man eine Menge von Nachrichten (z.B. Zahlen, Wörter, Signalstufen) mit verscheidensten Codierungsarten darstellen.
Jede Codierungsart hat eine bestimmte Anzahl Zeichen (z.B. Ziffern, Buchstaben) zur Verfügung.

Beispiele:

Codierungsart	**Anzahl Zeichenarten**
Chinesische Schrift	3000 bis 9000
Lateinische Schrift	25-29
Zahlen im Dezimalsystem	10
Zahlen im Binärsystem	2

Eine Menge Nachrichten kann man mit **vielen Zeichen und wenigen Stellen** darstellen oder mit **wenigen Zeichen und vielen Stellen**.

Technisch problematisch sind Codierungsarten, die viele Zeichenarten und wenige Stellen aufweisen. Man stelle sich eine chinesische Schreibmaschine vor, wenn für jedes Zeichen (Ton) eine Taste benötigt würde.
Technisch einfach sind Codierungsarten mit wenigen Zeichenarten, allerdings steigt dadurch die nötige Anzahl Stellen.

13.2 Binäre Codierung (Dualsystem)

Um technisch einfache Codierungen zu haben, wählte man das Binärsystem oder Dualsystem. Es kennt nur zwei verschiedene Zeichen.
An jeder Stelle kann es nur zwei Zustände annehmen.
Beispiele:

Stelle	**Zustand 1**	**Zustand 0**
Schalter	geschlossen	offen
Spannung	vorhanden	nicht vorhanden
Lochkarte	Loch	kein Loch
Transistor	leitend	nicht leitend
Magnetplatte	magnetisiert	nicht magnetisiert
Speicherzelle (RS-FlipFlop)	gesetzt	nicht gesetzt
Zeichen	1	0
Logik-Spannungspegel (Level)	HIGH (der positivere Spannungspege z.B. 5 V)	LOW (z.B. 0 V)
Logischer Zustand	wahr (TRUE)	falsch (FALSE)

Die Digitaltechnik müsste genauer **binäre Digitaltechnik** heissen, da man sich für die binäre Codierung entschieden hat.

13.3 Binäre Informationseinheiten

Ein Signal oder Zeichen, das nur zwei Werte annehmen kann,
heisst Bit (**bi**nary digi**t**, binäres Code-Element).
- 8 Bits bilden bekanntlich ein **Byte.**
- 4 Bits bilden ein **Nibble** (Halbbyte).
- 16 Bits bilden ein **Wort (Word).**
- 32 Bits bilden ein **Doppelwort.**

13.4 Zahlensysteme

Eine Zahl lässt sich im Prinzip in jedem beliebigen Zahlensystem, also einem Codierungssystem, darstellen. Zahlensysteme wie unser bekanntes Zehnersystem sind sogenannte **Stellenwertsysteme**. Das heisst, dass jede Stelle einer Zahl einen gewissen Stellenwert hat. Dieser Stellenwert ist ein Vervielfachungsfaktor, mit dem die Ziffer an dieser Stelle multipliziert werden muss.
Dieser Vervielfältigungsfaktor heisst auch Wertigkeit.
Jede Stelle hat eine Wertigkeit in Form einer Potenzzahl.
Diese Potenzzahl bildet sich aus der **Basis B** des Zahlensystems, potenziert mit der **Stellennummer n** als Exponent (B^n)**.**
Die Stelle Nummer null ist die letzte vor dem Komma.

Beispiel: Die Zahl 497,6 im **Zehner**system, Ziffernumfang: **10** Zeichen (0...9)

Ziffern der Dezimalzahl	4	9	7,	6
Basis	10	10	10	10
Stellennummer von rechts (Exponent)	2	1	0	-1
Wertigkeit (Potenzzahl)	10^2	10^1	10^0	10^{-1}
Ziffer x Faktor	**4 • 100**	**9 • 10**	**7 • 1**	**6 • 0.1**
Bedeutung: 497.6 = Summe aus:	+400	+90	+7	+0.6

Merkmale von Stellenwertsystemen:
- Jedes Zahlensystem hat nur soviele Ziffern Umfang, wie die Basis angibt, und zwar jeweils von 0 bis (Basis-1).
- Die höchstwertige Stelle ist links, die tiefstwertige Stelle rechts
- Zeichen nach dem Komma werden analog gebildet, die Stellennumerierung und damit der Exponent der Potenzzahl wird nach dem Komma negativ.

Beispiele zum Ziffernumfang:
- Im Siebnersystem kommen die 7 Ziffern 0 bis 6 vor.
- Im Dualsystem kommen die 2 Ziffern 0 und 1 vor.
- Im 16er-System braucht man 16 Ziffern:
  Die 10 Zeichen 0 bis 9 und für die "Ziffern" "10" bis "15" die Buchstaben A bis F.

Auf diese Weise lässt sich jede Zahl in **jedem beliebigen** Zahlensystem darstellen.

Beispiel:
Die Darstellung der dezimalen Zahl 72 im Siebnersystem:

Basis	7	7	7
Stellennummer von rechts (Exponent)	2	1	0
Stellenwertigkeit (Potenzzahl)	7^2	7^1	7^0
ergibt Faktor	**49**	**7**	**1**
Faktor hat wie oftmal Platz? (Division mit Restbildung)	1 • 49 Rest 23	23= 3 • 7 Rest 2	2 • 1
Ziffern der Siebnerzahl	**1**	**3**	**2**

Resultat: $72_{10} = 132_7$ (Lies: zweiundsiebzig im Zehnersystem = Eins drei zwei im Siebnersystem)

Aus diesen Beispielen wird schon erkennbar, wie andere häufig verwendete Systeme, das Binärsystem (2er-System) und das Hexadezimalsystem (16er-System) aufgebaut sind.

13.4.1 Das Zweiersystem (Binärsystem, Dualsystem)

Analog kommt man zum Zweiersystem oder Binärsystem.
Das Binärsystem wird in der ganzen Digitaltechnik universell verwendet.

