Digitaltechnik

Levin Baumann

30. Juni 2020

1 Begriffe

digital	analog
wertediskret	wertekontinuierlich
⇒ zwischen zwei benachbarten gibt es <u>keine</u> zwischenwerte	⇒ zwischen zwei beliebigen gibt es unendlich viele Zwischenwerte
⇒ endlich oder auch unendlich (aber abzählbar viele)	⇒ Immer unendlich (und sogar überabzählbar) viele Werte
meist zeitdiskret („getaktet“)	zeitkontinuierlich
⇒ unsere heutigen Rechner arbeiten digital	⇒ unsere Welt „arbeitet“ analog

2 Codierung

Definition 1

Codierung ist die Darstellung von Informationen mit einem „Alphabet“.

Definition 2

Alphabet ist eine <u>endliche</u> Menge von Symbolen

Anmerkung: Codierung ist immer mit Digitalisierung verbunden.

2.1 Arten von Codierungen

- Zeichencodierung
- Zahlencodierung (Text-, Bildbearbeitung,...)
- Verschlüssellung (*Achtung:* Unterschied zu anderen Codierungen: Aus der codierten verschlüsselten Info sollen die meisten Menschen nicht auf die Ursprungsinfo schließen können)
- Signalcodierung

2.2 Codierungen

2.2.1 Zeichencodierung

- ASCII: Alphabet besteht aus (ganzen) Zahlen von 0 bis 127
- ISO8859-x: Alphabet besteht aus 0...255

  -x: verschiedene Sprachräume

  -1: westeuropäisch

  -5: kyrillisch

  -7: griechisch

  -15: westeuropäisch inkl. €
- Unicode: Anspruch, alle (derzeitigen, künftigen, ehemaligen) Schriftsprachen abzudecken, auch Phantasiesprachen wie Klingonisch oder Elbisch

  UTF-8: 256 Zahlenwerte

  UTF-16: 65536 Zahlenwerte

  → Zeichen werden als variabel lange Symbolfolgen dargestellt. Gesetztes MSB zeigt an, dass mindestens ein Folgesymbol folgt.

2.2.2 Zahlencodierung

I. Abzählsysteme

Fingerabzählsysteme

Alphabet = {Finger}

Symbolwert (Finger) = 1

$\ominus$ stark beschränkter Wertebereich von 0 bis 10

$\ominus$ bis auf weiteres nur nicht-negative ganze Zahlen

$\oplus$ extrem einfach und verständlich

$\oplus$ extrem einfache Addition und Subtraktion

$\ominus$ Multiplikation und Divisionmit erhöhtem Aufwand

$\oplus$ Immer verfügbar/„zur Hand“

Strichliste (einfache)

Alphabet = {I}

Wert (I) = 1

$\oplus$ unbeschränkter Wertebereich

$\oplus$ Addition weiter einfach, Subtraktion braucht man eine Entfernungsmöglichkeit/Entwertungsmöglichkeit

$\ominus$ Hilfsmittel (Schreibwerkzeug) notwendig

$\ominus$ unübersichtlich darstellbarer Wertebereich bis etwa 10 (10 Symbole notiert)

erweiterte Strichliste („Lattenzaunsystem “)

Alphabet = I, ~~IIII~~

Wert (I) = 1

Wert (~~IIII~~) = 5

Regeln:Symbole müssen nach Wertigkeit sortiert notiert werden. Fünf einfache Striche müssen immer zu einem „Kombisymbol“ zusammengefasst werden.

$\oplus/\ominus$ Addition etwas erschwert durch Sortieren und Zusammenfassen; Subtraktion zusätzlich erschwert durch Auflösen des Kombisymbols

$\oplus/\ominus$ etwas komplexer durch zweites Symbol und Sortierregel

$\oplus/\ominus$ übersichtlich darstellbarer Wert bis etwa 50 erweitert

römisches Zahlensystem

Alphabet = {I, V, X, L, C, D, M}

Wert = {1, 5, 10, 50, 100, 500, 1000}

Zusatzregel: Symbol kleinerer Wertigkeit darf vor einem Symbol größerer Wertigkeit notiert werden Sein Symbolwert wird dann subtrahiert, statt addiert. Mehr als drei gleiche Symbole sind nicht hintereinander erlaubt

$\oplus/\ominus$ übersichtlich darstellbarer Wertebereich bis etwa 10.000 (?)

$\ominus$ beschränkter Wertebereich bis ¡ 4000

$\ominus$ recht hohe Komplexität, relativ geringe Verständlichkeit

$\ominus$ extrem komplizierte Addition und Subtraktion

II. Stellenwertsysteme (SWS)

Dezimalsystem

Alphabet = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Symbole werden auch Ziffern genannt

Zahl ist notiert als Folge von Ziffern: z.B. 4711

Ziffernwert = Symbolwert

Ziffernwert (4) = 4

Zahlenwert = Summe (Ziffernwert * Stellenwert)

Stellenwert (rechte Ziffer) = 1

Stellenwert wächst mit jeder Stelle nach links um Faktor 10

$$Wert(Z_{n-1} \dots Z_0) = \sum_{i=0}^{n-1} |Z_i| \cdot 10^i$$

SWS zur Basis b

Alphabet enthält b Ziffern

Alphabet beginnt bei Ziffer „0“ und endet bei Ziffer „b-1“

$$Wert(Z_{n-1} \dots Z_0) = \sum_{i=0}^{n-1} |Z_i| \cdot b^i$$

b_{EIN} mit $b > 1$
Anmerkung: $b = 1$ nicht sinnvoll, da im SWS zur Basis nur die 0 dargestelt werden könnte.

⊕/⊖ gewisses „Erstverständnis “Lernaufwand ist nötig

⊕/⊖ es gibt für jede Grundrechenart einVerfahren mit etwas erhöhter Komplexität, welches aber nach erstmaligem Lernaufwand doch relativ problemfrei realisierbar ist

⊕/⊖ unbeschränkter Wertebereich

⊕/⊖ Wertebereich bis etwa $b^1 0$ ist übersichtlich darstelbar

Umrechnung

- Von Basis b nach Basis 10?
  ⇒ Werteformel
- Von Basis 10 nach Basis b?
  ⇒ Werteformel umgekehrt
  ⇒ Ganzzahldivision

$$\begin{aligned}
42_{10} &= 101010 \\
43 : 2 &= 21R0 = Z_0 \\
21 : 2 &= 10R1 = Z_1 \\
10 : 2 &= 5R0 = Z_2 \\
5 : 2 &= 2R1 = Z_3 \\
2 : 2 &= 1R0 = Z_4 \\
1 : 2 &= 0R1 = Z_5
\end{aligned} \tag{1}$$

Hinweis: Führende „0 “ können bei Zahlen im SWS beliebig hinzugefügt oder weggelassen werden. allg: Umrechnung von Basis b_1 nach Basis b_2

- meist: von Basis b_1 nach $b_Z = 10$ dann von b_Z nach b_2
- theoretisch: Ganzzahldivision durch Zielbasis b_2, aber ausgeführt im SWS zur Ausgangsbasis ⇒ Nicht praktikabel

Direkte Umrechnung:
Falls $b_1 = {b_2}^n$, dann entsprechen n Ziffern zur Basis b_2 einer Ziffer zur Basis b_1 und es ist eine ziffern(block)weise Umrechnung.

Bsp.: $b_1 = 2$ und $b_2 = 16$
$\Rightarrow$ 4 Ziffern zur Basis 2 entsprechen einer Ziffer zur Basis 16.

Gängige Basen im SWS
$b = 10$ „Dezimalsystem" $\Rightarrow$ Mensch
$b = 2$ „Binärsystem"„Dualsystem" $\Rightarrow$ Digitalrechner
$b = 16$ „Hexadezimalsystem " $\Rightarrow$ Computernahe Darstellung von Zahlen

- weniger Stellen
- einfache Umrechnung

$b = 8$ „Oktalsystem" $\Rightarrow$ Computernahe Darstellung von Zahlen
$b = 16$ vs. $b = 8$ nur „normale" Zahlen notwendig

Hexadezimalsystem

Wert	Ziffer	Binär
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	10	1010
11	11	1011
12	12	1100
13	13	1101
14	14	1110
15	15	1111

BSP: $ABEF_{16}$

1010 | 1011 | 1110 | 1111

$$\begin{array}{c|c|c|c|c} 0001 & 1001 & 0001 & 1100 & 0101_2 \\ 1 & 9 & 1 & C & 5 \end{array} \quad = \quad 191C5_{16}$$

Falls $b_1^n = b_2^m$, dann entspricht ein Ziffernblock von n Ziffern zur Basis b_1 einem Ziffernblock von m Ziffern zur Basis b_2.

$$\begin{array}{lll} 2^{3n} = 2^{4m} & \Rightarrow & n = 4 \& m = 3 \\ & \Rightarrow & \text{4 Ziffern zur Basis 8 entrsprechen} \\ & & \text{3 Ziffern zur Basis 16 entrsprechen} \\ & & \text{Problem: Tabelle hat } 2^{12} \text{ Zeilen} \end{array}$$

Einschränkungen aufheben

auch negative Zahlen!