Zeichenumfang: Die 2 Ziffern 0 und 1

Basis	2	2	2	2	2	2	2	2
Stellennummer von rechts (Exponent)	7	6	5	4	3	2	1	0
Wertigkeit (Potenzzahl)	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
ergibt Faktor	128	64	32	16	8	4	2	1
Beispiel: Dualzahl 10110001 = ?	1	0	1	1	0	0	0	1
= Dezimalzahl **177** =	128		+32	+16				+1

Für Stellen nach dem Komma wird analog dem Zehnersystem vorgegangen.
Die erste Stelle nach dem Komma hat die Wertigkeit 2^{-1}, also 1/2 oder 0,5. Danach folgt 1/4, 1/8, 1/16 usw.
Auch hier wie in jedem Zahlensystem liegt auch hier die höchstwertige Stelle links, die tiefstwertige Stelle rechts.

Beim Binärsystem heissen die Stellen **Bits**.

Die höchstwertige Stelle heisst **most significant bit**, abgekürzt **MSB** (das gewichtigste Bit).

Die tiefstwertige Selle heisst **least significant bit**, abgekürzt **LSB** (das aussageschwächste Bit).

13.4.2 Das Hexadezimalsystem

In der Computertechnik wird es häufig verwendet, weil die Zahlen dadurch kürzer werden. 16 entspricht zudem einer WORD-Breite, und eine Hex.zahl kann auf einfache Weise in Bytes und Bits umgewandelt werden (siehe 13.4.4).

Zeichenumfang: Die 16 Ziffern 0 bis 9 und A bis F

Basis	16	16	16	16
Stellennummer von rechts (Exponent)	3	2	1	0
Wertigkeit (Potenzzahl)	16^3	16^2	16^1	16^0
ergibt Faktor	4096	256	16	1
Beispiel: Hexzahl 2FA9 = ?	2	F (=15d)	A (=10d)	9
bedeutet:	2 • 4096	15 • 256	10 • 16	9 • 1
= Dezimalzahl **12201** =	8192	+3840	+160	+9

13.4.3 Umwandlung in verschiedene Zahlensysteme

Die beiden Beispiele unter 13.4 zeigen das Umwandeln **ins Zehnersystem** (Summe aus den Gewichtungen x Ziffer) und das Umwandeln **vom Zehnersystem** (Division mit Restbildung). Für die Umwandlung von und in beliebige Systeme gibt es Algorithmen, die hier nicht näher behandelt werden. Man kann natürlich auch den Umweg über ein vertrautes System wähen.

13.4.4 Umwandlungen zwischen Dualsystem und Hexadezimalsystem

Einfacher ist die Umwandlung zwischen Systemen, deren Basis eine Zweierpotenz ist (2er, 4er, 8er,16er-System). Hier kann man

- **vom höheren ins tiefere System** wandeln, indem man **zeichenweise** vorgeht:
- **vom tieferen ins höhere System** wandeln, indem man **Gruppen bildet**.

Beispiele:

! Hausaufgabe !

Lernen Sie jene Funktionen Ihres Taschenrechners kennen, mit denen Sie alle **Umwandlungen** von natürlichen Zahlen zwischen Dualsystem, Dezimalsystem und Hexadezimalsystem vornehmen können oder schreiben Sie ein Programm für den Rechner, welches diese Aufgaben vornehmen kann! In Proben können solche Aufgaben kommen!

13.4.5 Addition von Zahlen in beliebigen Zahlensystemen

Im Prinzip erfolgen die mathematischen Operationen analog dem Zehnersystem. Allerdings erfordert dies besondere Aufmerksamkeit, da wir uns dieses Rechnen nicht gewohnt sind.

Beispiele:

Wir können natürlich auch alle Operanden zuerst ins Zehnersystem umwandeln und nach der Addition wieder zurückwandeln.
Die übrigen Operationen sind wesentlich komplizierter und werden hier nicht behandelt.

13.4.6 Der BCD-Code

BCD heisst **b**inary **c**oded **d**igit = binär codierte (Dezimal-)Ziffer
Beim BCD-Code wird eine Zahl aus dem Zehnersystem nicht als ganzes binär umgewandelt, sondern jede Stelle einzeln mit einer 4-bit-Binärzahl codiert.
Eine so codierte Zahl ist länger; es werden ja nur 10 der 16 möglichen 4-bit-Kombinationen genutzt.

Beispiele:

In der Praxis ist dieser Code recht verbreitet, zum Beispiel für die Ansteuerung von Digitalanzeigen (7-Segment-Anzeigen).

13.4.7 Überblick, Umwandlungstabelle

Dezimalzahl	**Dualzahl** (Binärsystem)	**BCD-Code**	**Oktalzahl** (Achtersystem)	**Hexadezimalzahl** (Sedezimalsystem, Sechzehnersystem
0	0	0	0	0
1	1	1	1	1
2	10	10	2	2
3	11	11	3	3
4	100	100	4	4
5	101	101	5	5
6	110	110	6	6
7	111	111	7	7
8	1000	1000	10	8
9	1001	1001	11	9
10	1010	nicht definiert	12	A
11	1011	nicht definiert	13	B
12	1100	nicht definiert	14	C
13	1101	nicht definiert	15	D
14	1110	nicht definiert	16	E
15	1111	nicht definiert	17	F
16	10000	nicht definiert	20	10

13.5 Aufgaben und Einsatzgebiete der Digitaltechnik

- Arithmetische Operationen mit Zahlen (Zahlen binär codiert)
- Vergleich digitaler Signale (grösser, gleich, kleiner)
- Zählen von Ereignissen und Darstellen in einem digitalen Code
- Code-Umwandlungen (z.B. von serieller in parallele Darstellung oder von einer Codierungsart in die andere)
- Signalspeicherung
- Logische Verknüpfungen (Zustandsabhängige Wenn-dann-Verknüpfungen)
- Ablaufsteuerungen (Ereignisabhängige Sequenzen, Automaten, Programme)

13.6 Schaltalgebra (Boolsche Algebra)

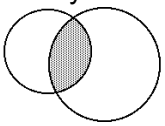
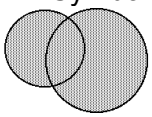
Schaltalgebra ist eine Algebra mit Variablen, die nur die beiden logischen Werte 0 und 1 (oder wahr und falsch) annehmen können.
Mit diesen Werten kann man auf verschiedene Weisen operieren:

- sprachlich-logisch (also mit Logik)
- algebraisch-theoretisch (mit Schaltalgebra)
- technisch-praktisch (mit digitalen, elektronischen Schaltelementen)

In der Schaltalgebra gibt es **drei Hauptoperationen** (-funktionen, -verknüpfungen):