2.3 Darstellung negativer Zahlen

Vorzeichen und Betrag

2.3.1 1er-Komplement

(ab jetzt immer zur Basis 2)
Alle Bits werden invertiert: $0 \to 1$; $1 \to 0$
Aber vorher: bei positiven Zahlen mindestens eine führende Null.
Außerdem: alle Zahlen auf gleiche Länge!

$$\begin{array}{rr|ccccccccc} & 42 & & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ - & 37 & & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ & & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & {\scriptstyle 1} & & \\ \hline & & \cancel{1} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} \quad \text{(9. Stelle wird ignoriert)}$$

Nachteile 1er-Komplement

- Manchmal (leicht) falsche ERgebnisse
- Zwei verschiedene Darstellungen der „0“
  $+0 \hat{=} 0000 \xrightarrow{e.K.} 1111 \hat{=} -0$

Wertebereich 8bit-1-IC-Zahlen:

$$01111111 = 127 - 127 = 10000000 \xrightarrow{eK} 01111111 = 127$$

⇒ nur 255 (statt $256 = 2^8$) verschiedene Zahlenwerte mit 8 Bit darstellbar

2.3.2 2er Komplement

Bildung: Wie 1er-Komplement, also Stellenzahl festlegen, Ziffern invertieren ($0 \to 1$; $1 \to 0$)
Zusätzlich: +1 addieren!

⊕ Ergebnis immer korrekt!

2.4 Darstellung nicht-ganzer (aber vorläufig nicht-negativer) Zahlen

2.4.1 Bruch: Darstellung mit Zähler & Nenner

$$\frac{1}{2} = \frac{2}{4}\frac{1}{4}$$

⊖ Es gibt unendlich viele Darstellungen jeder Zahl als Bruch. Lösungsmöglichkeit: nur gekürzte Darstellung.
⇒ Normalerweise keine Darstellung von Zahlen als Bruch im PC (Ausnahme: algebraisches Lösen von Gleichungssystemen)

2.4.2 Festkommazahlen (FKZ)

Alphabet mit Ziffern wird erweitert um „Komma “(„, “)
⇒ In der Symbolfolge ist maximal ein komma erlaubt.
Bsp: Zahl mit n Vor- und Nachkommastellen

$$Z_{n-1}Z_{n-2}\dots Z_2Z_1Z_0, Z_{-1}Z_{-2}\dots Z_{-m+1}Z_{-m+2}$$

$$\sum_{i=0}^{n-1} |z_i| \cdot b^i$$

$$\begin{aligned} z.B. \quad 110,011_2 &= 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 \\ &= 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} \\ &= 2 + 4 + \frac{1}{4} + \frac{1}{8} = 6 + 0,25 + 0,125 = 6,375 \end{aligned}$$

Umrechnung von Basis 10 nach Basis b?
→ Werteformel
$z.B. 6,375_{10} = ?,?_2$
Aufteilung in Vor- und Nachkommateil:
Vorkommateil über Ganzzahldivision
$6_{10} = 110_2$
Nachkommateil: Multiplikation mit Zielbasis und Aufteilen in Vor- und Nachkommateil
Vorkommateil ist die erste bzw. nächste Nachkommastelle

$$\begin{array}{rcrl} 0,375 & \cdot 2 & = & 0,75 \\ 0,75 & \cdot 2 & = & 1,5 \\ 0,5 & \cdot 2 & = & 1,0 \\ 0,0 & \cdot 2 & = & 0,0 \end{array}$$

$$0,375_{10} = 0,01100_2$$

2.4.3 Gleitkommazahlen (GKZ)(auch Fließkommazahlen)

$$\text{Wert} = \text{Mantisse} \cdot \text{Basis}^{\text{Exponent}}$$

Mantisse	Festkommazahl
Exponent	ganze Zahl, steht für die Verschiebung des Kommas bei der Mantisse
Basis b	beliebig (wie bei anderen Zahlen im SWS), aber b muss der für die Mantisse verwendeten Basis entsprechen

$$\underbrace{1,0}_{\text{Mantisse}} \cdot 10^{\overbrace{0}^{\text{Exponent}}} = 10,0 * 10^{-1} = 100,0 \cdot 10^{-2} = 0,1 \cdot 10^{1} \ldots$$

Hinweis

Es gibt unendlich viele Darstellungen jedes Zahlenwertes als GKZ

Abhilfe: normalisierte Darstellung

a) genau eine Vorkommastelle (Vkst), diese muss $\neq 0$ sein!

b) keine Vkst, erste Nkst $\neq 0$

Variante a) ist gängiger. In der Vorlesung wird diese üblicherweise verwendet.

negative GKZ ⇒ negative Mantisse

Darstellung der Mantisse: per Vorzeichen und Betrag (nicht Zweier-Komplement), um die Fallunterscheidungen beim Größenvergleich zwischen Vorzeichen- und normaler Stelle zu ersparen[1] und vor allem bei der Kommaverschiebung die Fallunterscheidung mit „0“(bei positiven) oder „1“(bei negativen) aufzufüllen zu ersparen.

Vorzeichen des Exponenten gibt die Richtung der Kommaverschiebung an.
Größenvergleich:

1. Vorzeichen Mantisse:
   unterschiedlich, dann gehört das positive Vorzeichen zur größeren
   gleich, dann Vergleich der Beträge und spätere Fallunterscheidung
2. Vergleich Betrag
   größerer Exponent gehört zum größeren Betrag
   ⇒ per Bias-Darstellung einfacher bitweiser Vergleich ohne Fallunterscheidung
   bei gleichem Exponent: Größenvergleich der Mantisse (bitweise)
3. Fallunterscheidung abhängig vom Vorzeichen
   positiv: größerer Betrag → größere Zahl
   negativ: größerer Betrag → kleinere Zahl

Darstellung des Exponenten

Exponent wird „künstlich“nicht-negativ gemacht, indem ein Bias aufaddiert wird.

$$\text{Exp}_{\text{gespeichert}} = \text{Exp}_{\text{real}} + \text{Bias}$$

Bias wird forher festgelegt, z.B. bei 8-bit-Exp: Bias=127.

Im Computer: GKZ zur Basis 2

normalisierte Darstellung ⇒ Vkst ist immer „1“.
⇒ diese „1“muss nicht explizit gespeichert werden
⇒ „Hidden Bit“⇒ dieses eine Bit wird genutzt, um eine Nkst mehr speichern zu können

Problem: Darstellung der „0“

→ für die „0“gibt es keine normalisierte Darstellung
→ mit „Hidden Bit“lässt sich der Zahlenwert „0“nie darstellen.

Abhilfe: kleinster Exponent (Bitmuster „0…0) steht für denormalisierte GKZ ohne Hidden Bit. Falls dann auch alle Mantissenbits „0“sind, handelt es sich um die Darstellung der „0“.
größter Exponent (Bitmuster „1…1“) steht für eine Zahl, welche nicht im Wertebereich der gewählten GKZ darstellbar ist.
falls Mantisse „$0\ldots0$“⇒ Zahl ist $\pm\infty$ (abhöngig von Mantissen-Vorzeichen)
falls Mantisse $\neq$ „$0\ldots0$“⇒ Zahl ist nicht als reele Zahl darstellbar (z.B. $\sqrt{-2}$) ⇒ NaN „Not a Number“.

Wertebereich für Zahlen mit 16 Bit:

[1] Ist dann trotzdem insgesamt für die Zahl notwendig, nicht aber für die einzelnen Stellen.

Genauigkeit	Speicher (bit)	Vz (bit)	Mantisse (bit)	Exponent (bit)	Bias
Single	32	1	23	8	127
Double	64	1	52	11	1023
Half	16	1	10	5	15

Tabelle 1: Gleitkommazahl gemäß IEEE 754

- nicht negative ganze Zahlen: $0 \dots 65535$
- 2er-Komplement (ganze Zahlen): $-32768 \dots +32767$
- GKZ half precision

Größte darstellbare Zahl:

$$\text{positiv} \quad \underbrace{0}_{\text{VZ}} \; \underbrace{11110}_{\text{Exponent}} \; \underbrace{1111111111}_{\text{Mantisse}} \qquad \boxed{1,} \longleftarrow \text{Hidden Bit}$$

$$\begin{aligned} Exp_{gesp} &= 2+4+8+16 \\ &= 30 \\ Exp_{real} &= Exp_{gesp} - Bias \quad 2 - \frac{1}{2^{10}} = 2 - 2^{-10} \\ &= 30 - 15 \\ &= 15 \end{aligned}$$

$$\text{Wert} = (2 - 2^{-10}) \cdot 2^{15} = 2^{16} - 2^5 = 65536 - 32 = 65504$$

Kleinste darstellbare Zahl:

$$1\ 11110\ 1111111111 \Rightarrow -65504$$

Kleinste Zahl >0:
normalisierte Darstellung

$$\text{negativ} \quad 0 \; \underbrace{00001}_{Exp_{gesp}=1} \; \underbrace{0000000000}_{\text{Mantisse}=1} \qquad \boxed{1,}$$

$$\begin{aligned} Exp_{real} &= 1 - 15 = -14 \\ Wert &= 1 \cdot 2^{-14} \end{aligned}$$

Half-precision:

Vz	Exp	Mantisse	Bias = 15
15bit	5bitr	10bit	

größte Zahl:	0 11110 1111111111
kleinste Zahl:	0 11110 1111111111

Kleinster Betrag einer Zahl > 0:

normalisierte:	0 00001 0000000000
denormalisierte	0 00000 0000000000

$$\begin{aligned} Exp_{gesp} &= 0 \\ Exp_{real} &= -14 \end{aligned}$$

(Um Darstellungslücke zwischen normalisierten und denormalisierten Zahlen zu verwenden)

$$\text{Zahl} = 2^{-10} \cdot 2^{-14} = 2^{-24} \approx \frac{1}{1600000} = 0100000015$$

Umrechnung einer Zahl (zur Basis 10) in eine GKZ zur Basis im Computer.
Bsp: $-4,2 \cdot 10^{-1}$

1. Vorzeichen Merken, weiter mit Betrag
2. Darstellen als FKZ
3. Umrechnen der FKZ in die Zielbasis
4. Exp_{real} durch Stellenverschiebung der FKZ zur BAsis 2 bestimmen

5. Exp_{gesp} durch Aufaddieren des Bias auf Exp_{real} berechnen
6. Exp_{gesp} in Binärsystem umrechnen und passende Stellenzahl verwenden
7. Bitmuster in entsprechender Reihenfolge notieren

$$
\begin{aligned}
&4,2 \cdot 10^{-1}\\
&0,42\\
0,42 \cdot 2 &= 0,84\\
0,84 \cdot 2 &= 1,68\\
0,68 \cdot 2 &= 1,36\\
0,36 \cdot 2 &= 0,72\\
0,72 \cdot 2 &= 1,44\\
0,44 \cdot 2 &= 0,88\\
0,88 \cdot 2 &= 1,76\\
0,76 \cdot 2 &= 1,52\\
0,52 \cdot 2 &= 1,04\\
0,04 \cdot 2 &= 0,08
\end{aligned}
$$

$$0,0\underset{\text{Hidden Bit}}{\uparrow 1}\underbrace{1010111000}_{\text{10 Stellen}}0$$

Wertecodierung: Darstellung des Wertes in einem bestimmten Zahlensystem bzw. Umrechnung des Wertes von einem in ein anderes Zahlensysem.

Zifferncodierung: Umrechnung der Darstellung einer Zahl in einem SWS Ziffer für Ziffer in eine andere Darstellung

Beispiel für Zifferncodierung:

- Darstellung für Zahlen in Fließtext
- Umrechnung von z.B. $b = 16$ zu $b = 2$ über Umrechnungstabelle

Nachteil: deutlich erhöhter Speicherplatzbedarf.

3 Signalkodierung

Definition 3

Darstellung von abstrakten Informationen als Signalfolge.
Signal: physisch messbare Größe.

3.1 mögliche Signalformen

- elektrisch
  Spannung, Stromstärke, elektromagnetische Wellen, Ladung
- optisch
  Helligkeit, Farbe
- akustisch Lautstärke, Tonhöhe
- Druck hydraulisch, pneumatisch

in der Computertechnik relevant:

optisch ⇒ für netzwerkschnittstellen

ekeltrisch ⇒ insbesondere in der Digitaltechnik

vor allem: Spannung (evtl. auch Stromstärke)
⇒ eher kleinere Spannungspegel in der Digitaltechnik

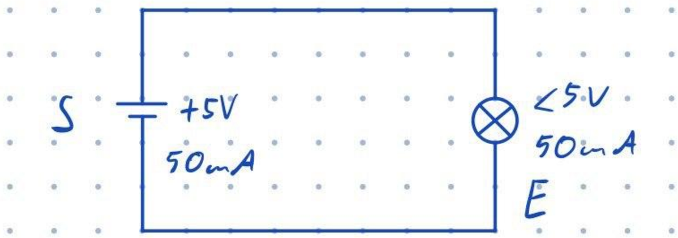


Abbildung 1: Schaltplan mit Spannungsquelle und Glühbirne

Strom Vs. Spannung

Spannung: Verringert sich beim E durch Spannungsfall auf der Leitung, wird verändert durch Störungen von außen („Übersprechen“ elektromagnetische Einstrahlungen)

Strom: Kaum davon betroffen; Stromstärke verändert sich im geschlossenen Stromkreis nicht.

Nachteil Strom: hoher Energieaufwand, große Wärmeentwicklung

⇒ deshalb Spannung statt Strom bei den meisten Computerschnittstellen verwandt.

typische Spannungspegel:

$$\left.\begin{array}{ccc} 0 & \hat{=} & 0V \\ 1 & \hat{=} & 5V \end{array}\right\} \text{TTL-Pegel}$$

„Transistor-Transistor-Logic“

⇒ „erfunden“um 1960 von TI (TexasInstruments)

große vs. kleine Spannungspegel

⊖ größere sind stromanfällig bei kleineren Spannungen

⊕ viel weniger Energieaufwand bei kleinen Spannungen, also auch weniger Wärmeentwicklung

⊕ weniger Störauswrkungen bei kleineren Spannungen

⊕ schnelleres Spannungswechseln bei kleinen Spannungshüben möglich

3.2 Umsetzung von Bitfolgen in Spannungspegelfolgen

meist getaktet, d.h. festes Zeitraster für die Bitfolge, d.h. jedes Bit braucht eine konstante, gleichlange zeitdauer

4 Verfahren (1-4) und 4 Eigenschaften(a-d)

a) Taktrückgewinnung (TRG)
Möglichkeit, nur aus dem übertragenen Datensignal eine Resynchronisierung biem Empfänger auf den Takt des Senders zu machen

b) Gleichspannungs-/stromfreiheit (GSF)
Motivation: Einsparung der Masseleitung. Im zeitlichen Mittel liegen auf der Signalleitung 0V an.
Ziel: Anhand des mittleren Signalpegels soll der Massepegel „errechnet“ werden.

-Vermeidung einer Potentialverschiebung beim E.

-Pseudoargument: keine Energieübertragung beim vom S zum E.

Grundvoraussetzung: symmetrische Pegel statt single-ended.
z.B. $1V \hat{=} +5v$ und $0 \hat{=} -5V$
dann: GSF bei Non-Return-to-zero (NRZ): falls #„0“ = #„1“ (bzw. „1“ und „0“ im Datenstrom gleichverteilt)

Bei den meisten Anwendungs-, Zeichen- und Zahlencodierungen kann keine Gleichverteilung angenommen werden. Ausnahme: Verschlüsselung und Kompression.

c) Störsicherheit (SSH)
(Un-)Anfälligkeit eines Verfahrens ggü. Störungen, welche durch Spannungsschwankungen auf der Leitung verursacht werden.
⇒ direkt abhängig von der Anzahl der verwendeten Pegel, welche auf einen vorgegebenen Potentialbereich verteilt werden müssen (und so natürlich auch beim Empfänger voneinander unterschieden werden müssen)
SSH bei Alternate Mark Inversion (AMI): schlecht, da 3 Pegel
SSH bei NRZ und Return-to-Zero (RZ): gut, da „nur“ 2 Pegel (und weniger Pegel geht nicht ☺)

1. NRZ
   Während der gesamten Schrittdauer wird der Pegel angegegt, welcher dem zu übertragenen Bitwert entspricht

2. RZ Jeder Schritt wird in zwei Schrittzeithälften eingeteilt. Während der ersten Hälfte wird der Pegel eingenommen, welcher den zu übertragenden Bitwert entspricht und während der zweiten Hälfte immer der „0“Pegel.

   TRG bei RZ: Bei jeder „1“ möglich, dann bei einer „1“ zu Beginn und in der Mitte der Schrittzeit ein Pegelwechsel stattfindet

3. AMI
   Ähnlich NRZ mit single-ended Pegeln, d.h. „0“ wird immer mit $0V$ (während der gesamten Schrittzeit) übertragen, aber „1“ abwechselnd mit z.B. $+5V$ und $-5V$ (während der gesamten Schrittzeit)

   GSF bei AMI: nachjeder 2. „1“: In der Praxis ist der GS-Anteil nach der ungeraden „1“ vernachlässigbar (bei langer Übertragungszeit und vielen übertragenen „1“), also praktisch immer GSF.

   TRG bei AMI: bei jeder „1“ nur bei langer Folge von „0“ keine TRG möglich

4. Manchester
   Bitwert wird über Spannungswechsel in der Mitte der Schrittzeit dargestellt, z.B. steigende Flanke ≙ „1“und fallende Flanke ≙ „0 “.
   ggf. ist ein weiterer Spannungswechsel zu Beginn der Schrittzeit notwendig, um den nachfogenden (inhaltsführenden) Spannungswechsel durchführen zu können.

   TRG bei Manchester: bei jedem Schritt möglich, da immer Regelwechsel in der Mitte der Schrittzeit.