• die **UND-Operation**	Operationszeichen: $\wedge$
• die **ODER-Operation** (gemeint ist das **einschliessende Oder !!!** also das sprachliche "und/oder") Gleichbedeutend mit *"und/oder"* ist *"mindestens eins"*	Operationszeichen: $\vee$
• die **Negation** (oder Inversion oder Umkehrung)	Über eine oder mehrere Variablen kommt ein Querstrich. Beispiel: $\overline{R}$

Schwierig ist das Unterscheiden der Operationszeichen UND $\wedge$ bzw. ODER $\vee$.
Das UND-Zeichen $\wedge$ hat immerhin noch eine Ähnlichkeit zum englischen **A**ND $\wedge$. Vielleicht hilft Ihnen der Vergleich zur Mengenlehre. Dort gibt es Symbole für die Schnittmenge und die Vereinigungsmenge. Von diesen sind auch die logischen Operatoren und Zeichen für UND $\wedge$ sowie ODER $\vee$ abgeleitet:.

Schnittmenge: A geschnitten mit B	**V**ereinigungsmenge: A vereinigt mit B
• Symbolische Darstellung: A $\cap$ B  • Schnittmenge = A $\wedge$ B (A **und** B) • Sprachlich: Zur Schnittmenge gehören jene Elemente, die gleichzeitig zu A **und** B gehören.	• Symbolische Darstellung: A $\cup$ B  • Vereinigungsmenge = A $\vee$ B (A **oder** B) • Sprachlich: Zur Vereinigungsmenge gehören jene Elemente, die entweder zu A **oder** B gehören (oder wie gezeichnet zu beiden, einschliessendes Oder!).

13.6.1 Rechenregeln der Schaltalgebra

Es gelten ähnliche Regeln wie bei den bekannten Operationen mit reellen Zahlen.
Neu sind vor allem die beiden Logik-**Regeln von De Morgaan**. (13.6.2)

-> Siehe "Gehlen"-Tabellenbuch S. 109 oder Kopie davon!

13.6.2 Erläuterungen zu den Regeln von De Morgaan (De Morgaansche Gesetze)

Regel 1: $\overline{A \vee B} = \overline{A} \wedge \overline{B}$ Nicht (A oder B) = Nicht A und nicht B	Beispiel aus der Sprachlogik: Gegeben die Aussage: Sabine geht weder ins Kino noch ans Konzert. Diese Aussage kann auf zwei Arten "ditgital" ausgedrückt werden: Sabine geht NICHT (ins Kino ODER ans Konzert) = Sabine geht (NICHT ins Kino) UND (NICHT ans Konzert).
Regel 2: $\overline{A \wedge B} = \overline{A} \vee \overline{B}$ Nicht (A und B) = Nicht A oder nicht B	Beispiel aus der Sprachlogik: Gegeben die Aussage: Martin besitzt entweder einen Mac oder einen PC (aber nicht beides). Mögliche "digitale" Formen nach De Morgaan: Martin hat nicht (einen Mac und einen PC) = Martin hat (keinen Mac) oder er hat (keinen PC)

13.7 Die logischen Grundfunktionen

Wie jede mathematische oder technische Funktion ermittelt auch eine logische Funktion aus **beliebig vielen** Eingangsvariablen (Eingängen) **genau eine** Ausgangsvariable (Ausgangszustand). Wenn es mehrere Ausgangsvariablen gibt, braucht es auch mehrere Funktionen. Die Funktion lässt sich als

- **Symbol** oder als
- **Wahrheitstabelle** oder als
- **Boolsche Funktionsgleichung** darstellen.

Details siehe "Gehlen"-Tabellenbuch S. 108 bzw. Kopie davon!

13.8 Elementare logische Verknüpfungen

Noch einmal: Die kopierten Tabellen aus dem Buch mit den Überschriften
- "**Elementare logische Verknüpfungen...**" und
- "**Regeln der Schaltalgebra**"
sind Bestandteil der **Theorie** und werden als **Grundlage** vorausgesetzt!
Die Tabellen dürfen und sollen auch in Proben verwendet werden.

Prioritätsordnung der Boolschen Operationen

1.	Höchste Priorität:	Negation	NOT
2.	Punktrechnung vor	UND	AND
3.	Strichrechnung	ODER	OR
4.	gleichgestellt:	Äquivalenz, Antivalenz	EXNOR EXOR

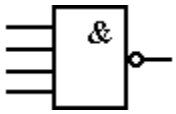
Falls in Boolschen Gleichungen eine andere Priorität erforderlich ist, **müssen** Klammern gesetzt werden.
Zwecks besserer Übersichtlichkeit **dürfen** auch bei klarer Prioriätsordung Klammern gesetzt werden.

Beispiele von elementaren Verknüpfungen in Kombination mit Negationen:

13.9 Kombinatorische Schaltungen: Kombinationen von elementaren Verknüpfungen

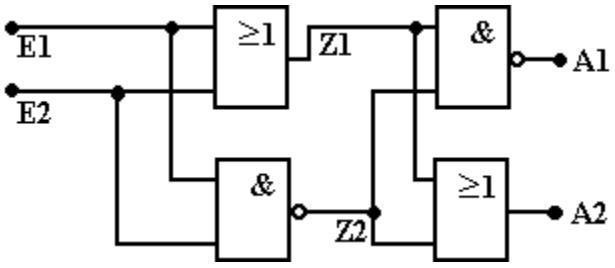
Die elektronischen Elemente, welche die elementaren Verknüpfungen ermöglichen, werden auch Gatter genannt, englisch: gate. Die Gatter können dabei mehrere Eingänge haben.

Bsp: Ein 4-input-NAND-gate ist ein NAND-Gatter mit 4 Eingängen.



Der Ausgang ist nur dann 0, wenn alle Eingänge 1 sind.
Oder:
Der Ausgang ist immer dann 1, wenn mindestens ein Eingang 0 ist.

Bildet man Kombinationen aus beliebigen Gattern, so ergibt sich eine **kombinatorische Schaltung** wie im nebenstehenden Beispiel.
Eine Kombinatorische Schaltung kann mehrere Ausgänge haben. Jeder einzelne Ausgang muss dann als Funktion der Eingange beschrieben werden können.



Wahrheitstabelle der Schaltung:

E2	E1	Z1	Z2	A1	A2

Merkmale und Regeln Kombinatorischer Schaltungen

- Beliebig viele Grundelemente mit jeweils beliebig vielen Eingängen dürfen zusammengeschaltet sein.
- Die gesamte Kombinatorische Schaltung hat dann **beliebig viele Eingänge** und kann auch **beliebig viele Ausgänge** haben.
- Jede Eingangskombination ergibt einen **genau bestimmten** Zustand am Ausgang oder an den Ausgängen.
- Ein Ausgangszustand hängt nie vom vorhergehenden Zustand ab.
- Jeder Ausgang lässt sich mit einer Wahrheitstabelle oder einer Funktionsgleichung beschreiben.
- **Rückführungen** von Ausgängen auf Eingänge **darf es nicht geben**!
  Sonst sind es keine kombinatorischen Schaltungen (sondern unter Umständen so genannt sequentielle Schaltung)
- Es dürfen **nie mehrere Ausgänge zusammengeschaltet** werden! Sie wirken wie Quellen und würden einander "bekämpfen".
- Aus dem gleichen Grund dürfen keine Eingangssignale auf Ausgänge wirken.

Anhand von Beispielen sollen im Folgenden
- die Wahrheitstabellen
- die Boolschen Funktionen, Gleichungen und Ausdrücke
- die Symbolik mit den Schaltungselementen
- die DeMorganschen Regeln

vertieft werden.

Anwendung zu 13.9: Entwicklung einer Kombinatorischen Schaltungen:

In einem Chemielager soll eine automatische Löschanlage eingerichtet werden.
Um irrtümliche Alarm-Auslösungen zu vermeiden und gleichzeitig eine hohe Ansprechchance zu wahren, werden an jeder Messstelle 3 Feuermelder installiert.
Die Löschanlage soll nur dann aktiviert werden, wenn mindestens zwei der drei Feuermelder an einer bestimmten Messstelle Alarm aauslösen.

Gesucht Eine Digitalschaltung für **eine** Messstelle, welche nur dann eine 1 am Ausgang hat, wenn mindestens zwei der drei Eingänge 1 sind. (So genannte Zwei-aus-drei-Schaltung)

Im vorigen Beispiel (Alarmanlage Chemielager) haben wir gesehen, dass man die Digitalschaltung auch aus NAND-Gattern aufbauen kann.

Diese Erkenntnis lässt sich verallgemeinern und erweitern:

> Jede kombinatorische Schaltung lässt sich ausschliesslich mit NAND-Elementen oder ausschliesslich mit NOR-Elementen realisieren.

Wo Negationen nötig sind, nimmt man dabei einfach ein NAND bzw. NOR, welches alle Eingänge zusammengehängt hat.

Die Lösung des Problems "Chemielager" haben wir nun vorerst intuitiv durch logisches Ueberlegen gefunden.
Man kann aber auch "stur" mathematisch (=schaltalgebraisch) vorgehen.
Dies soll hier auch übungshalber tun:

13.10 Die ODER-Normalform von Boolschen Gleichungen

Im Beispiel mit der Rauchmeldeanlage im Chemielager haben wir aus der Wahrheitstabelle

M3	M2	M1	A
0	0	0	0
0	0	1	0
0	1	0	0
0	**1**	**1**	**1**
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**

die Bedingungen für den Ausgang A beim Ansprechen der Rauchmelder M abgeleitet:
"Der Ausgang ist dann = 1, wenn
entweder
M1 und M2
oder
M1 und M3
oder
M2 und M3
oder
alle 3 Melder = 1 sind." **Als Boolsche Gleichung:**

$$A = (M1 \wedge M2) \vee (M1 \wedge M3) \vee (M2 \wedge M3) \vee (M1 \wedge M2 \wedge M3)$$

In den ersten drei Klammerausdrücken (Konjunktionen, = UND-Verknüpfungen) kommen **nicht** alle Variablen vor. Dies war nur möglich, weil wir die invertierten Zustände der Melder (also 0) aufgrund der Augabenstellung nicht verwenden mussten. Dies war Zufall und ist **nicht der allgemeine Fall!** Damit man alle Fälle abdeckt, muss man für jeden Fall, der zu A=1 führt, die so genannten **Voll**konjunktionen bilden.

Definition der **Vollkonjunktion**:
- Konjunktionen sind UND-Verknüpfungen
- "Voll" bedeutet, dass alle Eingangsvariablen vorkommen müssen.

In unserem Beispiel hiesse das also, ich muss die Bedingungen konkreter fassen, es müssen in jedem betrachteten Fall, der zu A=1 führt, **alle** Eingänge (M1 bis M3) vorkommen, sei es negiert oder normal. Sprachlich exakt umgesetzt hiesse das:
"Der Ausgang ist dann = 1, wenn
entweder
M1 und M2 **und nicht M3**
oder
M1 und M3 **und nicht M2**
oder
M2 und M3 **und nicht M1**
oder
alle 3 Melder = 1 sind." **Als Boolsche Gleichung:**

$$A = (M1 \wedge M2 \wedge \overline{M3}) \vee (M1 \wedge M3 \wedge \overline{M2}) \vee (M2 \wedge M3 \wedge \overline{M1}) \vee (M1 \wedge M2 \wedge M3)$$

Nun kommen in jeder Konjunktion **alle Eingangsvariablen** vor.
Man nennt sie nun Vollkonjunktionen.
Die Oder-Verknüpfung der Vollkonjunktionen nennt man **ODER-Normalform.**

Eine **ODER-Normalform** enhält **ODER-Verknüpfungen von** jenen **Vollkonjuktionen,** die im **Ausgang zu einer 1 führen.**

Die Vereinfachung auch dieser Boolschen Gleichung führt natürlich zum gleichen, bereits früher gefundenen Resultat.