   GSF bei Manchester: immer, da sich die Pegel in erster und zweiter Schrittzeithälfte gegenseitig ausgleichen

   SSH bei Manchester: optimal, da „nur “2 Pegel

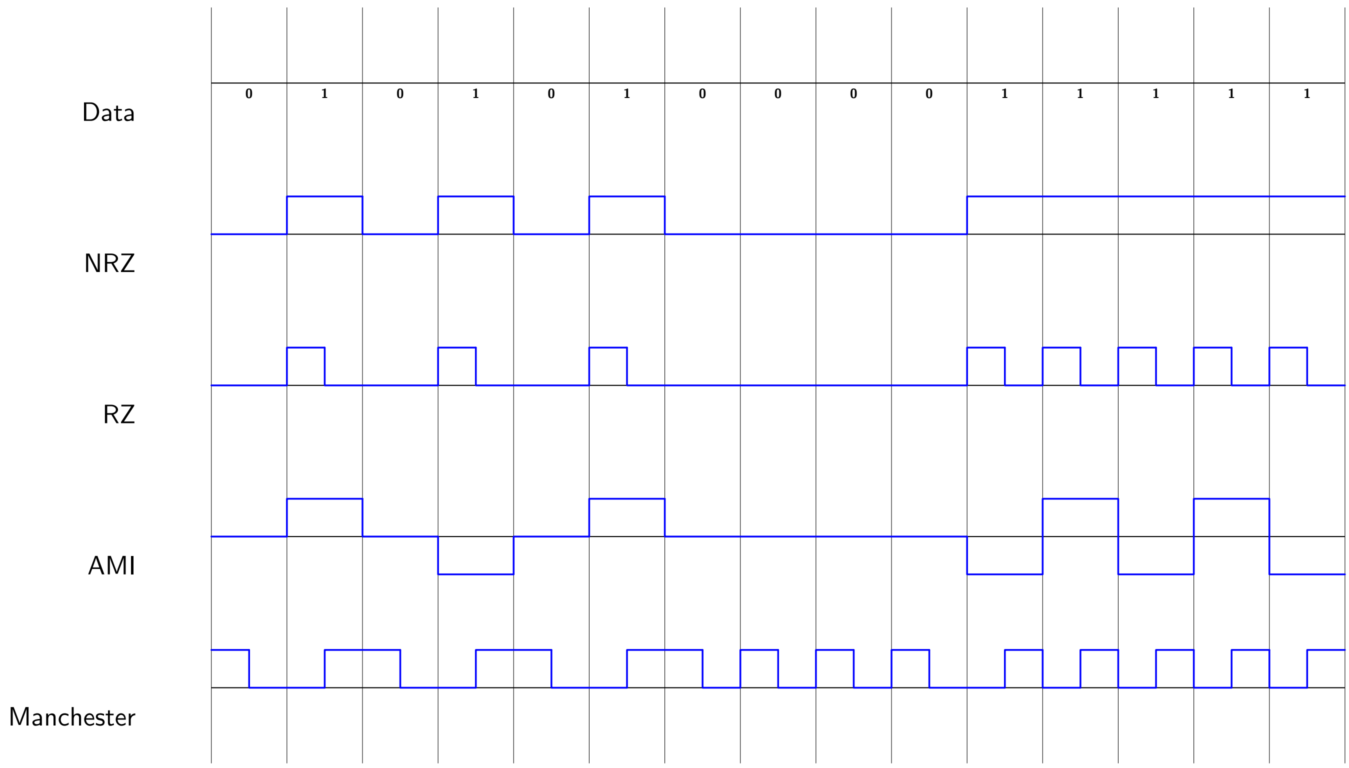


Abbildung 2: Pegelgraphen der einzelnen Verfahren

Bandbreitenbedarf, bandbreitenbegrenzte Übertragungskanäle

Nyquist-Theorem: Über einen Übertragungskanal mit beschränkter Bandbreite kann maximal mit der Schrittrate übertragen werden, welche der doppelten Bandbreite entspricht.
Bandbreitenbedarf (BBB): notwendige Bandbreite für ein bestimmtes Signalcodierungsverfahren bei einer bestimmten vorgegebenen Schrittrate
BBB bei NRZ: halbe Schrittrate, d.h. optimal H. Nyquist (max Frequenz bei „010101...“)
BBB bei RZ: Schrittrate, d.h. doppelt so viel wie nötig. (max Frequenz bei „000000...“, oder „111111...“)
BBB bei AMI: halbe Schrittrate (max Frequenz bei „111111...“)
BBB bei Manchester: Schrittrate (max Frequenz bei „000000...“oder „111111 “)

	TRG	GSF symm. Pegel als Grundvoraussetzung	SSH abhängig von Anzahl Pegel	BBB abhängig von Schrittrate
NRZ	bei „01 “und „10 “ ⊜	#„1 “= # „0 “ (1 und 0 gleichverteilt) ⊖	2 ⊕	halbe ⊕
RZ	bei jeder „1 “ ⊖	symm. Pegel & nur „1 “, single ended & nur „0 “, #1 = #0 und umgekehrt ⊜	2 ⊕	ganze ⊖
AMI	bei jeder „1 “ ⊖	nach jeder zweiten „1 “, in der Praxis „immer “ ⊕	3 ⊖	halbe ⊕
Manch.	immer ⊕	immer ++	2 ⊕	ganze ⊖

Einsatz:

- NRZ für interne Schnittstellen, bei denen der Verzicht auf Takt- oder Masseleitung nicht relevant ist
- Manchester gerne für Netzwerkschnittstellen (z.B. Ethernet) um auf Takt- und Masseleitung verzichten zu können

3 Verfahren zur sicheren Taktrückgewinnung:

1. **Startbitsequenz**
   Vor n Nutzdatenbit wird eine Startbitsequenz gestellt, welche sichere TRG ermöglicht. z.B. bei NRZ: „01“oder „10 “. n ist abhängig von der Genauigkeit der Uhren.
   Nachteil: relativ großer Overhead, effektive Nutzdatenrate deutlich kleiner als die Schrittrate.
   Im Beispiel: Nutzdatenrate $= \frac{n}{n+2} \cdot Schrittweite$
   Bsp: bei RZ oder AMI reicht ein einfaches „1 “-Startbit. (Keine „Sequenz“, weniger Overhead)
   Anwendung: z.B.serielle Schnittstelle RS232

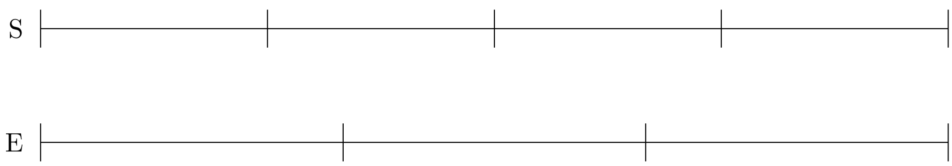
2. **Bitstuffing („Bitstopfen“)**
   Nach jeweils n gleichen direkt aufeinanderfolgenden Bitwerten wird ein Bit mit dem eingesetzten Bitwert eingefügt.
   z.B. n=3:

0	0	1	1	1		0	0	0		0	1	1	1		1	1	1	
1	2	1	2	3	↑	1	2	3	↑	1	1	2	3	↑	1	2	3	↑
					0				0					0				0

   Hinweis: Der Empfänger schaut nach n gleichen Bitwerten den nächsten Bitwert an. Bei entgegengesetztem Bitwert wird dieses „Stopfbit“entfernt. Bei gleichem Bitwert wird ein Fehler nach oben gemeldet.
   Vor-/Nachteile: Overhead bei NRZ im schlimmsten fall nur halb so groß wie bei der Startbitsequenz (nur eine statt zwei Schrittzeiten pro n Nutzdatenbit) und im besten Fall gar kein Overhead!
   bei RZ: nur nach n „0“wird eine „1 “eingefügt, d.h. weniger Overhead als bei NRZ und Bitstuffing. Verfahren ist komplex und deshalb “teuer“und „fehleranfällig“. Nutzdatenrate ist nicht konstant abhängig von Schrittrate, sondern sie variiert abhängig von den zu übermittelnden Nutzdaten (Overhead ist variabel) ⇒ schlecht für Anwendungen mit konstanter Nutzdatenrate (z.B. PCM-kodiertes Audio)
   Anwendung: Ethernet (um Bitmuster des Frame-Delimiters „01111110“auszuschließen ⇒ nach fünf „1“wird eine Stopf-„0“eingeschoben)

3. **Blockcodierung**
   Ein Block von n Nutzdatenbit wird als Block von $(n + i)$ zu übertragende Datenbit codiert, wobei nur solche Blöcke verwendet werden, welche sichere TRG ermöglichen.
   Nachteil: konstant großer Overhead
   Vorteil: ggf. sind weitere positive Eigenschaften erzielbar durch geeignete Auswahl der zu verwendenden Datenblöcke bei der Übertragung (vgl. 8B10B-Codierung und GSF!)
   Anwendung: z.B. ISDN und viele andere

Ganggenauigkeit der Uhren und Anzahl der Schritte ohne Resynchronisierung:



Der Pegel wird beim Empfang in der Mitte der angenommenen Schrittzeit abgetastet.
$\Rightarrow$ Die erlaubte Abweichung der Uhr bei E von der uhr bei S ist (weniger als) eine halbe Schrittzeit. Da sowohl S- als auch E-Uhr eine Ganggenauigkeit aufweisen können, darf jede Uhr um maximal 25% einer Schrittzeit abweichen.
Bsp: Bei 5% spezifischer Ganggenauigkeit wären 5 Schirrte „zu viel“