13.11 KV-Diagramme: Grafische Vereinfachung boolscher Ausdrücke

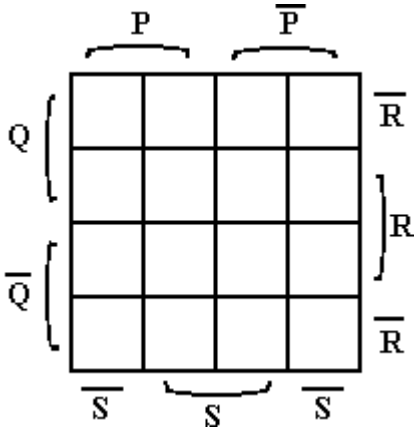
Das Vereinfachen boolscher Ausdrücke kann recht kompliziert werden, vor allem, wenn z.B. 3 oder 4 Variablen (Eingänge) vorhanden sind, und wir uns das Operieren mit den ODER- sowie UND-Zeichen nicht gewohnt sind. Es gibt deshalb eine grafische Methode mit den sogenannten KV-Diagrammen (Abkürzungen der Erfinder des Diagramms, Karnaugh und Veitch).

Als Beispiel nehmen wir eine boolsche Gleichung mit den 4 (Eingangs-)Variablen P, Q, R, S.
Sie muss in der ODER-Normalform vorliegen:

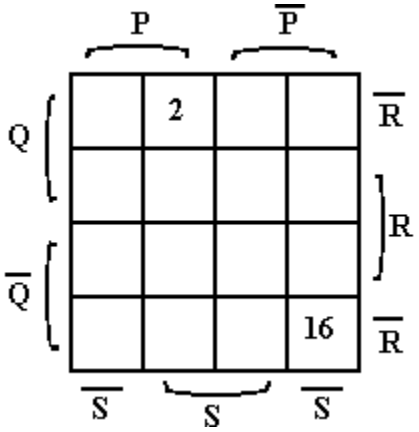
$$A = (\overline{P} \wedge \overline{Q} \wedge \overline{R} \wedge \overline{S}) \vee (P \wedge \overline{Q} \wedge \overline{R} \wedge \overline{S}) \vee (\overline{P} \wedge Q \wedge \overline{R} \wedge \overline{S}) \vee (P \wedge Q \wedge \overline{R} \wedge \overline{S}) \vee (P \wedge \overline{Q} \wedge \overline{R} \wedge S) \vee (P \wedge Q \wedge \overline{R} \wedge S)$$

Vorgehen:

1. Jeder der 16 möglichen Vollkonjunktionen aus 4 Variablen ist ein Feld im Diagramm zugeordnet (Bei 3 Variablen hätte es nur 8 Felder). Jedes Feld hat "Koordinaten" aus 4 Variablen, entweder normal oder negiert.
Die Felder sind nach folgender Art angeordnet:



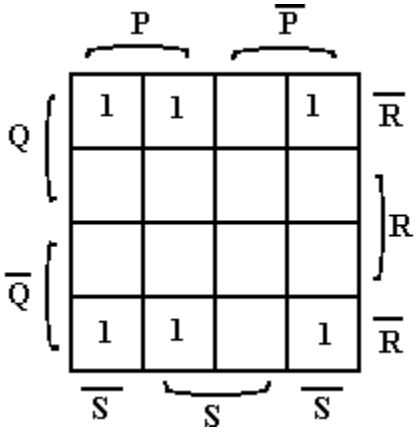
leer



mit Feldnummern

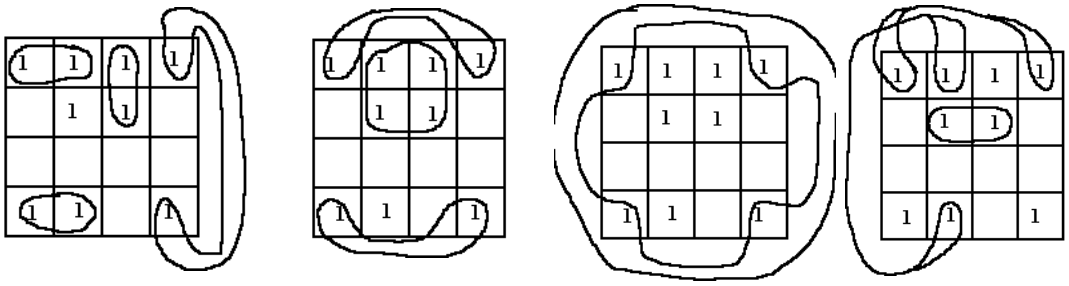
Bsp rechts:
Der erste Klammerausdruck entspricht dem Feld Nr. 16, weil dort alle Variablen negiert sind.
Der letzte Klammerausdruck entspricht dem Feld Nr. 2, wo nur R negiert ist.

2. Jede Vollkonjunktion, die im Resultat (Ausgang) zu einer 1 führt (also jene, die in der Gleichung vorkommen), bekommt im Diagramm eine 1 an der entsprechenden Koordinate.



3. Als nächstes werden so genannt **benachbarte Felder** gesucht und zu **Paketen** zusammengefasst.

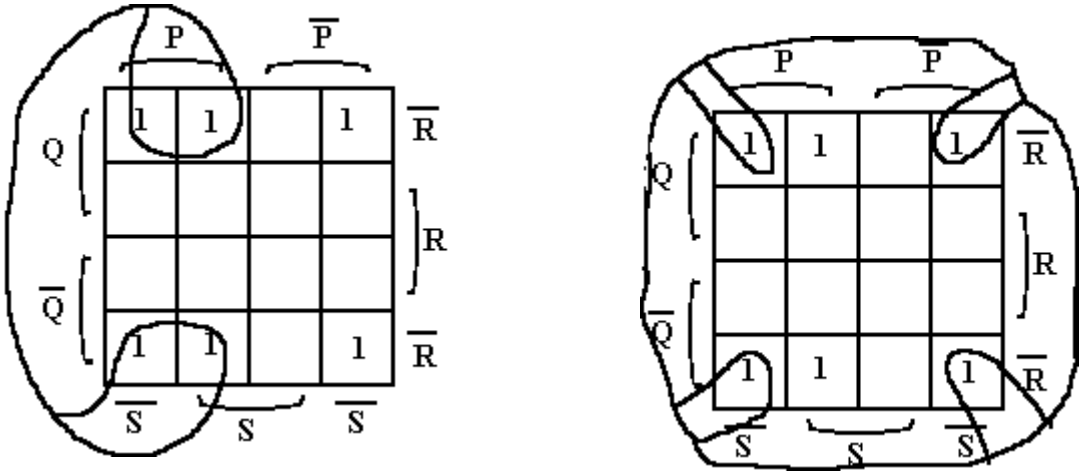
Als benachbart gelten beispielsweise jeweils die eingezonten Felder (hat nichts mit unserem Beispiel zu tun):



Es gilt:

- Es sollen möglichst viele Einsen zusammengefasst werden
- Es dürfen nur 2er, 4er, 8er oder 16er-Pakete gebildet werden
- Es muss möglichst wenige und gleichzeitig möglichst grosse Pakete geben
- Die Pakete dürfen sich überlappen.

In unserem Beispiel finden wir im besten Fall zwei Pakete, welche benachbarte Feldern beinhalten und jeweils 4 Felder (=4 Vollkonjunktionen) enthalten:



Die beiden Felder sind hier noch in zwei Diagrammen gezeichnet, können aber im gleichen Diagramm gezeichnet werden.

4. Jene Variablen, die in einem Paket normal **und** negiert vorkommen, fallen weg.
Die verbliebenen Variablen bilden eine vereinfachte Konjunktion, die am Schluss mit den andern Konjunktionen aus den andern Felder ODER-verknüpft wird.

Im linken Diagramm kommen im Paket die Variablen Q und S normal **und** negiert vor. Sie verschwinden deshalb.
Gleich sind P und das **negierte** R. Sie bleiben.
Dies ergibt den Ausdruck

$(P \wedge \overline{R})$

Im rechten Diagramm sind P und Q in beiden Formen vorhanden, während das negierte R und das negierte S alleine vorkommen und bleiben.
Der entsprechende Ausdruck heisst

$(\overline{R} \wedge \overline{S})$

Diese beiden verbliebenen Konjunktionen (UND-Ausdrücke) werden nun ODER-verknüpft:

$A = (P \wedge \overline{R}) \vee (\overline{R} \wedge \overline{S})$ Hier kann man noch das negierte R ausklammern:

$A = \overline{R} \wedge (P \vee \overline{S})$ **Dies ist das Ergebnis der Vereinfachung.**

Wohlgemerkt:
Diese Form erfüllt die gleiche Wahrheitstabelle wie die ursprüngliche Gleichung.
Eine doch beachtliche Vereinfachung. Die Variable **Q** kommt gar nicht mehr vor, das heisst, der Ausgang ist **unabhängig von Q**!

Anwendung / Uebung zu KV-Diagrammen

Die Aufgaben mit der Rauchmeldeanlage im Chemielager soll mit KV-Diagrammen gelöst werden.

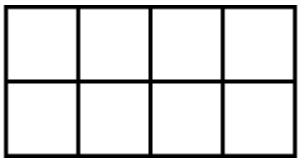
Zur Erinnerung: Die Schaltung muss folgende Wahrheitstabelle erfüllen:

M3	**M2**	**M1**	**A**
0	0	0	0
0	0	1	0
0	1	0	0
0	**1**	**1**	**1**
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**

Dies führt zu folgender ODER-Normalform:

$$A = (M1 \wedge M2 \wedge \overline{M3}) \vee (M1 \wedge M3 \wedge \overline{M2}) \vee (M2 \wedge M3 \wedge \overline{M1}) \vee (M1 \wedge M2 \wedge M3)$$

Wir haben 3 Variablen, das entsprechende KV-Diagramm hat deshalb nur 8 Felder:



Es lassen sich ..... Pakete machen.

Paket Nr.	Normal **und** negiert vorhandene Variablen -> fallen weg	Einfach vorhandene Variablen -> bleiben	Verbleibende Konjunktion

Die verbleibenden Konjunktionen werden ODER-verknüpft, die vereinfachte Boolsche Gleichung lautet:

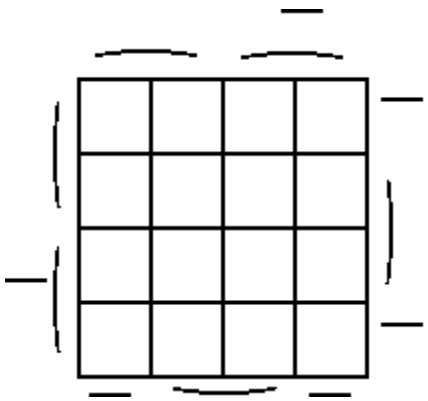
Schaltungssynthese: Weitere Uebung zu KV-Diagrammen

Gesucht ist eine Schaltung, welche die folgende Wahrheitstabelle erfüllt:

Fall	D	C	B	A	Z
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	0
6	0	1	0	1	0
7	0	1	1	0	0
8	0	1	1	1	0
9	1	0	0	0	0
10	1	0	0	1	1
11	1	0	1	0	0
12	1	0	1	1	1
13	1	1	0	0	0
14	1	1	0	1	0
15	1	1	1	0	0
16	1	1	1	1	0

Aus der Wahrheitstabelle wird die ODER-Normalform abgeleitet:

Diese Form kann am einfachsten mit dem KV-Diagramm vereinfacht werden. Wir brauchen 2^4, also 16 Felder.



Paket Nr.	Normal und negiert vorhandene Variablen -> fallen weg	Einfach vorhandene Variablen -> bleiben	Verbleibende Konjunktion

Dies ergibt als Boolsche Gleichung:

Die Schaltung muss dann folgendermassen aussehen:

13.12 Spezielle Kombinatorische Schaltungen

Viele kombinatorische Schaltungen sind wegen ihrer speziellen Funktionen seit Jahrzehnten als separate Bausteine (Chips, ICs, integrierte Schaltungen, "Käfer") im Handel.

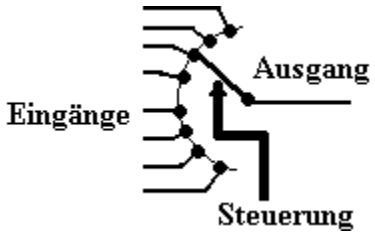
13.12.1 Datenselektor, Multiplexer, Demultiplexer

Ein **Datenselektor** (Datenwähler) ist ein Baustein, der je nach Zustand der Steuerleitungen **einen** von mehreren Eingängen auf einen Ausgang durchschaltet.