4 Boolsche Algebra

(angelehnt an das Skript von Burkhard Stiller an der Uni Zürich „Info3 Modul Schaltnetze“)

Benannt nach irischem (?) Mathematiker George Boole (1815-1864)
Rechensystem mit bestimmten Regeln:

- endliche Wertemenge W
- zwei zweistellige Operatoren $\otimes, \oplus$
- Abgeschlossenheit: $\forall a, b \in W: \; a \otimes b \in W, \; a \oplus b \in W$

Es gelten die 4 huntington'schen Axiome: $\forall a, b, c \in W$

(H1) **Kommutativgesetz**
$a \oplus b = b \oplus a, \; a \otimes b = b \otimes a$

(H2) **Distributivgesetz**
$a \oplus (b \otimes c) = (a \oplus b) \otimes (a \oplus c)$
$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$

(H3) **Neutrales Element:** $\exists n, e \in IV$
$a \oplus n = a \;, \; a \otimes e = a$

(H4) **Inverses Element:** $\exists \overline{a} \exists IV$
$a \oplus \overline{a} = e \;, \; a \otimes \overline{a} = n$

Spezialfall: Schaltalgebra
Wertemenge besteht aus zwei Werten: $IV = \{0, 1\} = \{false, true\} = \{falsch, wahr\} = \{off, on\} = \{aus, an\}$
Operatoren: statt $\oplus$: $\vee$, $ODER$, OR (, +)
statt $\otimes$: $\wedge$, UND, AND
(statt $a \wedge b$ geht auch ab)
$\Rightarrow$ zweistellige Operatoren

Durch (H4) wird ein einstelliger Operator definiert:
$\overline{a} = \neg a \;\; NICHT, NOT$

4.1 Schaltalgebra

4.1.1 Huntingtonsche Axiome in der Schaltalgebra

(H1) $a \vee b = b \vee a, \; a \wedge b = b \wedge a$

(H2) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

(H3) $a \vee 0 = a, \; a \wedge 1 = a$

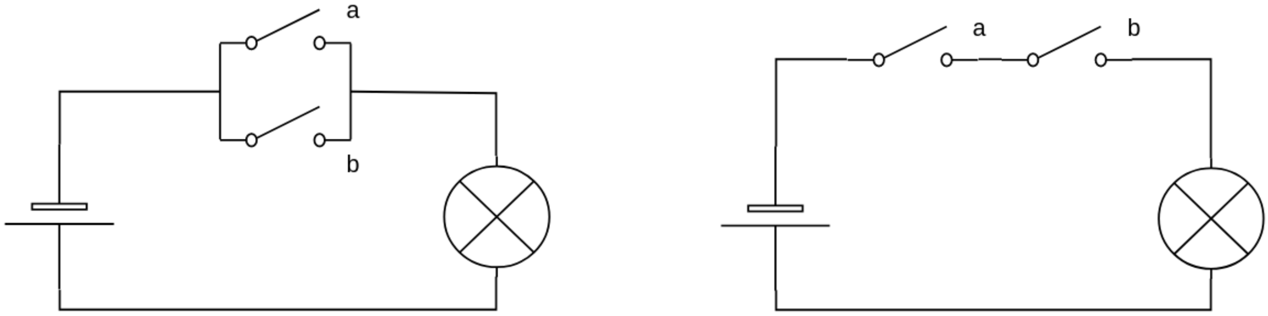
(H4) $a \vee \overline{a} = 1, \; a \wedge \overline{a} = 0$
(oder: $a \vee \neg a = 1, \; a \wedge \neg a = 0$)

4.1.2 Warum Schaltalgebra?

$\Rightarrow$ Darstellung der zweistelligen Operatoren mit Schalttasten, wobei die Werte durch Schalter dargestellt wurden.
Darstellung mit Wertetabellen:

Ausdrücke der Schaltalgebra („boolsche Ausdrücke“) bestehen aus:

- ein- und zweistelligen Operatoren
- Variable (als Platzhalter für einen Wert)



(a) ODER: $a \lor b$

(b) UND: $a \land b$

Abbildung 3: Darstellung zweistelliger Operatoren mit Schaltnetzen

b	a	$a \lor b$	$a \land b$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

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

Tabelle 2: Wahrheitstabellen für UND/ODER und Negierung

- Wert
- Klammern

Definition 4: Eingangsbelegung

Jeder Variable wird ein konkreter Wert zugeordnet

Definition 5: Ausgangsbelegung

Der Wert, welcher sich bei einem boolschen Ausdruck bei einer konkreten E-Belegung ergibt, wenn man den boolschen Ausdruck „auswertet“.

Auswertung eines boolschen Ausdrucks: $(a \land \neg c) \lor 1 \land (b \land c) \lor (0 \land d)$ (Siehe Tabelle 3)

- Festlegung der E-Belegung
- Ersetzen der Variablen durch die entsprechenden Werte
- Auswerten „von innen nach außen“

  Zunächst den Teilausdruck mit der stärksten Bindungskraft, zuletzt der Teilausdruck mit der schwächsten Bindungskraft: am stärksten... NOT, Klammer, AND, OR ...am schwächsten
- bei gleicher Bindungskraft Auswertug von links nach rechts

4.1.3 Aus den Huntingtonschen axiomen abgeleitete (beweisbare Gesetze)

Assoziativgesetz	$(a \land b) \land c = a \land (b \land c)$
	$(a \lor b) \lor c = a \lor (b \lor c)$
Idempotenzgesetz	$a \land a = a$
	$a \lor a = a$
Absorptionsgesetz	$a \land (a \lor b) = a$
	$a \lor (a \land b) = a$
DeMorgan-Gesetz	$\overline{(a \land b)} = \overline{a} \lor \overline{b}$
	$\overline{(a \lor b)} = \overline{a} \land \overline{b}$

4.1.4 Beweismethoden

- algebraische Umformungen mit Hilfe der Axiome und bereits bewiesener Gesetze:
  $a \overset{!}{=} a \land a$ (Idempotenzgesetz)
  $a \overset{H3}{=} a \land 1 \overset{H4}{=} a \land (a \lor \overline{a}) \overset{H2}{=} (a \land a) \lor (a \land \overline{a}) \overset{H4}{=} (a \land a) \lor 0 \overset{H3}{=} a \land a$

#	d	c	b	a	$\neg c$	$a \wedge \neg c$	$1 \wedge (b \wedge c)$	$0 \wedge d$	$x \vee y$	f
0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	1	1	1	0	0	1	1
2	0	0	1	0	1	0	0	0	0	0
3	0	0	1	1	1	1	0	0	1	1
4	0	1	0	0	0	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0	0
6	0	1	1	0	0	0	1	0	1	1
7	0	1	1	1	0	0	1	0	1	1
8	1	0	0	0	1	0	0	0	0	0
9	1	0	0	1	1	1	0	0	1	1
10	1	0	1	0	1	0	0	0	0	0
11	1	0	1	1	1	1	0	0	1	1
12	1	1	0	0	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0	0
14	1	1	1	0	0	0	1	0	1	1
15	1	1	1	1	0	0	1	0	1	1

Tabelle 3: Wahrheitstabelle für $f = (a \wedge \neg c) \vee 1 \wedge (b \wedge c) \vee (0 \wedge d)$

#	c	b	a	$b \vee c$	$a \vee (b \vee c)$	$a \vee b$	$(a \vee b) \vee c$
0	0	0	0	0	0	0	0
1	0	0	1	0	1	1	1
2	0	1	0	1	1	1	1
3	0	1	1	1	1	1	1
4	1	0	0	1	1	0	1
5	1	0	1	1	1	1	1
6	1	1	0	1	1	1	1
7	1	1	1	1	1	1	1
					$\uparrow$		$\uparrow$

gleiche Ausgangsbelegungen

Tabelle 4: Beweis anhand einer Wahrheitstabelle

- Wahrheitstabelle: Terme auf linker und rechter Seite müssen für alle Eingangsbelegugen dieselbe Ausgagsbelegung haben (Tabelle **??**)
- spezielle Interpretation von $H4$:
  falls $a \vee \overline{b} = 1$ und $a \wedge \overline{b} = 0$, dann $a = b$
  $a \vee (b \vee c) \overset{!}{=} (a \vee b) \vee c$ (Assoziativgesetz)

$\overline{a \wedge b} \overset{!}{=} \overline{a} \vee \overline{b}$

Beweis:

$$\overline{\overline{a \vee b}} \wedge (\overline{a} \vee \overline{b}) \overset{!}{=} 0$$

$$\overline{\overline{a \wedge b}} \vee (\overline{a} \vee \overline{b}) \overset{!}{=} 1$$

$$\overline{\overline{a \wedge b}} \wedge (\overline{a} \vee \overline{b}) \overset{\text{dopp. Neg.}}{=} (a \wedge b) \wedge (\overline{a} \vee \overline{b}) \overset{H2}{=} (a \wedge b) \wedge \overline{a} \vee (a \wedge b) \wedge \overline{b}$$

$$\overset{H1 \text{ \& AssG.}}{=} b \wedge (a \wedge \overline{a}) \vee a \wedge (b \wedge \overline{b}) \overset{H4}{=} (b \wedge 0) \vee (a \wedge 0) \overset{0-Abs.}{=} 0 \vee 0 \overset{H3 \text{ oder Id.pot.}}{=} 0$$

1. Hälfte
■

$$\overline{\overline{a \wedge b}} \vee (\overline{a} \vee \overline{b}) \overset{\text{dopp. Neg.}}{=} (a \wedge b) \vee (\overline{a} \vee \overline{b}) \overset{H2}{=} (a \vee (\overline{a} \vee \overline{b})) \wedge (b \vee (\overline{a} \vee \overline{b}))$$

$$\overset{H1 \text{ \& AssG.}}{=} ((a \vee \overline{a}) \vee \overline{b}) \wedge ((b \vee \overline{b}) \vee \overline{a})$$

$$\overset{H4}{=} (1 \vee \overline{b}) \wedge (1 \vee \overline{a})$$

$$\overset{\text{1-Absorp.}}{=} 1 \wedge 1$$

$$\overset{H3 \text{ oder Id.pot.}}{=} 1$$

2. Hälfte
■

doppelte Negation: $\overline{\overline{x}} \overset{!}{=} x$

Beweis über: $\overline{\overline{x}} \wedge \overline{x} \overset{!}{=} 0$ und $\overline{\overline{x}} \vee \overline{x} \overset{!}{=} 1$

$$\left.\begin{array}{ll} \overline{(\overline{x})} \wedge (\overline{x}) \overset{H4}{=} 0 & \overset{\text{1. Hälfte}}{\blacksquare} \\ \overline{(\overline{x})} \vee (\overline{x}) \overset{H4}{=} 0 & \overset{\text{2. Hälfte}}{\blacksquare} \end{array}\right\}\blacksquare$$

0-Absorption: $x \wedge 0 \overset{!}{=} 0$
Beweis: $x \wedge 0 \overset{H4}{=} x \wedge (x \wedge \overline{x}) \overset{\text{AssG.}}{=} (x \wedge x) \wedge \overline{x} \overset{\text{Id.pot.}}{=} x \wedge \overline{x} \overset{H4}{=} 0\blacksquare$

1-Absorption: $x \vee 1 \overset{!}{=}$
Beweis:

Definition 6

Boolsche Funktionen abhängig von n Eingangsvariablen: Jeder Eingangsbelegung (der n Eingangsvariablen) wird genau eine Ausgangsbelegung zugeordnet.

Darstellung von Funktionen:

1. algebraischer Funktionsterm
2. Wahrheitstabelle

Definition 7

Zwei Funktionen sind äquivalent, wenn sie dieselbe Wahrheitstabelle aufweisen

Hinweis

Zwei äquivalente Funktionen können durch sehr unterschiedliche Funktionsterme dargestellt werden. (algebraische Umformungen mit Hilfe der Axiome und abgeleiteten Gesetze sind immer möglich)

4.1.5 Funktionen abhängig von Ausgangsvariablen

$n = 0$	$f_0() = 0$	„Nullfunktion“
	$f_1() = 1$	„Einsfunktion“
$\Rightarrow$ (nur) 2 verschiedene Funktionen abhängig von 0 Variablen!		
$n = 1$	$f_0(a) = 0$	„Nullfunktion“
	$f_1(a) = \overline{a}$	„Negation “
	$f_2(a) = a$	„Identität “
	$f_3(a) = 1$	„Einsfunktion “
$\Rightarrow$ genau 4 verschiedene Funktionen abhängig vn 1 Variablen!		

Tabelle 5: Funktionen abhängig von Ausgangsvariablen ($n = 0 \dots 1$)

Anzahl verschiedener Funktionen?

Anzahl Zeilen in der Wertetabelle: 2^n
Anzahl verschiedener Wertetabellen: 2^{2^n} $\Rightarrow$ Anzahl der Funktionen abhängig von Eingangsvariablen.

n	2^n	2^{2^n}	
0	1	2	
1	2	4	
2	4	16	
4	16	65536	
5	32	$\approx$ 4 Mrd	
6	64	$\approx$ 16 Trio	$\approx$ 1.000.000.000.000.000.000

$$\begin{array}{|lcr|} 2^{10} & = & 1024 \\ & \approx & 1000 \end{array}$$

Implikation: $a \Rightarrow b$ („aus a folgt b“)
Aussage a: „An der DHBW KA gibt es einen Corona-Fall.“
Aussabe b: „An der DHBW KA finden keine Präsenzvorlesungen statt.“

#	b	a	$a \Rightarrow b$
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	1

b	a	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$f(a,b) =$		0	$\overline{a \vee b}$	$\overline{a \Rightarrow b}$	$\bar{b}$	$\overline{b \Rightarrow a}$	$\bar{a}$	$\overline{a \Leftrightarrow b}$	$\overline{a \wedge b}$	$a \wedge b$	$a \Leftrightarrow b$	a	$b \Rightarrow b$	b	$a \Rightarrow b$	$a \vee b$	1
		Nullfunktion	NOR	Inhibition	Negation von b	Inhibition	Negation von a	XOR/Antivalenz	NAND	UND/AND	Äquivalenz	Identität von a	Implikation; aus b folgt a	Identität von b	Implikation: aus a folgt b	ODER/OR	Einsfunktion

Tabelle 6: Wahrheitstabelle boolscher Operationen

Satz:	$\{\wedge, \vee, \neg\}$ ist ein vollständiges Operatorensystem.
Bew.:	Definition der Boolschen/Schaltalgebra.
Satz:	$\{\wedge, \neg\}$ ist ein volständiges Operatorensystem.
Bew.:	$a \vee b \overset{\text{dopp.Neg.}}{=} \overline{\overline{a \vee b}} \overset{\text{De Morgan}}{=} \overline{\bar{a} \wedge \bar{b}}$ ■
Satz:	$\{\vee, \neg\}$ ist ein vollständiges Operatorensystem.
Bew.:	$a \wedge b \overset{\text{dopp.Neg.}}{=} \overline{\overline{a \wedge b}} \overset{\text{De Morgan}}{=} \overline{\bar{a} \vee \bar{b}}$ ■
Satz:	$\overline{\wedge}$ („NAND “) ist ein vollständiges Operatorensystem.
Bew.:	anhand des vollständigen Operatorensystems $\{\vee, \neg\} \ldots$
	$\neg a = \bar{a} \overset{Id.pot.}{=} \overline{(a \wedge a)}$ $^{\text{1. Hälfte}}$ ■
	$a \vee b \overset{\text{dopp.Neg.}}{=} \overline{\overline{a \vee b}} \overset{\text{De Morgan}}{=} \overline{\bar{a} \wedge \bar{b}} = \overline{\overline{a \wedge a} \wedge \overline{b \wedge b}}$ $^{\text{2. Hälfte}}$ ■
Satz:	$\{\overline{\vee}\}$ („NOR “) ist ein vollständiges Operatorensystem.
Bew.:	

Tabelle 7: Sätze/Beweise über vollständige Operatorensysteme

Definition 8

Ein vollständiges Operatorensystem (der Boolschen Algebra/Schaltalgebra) ist eine Menge von Operatoren, mit denen jede (Boolsche) Funktion (der Schaltalgebra) dargestellt werden kann.

Bsp: Speicherbausteine in „NAND “-Technologie.

Hinweis

Funktionen können eindeutig durch die Wahrheitstabelle dargestellt werden. Die Darstellung als Funktionsterm ist dagegen nicht eindeutig.
⇒ für jede Funktion gibt es unendlich viele äquivalente Terme!
Wir suchen einen „standardisierten“Funktionsterm!

Wir suchen einen standardisierten Funktionsterm!
Ausgangspunkt: Wertetabelle

#	c	b	a	$f(a,b,c)$	Minterm	
0	0	0	0	1	$\bar{c}\bar{b}\bar{a} = (\bar{c} \wedge \bar{b}) \wedge \bar{a}$	$\bar{c}\bar{b}$
1	0	0	1	1	$\bar{c}\bar{b}a$	
2	0	1	0	0		$\bar{c}a$
3	0	1	1	1	$\bar{c}ba$	
4	1	0	0	0		
5	1	0	1	0		ba
6	1	1	0	0		
7	1	1	1	1	cba	

- Ein Literal ist eine Variable in negierter oder nicht-negierter Form

$$f(a,b,c) = \overline{c}\overline{b}\overline{a} \vee \overline{c}\overline{b}a \vee \overline{c}ba \vee cba \qquad \text{Disjunktive Normalform (DNF)}$$
$$g(a,b,c) = (a \wedge b) \vee (\overline{b} \wedge \overline{c}) \qquad \text{Disjunktivie Minimalform (DMF)}$$

$$\overline{c}\overline{b}\overline{a} \vee \vee \overline{c}\overline{b}a \overset{H2}{=} (\overline{c}\overline{b}) \wedge (\overline{a}\ veea) \overset{H4}{=} \overline{c}\overline{b} \wedge 1 \overset{H3}{=} \overline{c}\overline{b}$$

- Ein Implikant ist eine Konkunktion von Literalen, für den gilt $i \implies f$
- Ein Minterm ist ein Implikant, bei dem es für jede E-Variable ein Literal gibt. (Auch Viollkonjunktion genannt)