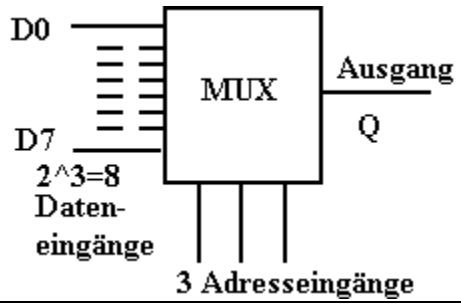
Multiplexer

Ein Datenselektor, der **zeitabhängig** einen von mehreren Eingängen nacheinander auf einen Ausgang durchschaltet, heisst **Multiplexer**. Beispiel: ein 1-aus-8-Datenselektor-Multiplexer:

Funktion



Schema



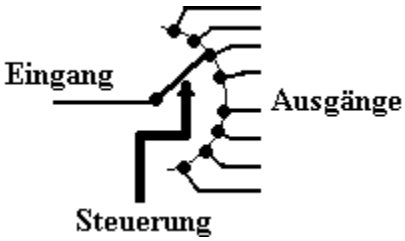
Wahrheitstabelle:
(für die aktuellen Eingänge nehmen wir nicht 0 oder 1, sondern die Variablen D0 bis D7)

A2	A1	A0	Q
0	0	0	**D0**
0	0	1	**D1**
0	1	0	**D2**
0	1	1	**D3**
1	0	0	**D4**
1	0	1	**D5**
1	1	0	**D6**
1	1	1	**D7**

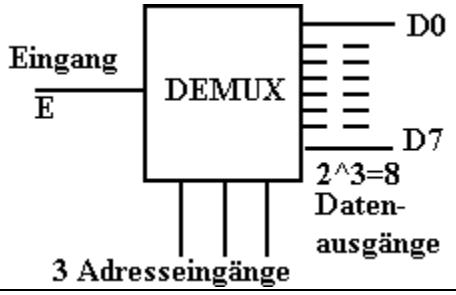
Demultiplexer (auch **Adressdecoder** genannt)

Die umgekehrte Arbeitsweise hat ein Demultiplexer. Er schaltet je nach Zustand der Steuerleitungen einen Eingang auf **einen** von mehreren möglichen Ausgängen durch.

Funktion



Schema



Wahrheitstabelle (aktueller Zustand des Eingangs = Variable **E**)

A2	A1	A0	D0	D1	D2	D3	D4	D5	D6	D7	
0	0	0	**E**								**Restliche Felder je nach**
0	0	1		**E**							**Gestaltung der Hardware**
0	1	0			**E**						**0 oder 1**
0	1	1				**E**					
1	0	0					**E**				
1	0	1						**E**			
1	1	0							**E**		

1	1	1								**E**

13.12.2 Decoder, Encoder, Code-Wandler, Code-Umsetzer

Alle obigen Bezeichnungen bedeuten technisch das Gleiche.
Nämlich einen Baustein, der aus einem Code einen andern macht.
Man trifft je nach dem genauen Einsatz des Wandlers manchmal andere Bezeichnungen:

- Als **En**coder bezeichnet man einen Baustein, der aus einer Information einen Code **erstellt**.

- Ein **Decoder** macht aus einem für uns nicht aussagekräftigen Code einen darstellbaren, verständlichen, brauchbaren Code. Dies ist die meistgebrauchte Bezeichnung

Beispiel:

BCD-zu-7-Segment-Decoder
Dieser Decoder macht aus einem BCD-Code den Code, der zur Ansteuerung einer 7-Segment-Anzeige nötig ist.

Die im BCD-Code verbotenen Kombinationen (ab 1010) können je nach innerem Aufbau des Decoders eine Fehlanzeige verursachen, oder die Anzeige wird unterdrückt.

Bezeichnung der Segmente einer 7-Segment-Anzeige und die definierten Ziffernformen:

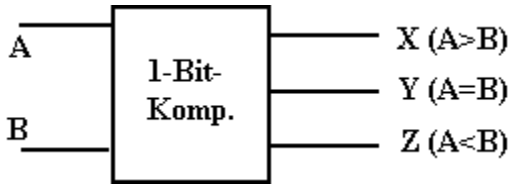
Wahrheitstabelle für 7 Ausgänge (2 Beispiele, Rest selber ausfüllen):

Eingänge im BCD-Code				Ausgänge Segmente						
D	C	B	A	a	b	c	d	e	f	g
0	0	0	0							
0	0	0	1	0	**1**	**1**	0	0	0	0
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0	**1**	0	**1**	**1**	**1**	**1**	**1**
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	-	-	-	-	-	-	-

13.12.3 Komparator

Ein Komparator ist ein **Vergleicher**.
Er vergleicht zwei binäre Ausdrücke A und B, welche auf die Eingänge wirken.
An den Ausgängen meldet er A>B, A<B oder A=B.
Man kann dazu drei Ausgänge verwenden (für jeden möglichen Fall einen),
oder zwei Ausgänge mit insgesamt 4 Ausgangskombinationen (eine wäre dann zuviel).

Mögliches Symbol eines 1-Bit-Komparators:

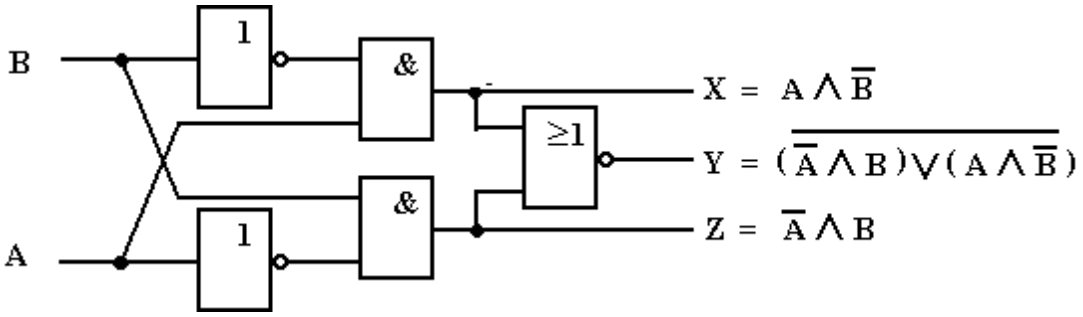


Wahrheitstabelle:

B	A	X (=1 falls A>B)	Y (=1 falls A=B)	Z (=1 falls A<B)	ergibt direkt Boolsche Gleichung:
0	0	0	**1**	0	Y, siehe Gleichung unten
0	1	**1**	0	0	$X = A \wedge \overline{B}$
1	0	0	0	**1**	$Z = \overline{A} \wedge B$
1	1	0	**1**	0	Y, siehe Gleichung unten

$$Y = (\overline{A} \wedge \overline{B}) \vee (A \wedge B) = \overline{(\overline{A} \wedge B) \vee (A \wedge \overline{B})}$$