Satz:	Ein Minterm hat nur bei einer einzigen E-Belegung „1 “als A-Belegung, in allen anderen Fällen der E-Belegung ergibt sich „0“als A-Belegung.
Bew.:	geschenkt (Verständnis)
Satz:	Die Disjunktive Normalform (DNF) der Funktion f ist ein äquivalent zur Funktion f.
Bew.:	Verständnis/Bildungsregel
Satz:	DieDNF ist (ausgenommen die Reihenfolge der Minterme, sowie die Reihenfolge der Literale in den Mintermen) für eine gegebene Funktion eindeutig.
Bew.:	Bildungsregel & Wertetabelle ist eindeutig.
Satz:	Zwei Implikanten einer Funktion lassen sich zu einem Implikanten zusammenfassen, falls: • sie ausgenommen ein Literal identisch sind • das sich unterscheidende Literal dieselbe Variable einmal in negierter und einmal in nicht-negierter Form sein muss Der Zusammengefasste Implikant ist dann der, welcher durch weglassen des unterschiedlichen Literals entsteht.
Bew.:	$x \wedge a \vee x \wedge a \overset{H2}{=} x(a \vee \overline{a}) \overset{H4}{=} x1 \overset{H3}{=} x\blacksquare$
Satz:	Eine Disjunktivie Minimalform (DMF) enthält ausschließlich Primimplikant (PI).
Bew.:	Siehe Definition 11
Satz:	Eine DMF enthält mindestens alle Kernprimzahlen

Definition 9

Die Disjunktion aller Minterme der Funktion f heißt DNF.

Definition 10

Ein PI ist ein Implikant, der mit keinem anderen Implikanten zusammengefasst werden kann

Definition 11

Ein Kernprimimplikant (KPI) ist ein PI, welcher mindestens eine der Ausgangsbelegung exklusiv abdeckt (bzw. bei dem mindestens ein Minterm beim Zusammenfassen ausschließlich für diesen PI verwendet wurde.)

4.2 Darstellung von Funktionen

4.2.1 Schaltnetze

Schaltnetze bestehen aus Gattern und Leitungsverbindungen. Sie sind die Darstellung von Funktionen als Graph, genauer als gerichteter Graph („Reihenfolge“!)

Gatter: Knoten	$\hat{=}$ Operatoren
Leitungsverbindungen: Kanten	$\hat{=}$ Reihenfolge

Konvetionen:

- Jeder Ein- oder Ausgang eines Gatters kann durch einen nicht-ausgefüllten Kreis invertiert werden.
- Eingänge bei Gattern werden meist links (oder alternativ oben) und die Ausgänge meist rechts (oder unten) notiert. Nur in Ausnahmefällen sind Eingänge rechts oder unten bzw. Ausgänge links oder oben!
- UND- und ODER-Gatter sind auch mit mehr als zwei (mit beliebig vielen) Eingängen möglich (Abb. 5)
- „Verzweigen“von Leistungen auf mehrere nachfolgende Gattereingänge ist möglich, in dem an der Verzweigungsstelle ein ausgefüllter Kreis gezeichnet wird

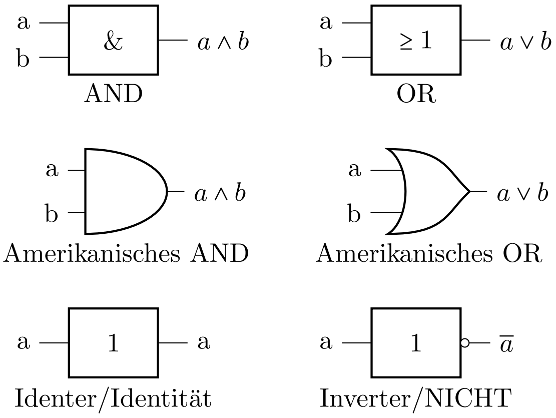


Abbildung 4: Symbole für die Operatoren („Gatter“) nach DIN/IEC

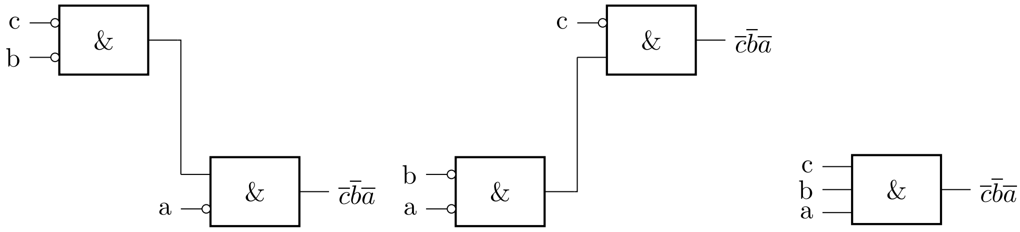


Abbildung 5: Unterschiedliche Darstellungen einer UND-Verschaltung

$$f(a,b,c) = \overline{c}\overline{b}\overline{a} \vee \overline{c}\overline{b}a \vee \overline{c}ba \vee cba$$

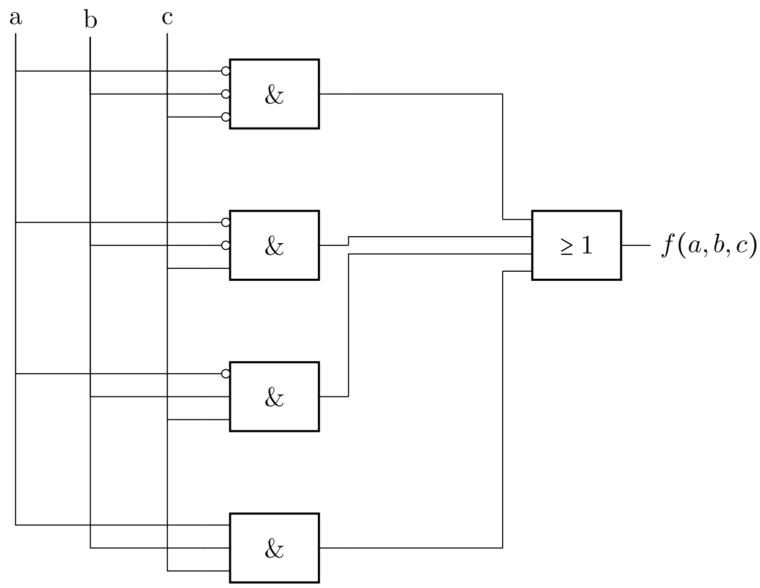


Abbildung 6: Realisierung der DNF als Schaltnetz

- Kreuzungen von Leitungen, die nicht miteinander verbunden sind, sind möglich. An der Kreuzungsstelle darf auch kein ausgefüllter Kreis notiert werden

Die Realisierung der DNF als Schaltnetz (Abb. 6) ist meist ungünstig, da es meist „günstigere“(weniger Hardware-aufwändige) Schaltnetze für dieselbe Funktion gibt.

4.3 Aufwand

- Hardware-Aufwand
- Zeitaufwand (nicht näher behandelt)

4.3.1 Hardware-Aufwand

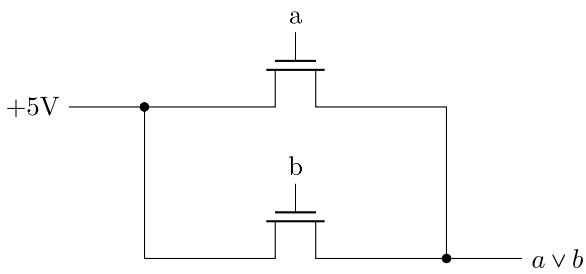


Abbildung 7: ODER

Abbildung 8: UND

⇒ wir „messen“den Hardware-Aufwand in Anzahl Transistoren.
Bei UND (Abb. 8) und ODER (Abb. 7) brauchen wir so viele Transistoren, wie das Gatter Eingänge hat (bei nicht negierten Ein- und Ausgängen).

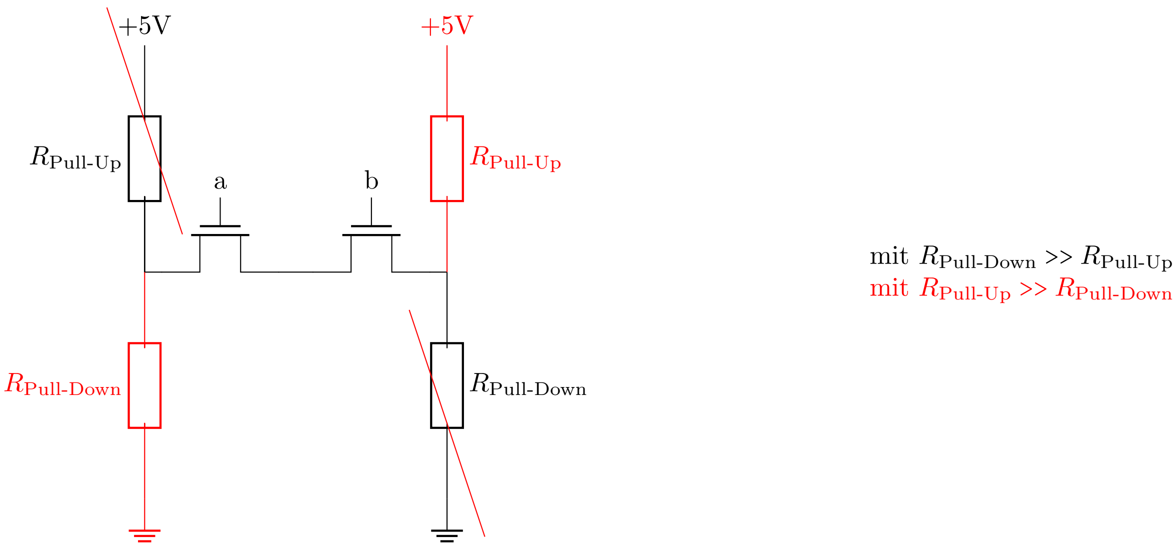


Abbildung 9: UND negierter Ausgang

npn-Transistor
schaltet bei positiver Basisspannung durch

pnp-Transistor
schaltet bei negativer Basisspannung durch

Abbildung 10: NPN und PNP-Transistor

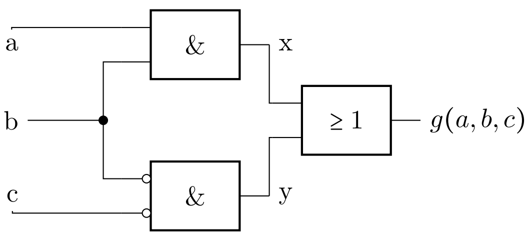
⇒ negierte Eingänge durch Ersetzen der üblichen npn-Transistoen durch pnp-Transistoren
⇒ auch bei negierten Eingängen reicht ein Transistor je Eingang

nMOS: Realisierung nur mit npn-Transistoren
pMOS: Realisierung nur mit pnp-Transistoren
⇒ früher gängige Realisierungstechnologien
heute: CMOS: (**C**omplimentary **M**etal **O**xide **S**miconductor)
Realisierung jedes Eingang durch 2 Transistoren (ein npn- und ein pnp-Transistor)
jeweils ein Transistor zeiht wechselweise nach oben oder nach unten

⇒ der Einfachheit halber trotzdem:
bei UND und ODER („Elementargatter“) entspricht die Anzahl der Eingänge der Anzahl dafür notwendiger Transistoren (nur zu Vergleichszwecken eingesetzt)

4.3.2 Schaltnetzanalyse

$$g(a,b,c) = (a \wedge b) \vee (\overline{b} \vee \overline{c})$$



#	c	b	a	x	y	$x \vee y = g(a,b,c)$
0	0	0	0	0	1	1
1	0	0	1	0	1	1
2	0	1	0	0	0	0
3	0	1	1	1	0	1
4	1	0	0	0	0	0
5	1	0	1	0	0	0
6	1	1	0	0	0	0
7	1	1	1	1	0	1

Abbildung 11

Im Schaltnetz (Abb. **??**) ist g äquivalent zu f (Abb. 6)! Für die Realisierung von g sind aber nur 6 Transistoren nötig, d.h. 10 Transistoren weniger als für f.

Wie finden wir eine weniger aufwändige Realisierung als die DNF?

Definition 12

Eine DMF ist eine Disjunktion von Konjunktionen von Literalen, welche die Funktion darstellt und „minimal“ist. Minimal bedeutet, mit am wenigsten Hardware-Aufwand zu realisieren.

4.3.3 KV-Diagramm

Graphische Darstellung der Ausgangsbelegung

#	a	$f(a)$
0	0	$f(0)$
1	1	$f(1)$

$\overline{a}$	a
$f(0)_0$	$f(1)_1$

Tabelle 8: KV-Diagramm mit 1 Variable

#	b	a	$f(a,b)$
0	0	0	$f(0,0)$
1	0	1	$f(0,1)$
2	1	0	$f(1,0)$
3	1	1	$f(1,1)$

	$\overline{a}$	a
$\overline{b}$	$f(0,0)_0$	$f(1,0)_1$
b	$f(0,1)_2$	$f(1,1)_3$

Tabelle 9: KV-Diagramm mit 2 Variablen

$\overline{b}$	1_0	1_1	0_5	0_4
b	0_2	1_3	1_7	0_6

$\underbrace{\quad}_{\overline{c}}$ $\underbrace{\quad}_{c}$

$f(a,b,c)$
- $\overline{c}(b)(0+1)$
- $\overline{c}a(1+3)$
- $ba(3+7)$

Tabelle 10: KV-Diagramm mit 3 Variablen

	$\overline{a}$	a	a	$\overline{a}$	
$\overline{b}$	0	1	5	4	$\overline{d}$
b	2	3	7	6	$\overline{d}$
b	10	11	15	14	d
$\overline{b}$	8	9	13	12	d

$\underbrace{\quad}_{\overline{c}}$ $\underbrace{\quad}_{c}$

3:	$ab\overline{c}\overline{d}$
1:	$a\overline{b}\overline{c}\overline{d}$
2:	$\overline{a}b\overline{c}\overline{d}$
7:	$abc\overline{d}$
11:	$ab\overline{c}d$
3+7	$ab\overline{d}$
11+15	abd
3+7+11+15	ab

Tabelle 11: KV-Diagramm mit 4 Variablen

PI können im KV-Diagramm gefunden werden, indem wir Bereiche von Feldern mit „1"-Belegung suchen, welche:

- maximal groß sind
- rechteckige Form haben
- eine 2er-Potenz an Feldern umfassen

Kernprimipimplikanten (KPI) erkennt man daran, dass mindestens ein Feld von keinen anderen PI abgedeckt wird.

Hinweis

Zwei benachbarte Felder unterscheiden sich immer in der Belegung genau einer Variable. Nachbarfelder von Randfeldern finden sich am gegenüberliegenden Rand.

	$\overline{a}$	a	a	$\overline{a}$	
$\overline{b}$	0	1	5	4	$\overline{d}$
b	1_2	3	7	1_6	$\overline{d}$
b	1_{10}	1_{11}	15	14	d
$\overline{b}$	8	9	13	1_{12}	d

$\underbrace{\quad}_{\overline{c}}$ $\underbrace{\quad}_{c}$

		KPI
~~2+6:~~	$\overline{a}b\overline{d}$	~~6~~
~~2+10:~~	$\overline{a}b\overline{c}$	–
~~11+10:~~	$b\overline{c}d$	~~11~~
12:	$\overline{a}\overline{b}cd$	12
DMF: $\overline{a}b\overline{d} \vee b\overline{c}d \vee \overline{a}\overline{b}cd$		
~~6+7:~~	$bc\overline{d}$	~~7~~

2+3+6+7: $b\overline{d}$ 6+7/KPI
2+3+10+11: $b\overline{c}$ 10+11/KPI
DMF: $b\overline{d} \vee b\overline{c} \vee \overline{a}\overline{b}cd$

DMF: $b\overline{c}d \vee \overline{a}\overline{b}cd \vee bc\overline{d} \vee \overline{a}b\overline{d} = b\overline{c}d \vee \overline{a}\overline{b}cd \vee bc\overline{d} \vee \overline{a}b\overline{d}$
$\Rightarrow$ Es gibt 2 DMFs für diese Funktion $\Rightarrow$ DMF nicht eindeutig!

Anmerkung:

- Eine DMF, welche nur aus KPI besteht, ist immer eindeutig.
- Wenn die KPI nicht alle „1"-Felder abdecken, benötigt man auch einfache PI für eine DMF.
- Für jedes nicht von einem KPI abgedeckte „1"-Feld gibt es immer mindestens 2 PI, welche dieses Feld abdecken. (Wenn es nur einen PI gäbe, wäre dieser PI ein KPI!)
- Falls ein PI für die DMF gebraucht wird, kann die DMF nicht eindeutig sein; sie kann aber auch eindeutig sein, falls z.B. die beiden fas fehlende Feld abdeckender PI unterschiedliche Größe haben.

	$\overline{a}$	a	a	$\overline{a}$	$\overline{a}$	a	a	$\overline{a}$	
$\overline{b}$	0	1	5	4	20	21	17	16	$\overline{d}$
b	2	3	7	6	22	23	19	18	$\overline{d}$
b	10	11	15	14	30	31	27	26	d
$\overline{b}$	8	9	13	12	28	29	25	24	d

$\underbrace{\quad}_{\overline{c}}$ $\underbrace{\quad}_{c}$ $\underbrace{\quad}_{\overline{c}}$ $\underbrace{\quad}_{c}$

$\underbrace{\qquad}_{\overline{e}}$ $\underbrace{\qquad}_{e}$

3: $ab\overline{c}\overline{d}\overline{e}$
19: $ab\overline{c}\overline{d}e$ (Nachbar von 3?)

$\Rightarrow$ KV-Diagramm im 2-dimensionalen bis 4 E-Variablen, im 3-dimensionalen bis 6 E-Variablen, ...

Abkürzungsverzeichnis

SWS Stellenwertsysteme

FKZ Festkommazahl

GKZ Gleitkommazahl

BBB Bandbreitenbedarf

NRZ Non-Return-to-zero

RZ Return-to-Zero

AMI Alternate Mark Inversion

TRG Taktrückgewinnung

GSF Gleichspannungs-/stromfreiheit

SSH Störsicherheit

DNF Disjunktive Normalform

DMF Disjunktivie Minimalform

KPI Kernprimimplikant

PI Primimplikant