Mögliche Realisation des Komparators:

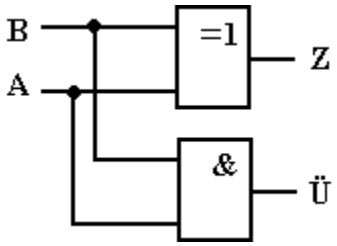


13.12.4 Addierer

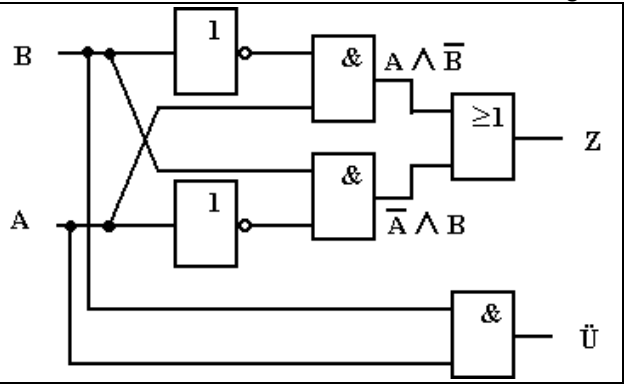
Das "schriftliche" Addieren von Dualzahlen haben wir kennengelernt. Man kann von rechts her stellenweise vorgehen, zwei Zahlen zusammenzählen und ev. einen Übertrag ("Behalte 1") bilden, der auf die nächste Stelle übertragen wird. In diesem Fall (Übertrag) muss man im nächsten Schritt drei Zahlen addieren. Betrachtet man den einfachsten Fall von zwei 1-Bit-Zahlen, so gelten bekanntlich die Rechenregeln:
0+0= 0
0+1= 1
1+0= 1
1+1=10 Im letzten Fall entsteht bereits ein Übertrag auf die nächste Stelle.
Eine Additionsschaltung muss diesen Übertrag also liefern können.
Gesucht ist also eine kombinatorische Schaltung, die zwei Summanden A und B als Eingang hat, und das Resultat Z und den Übertrag Ü ausgibt.
Dies ergibt folgende Wahrheitstabelle:

A	**B**	**Z**	**Ü**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Für Z ist das eine EXOR-Verknüpfung der Eingänge, für Ü eine AND-Verknüpfung.
Dies kann mit der folgenden Schaltung verwirklicht werden:
Sie heisst **Halbaddierer**, weil sie nur **zwei** Dualzahlen addieren kann und keine Überträge.

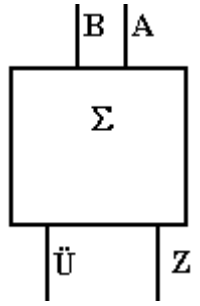


Falls das EXOR-Glied (Antivalenz) mit den Grundfunktion en aufgebaut wird, ergibt sich die Schaltung rechts:



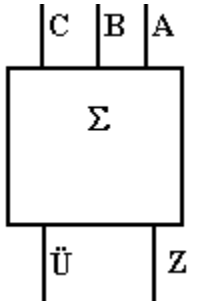
Für den Aufbau eines Addierwerks benötigt man Schaltungen, die **drei** Dualzahlen, nämlich die eigentlichen **Summanden** und einen allfälligen **Übertrag** zusammenzählen kann. Eine solche Schaltung heisst dann **Volladdierer**.

Halbaddierer



Ein Halbaddierer muss "nur" 2 Summanden addieren können.
Er kann aber nur an der ersten Stelle eingesetzt werden, wo kein Uebertrag anfällt.
Der Ausgang Ü wird auch **Carry Out** genannt (**CO**).

Volladdierer



Ein Volladdierer kann drei Summanden verarbeiten, er hat einen dritten Eingang C.

Dieser wird auch **Carry In** genannt (**CI**).

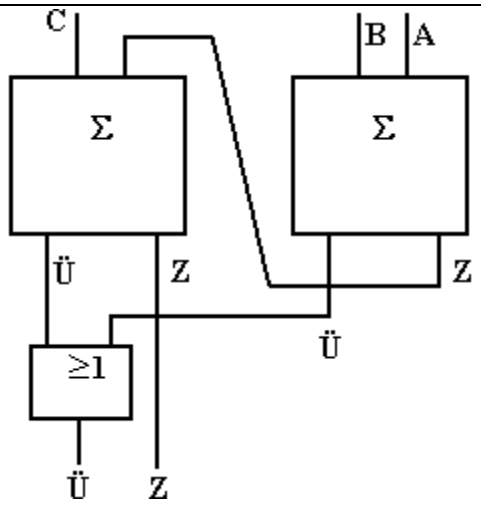
In der Wahrheitstabelle ergeben sich demnach 8 Fälle, aber immer noch nur zwei Ausgänge:

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

Für die Ausgänge könnte nun nach bekannter Methode die Boolsche Gleichung und daraus eine Schaltung aus Grundfunktionen gesucht werden. Das Ergebnis daraus ist bereits eine recht komplexe Schaltung, nämlich aus 7 AND-Gattern, 2 OR-Gattern und 3 Invertern.

Man kann jedoch auch aus zwei Halbaddierern und einem OR-Glied einen Volladdierer machen:

Diese Schaltung erfüllt die geforderte Wahrheitstabelle für einen Volladdierer.



Prinzipieller Aufbau eines Addierers für mehrstellige Zahlen

Meist haben die zu addierenden Zahlen nicht nur eine Stelle.
Für die Addition mehrstelliger Zahlen benötigt man mehrere Volladdierer.
An der ersten Stelle genügt ein Halbaddierer, weil dort noch keine Ueberträge auftreten können. Ein allfälliger Uebertrag des letzten Volladdierers ergibt eine zusätzliche Stelle.
Die Addition zweier 4-Bit-Zahlen ergibt demnach im Maximum eine 5-Bit-Zahl.
(Extremfall: 1111+1111=1'1110)
Im Bild sehen wir einen **Paralleladdierer**: Er heisst so, weil jede Stelle der Summanden gleichzeitig, also parallel, addiert wird.