Dies ist nun also die freundlicherweise von mir mitgetippte Fassung der Vorlesung **Digitaltechnik** (1. Semester) bei **Hr. Schillack** an der **BA-Mannheim**.

Ich hoffe ihr könnt damit was anfangen.

Fehler, Kritik, Anregungen und alles was euch sonst noch dazu einfällt mailt bitte an
st_tost@hotmail.com

Literatur

H. Lindner / C. Lehmann / H. Brauner: Taschenbuch der Elektrotechnik und Elektronik
Fachbuchverlag Leipzig, 1999, ISBN 3-446-21056-3, 30 Mark

Inhalte / Gliederung **19 Seiten**

Stephan Tost
st_tost@hotmail.com

Verknüpfungen

1 + 1 = 1 logische Verknüpfung (AND)
1 + 1 = 10 arithmetische Verknüpfung (1 + 1 = 10 (binär))

Ein Prozessor zur digitalen Datenverarbeitung besitzt als Kern eine
logisch-arithmetische Einheit (ALU – **A**rithmetic **L**ogic **U**nit)

Realisationselemente:
- Verknüpfungsglieder
- Zeitglieder
- Speicherglieder

UND, ODER, NICHT sind logische Grundelemente

Beispiel: Fahrstuhl
Fahrstuhl fährt erst los, wenn Tür geschlossen UND Zieltaste gedrückt UND NICHT Nottaste gedrückt
Tür schließt erst, wenn Lichtschranke für bestimmte Zeit nicht ausgelöst wurde

➔ **Es gibt Zeitintervall oder Zeit definierende Funktionsglieder**

Bei mehreren gedrückten Tasten müssen Zieltasten gespeichert werden.
Speicherglieder dienen zum speichern und zur Verfügung stellen von Ziel- oder Vergleichsinformationen

➔ **Dazu dienen Zähler, Register und Speicher, welche auf Flip-Flops basieren**

Die logische Grundverknüpfungen

Signale können als Variablen betrachtet werden
Informationen bestehen aus Bitfolgen → **binäre Variablen** (sind also Element aus (0 ; 1))

Kodierung: 0 entspricht Signalpegel **Low**
1 entspricht Signalpegel **High**
→ positive **Logikzuordnung**
(negative Logikzuordnung entspricht also: 0 – high und 1 – low)

Signale sind elektrische Spannungen (U)
nach dem TTL (Transistor-Transistor-Logik) wird der Spannung von
+5V der Zustand **high** zugewiesen
0V entspricht dem **low** - Zustand

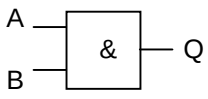
Schalter:
Schließerschalter: 0 – offen, 1 – geschlossen (betätigt)
Öffnerschalter: 0 – geschlossen (betätigt), 1 – offen

1. UND-Verknüpfung (AND)

a) Kontakttechnik / Kodierung
Beispiel: UND – Schaltung (zwei Schalter in Reihe mit Lampe)
Lampe: an = 1, aus = 0
Schalter: offen = 0, betätigt = 1
Bedingung für Ausgangssignal: an allen Eingängen muß High anliegen

b) Schaltalgebraischer Ausdruck
Q = A ^ B bzw. **Q = A * B** also **AB**

c) Schaltzeichen



Stephan Tost
st_tost@hotmail.com

d) Funktions - und Arbeitstabelle
- zwei Zustände der Eingangsvariablen = 2^n Kombinationen
- zwei Variablen → 4 mögliche Ergebnisse

Nr. dezimale Zahl der binären Kombination der Ein- und Ausgangswerten
Q Pflichtausgänge
P Produktterme

Nr.	A	B	Q	P
0	0	0	0	
1	0	1	0	
2	1	0	0	
3	1	1	1	A ^ B = Q

$Q = P_3$

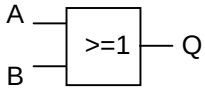
e) Realisation durch elektrische Spannung
nur sich überschneidene Partien der Spannungen der Eingänge ergeben eine Ausgangsspannung

2. ODER-Verknüpfung (OR)

a) Kontakttechnik / Kodierung
Beispiel: ODER – Schaltung (zwei Schalter parallel mit Lampe)
Lampe: an = 1, aus = 0
Schalter: offen = 0, betätigt = 1
Bedingung für Ausgangssignal: an einem der Eingängen muß high anliegen
(die arithmetische Summe der Eingänge muß größer oder gleich 1 sein [>=1])

b) Schaltalgebraischer Ausdruck
$Q = A \vee B$ bzw. **Q = A + B**

c) Schaltzeichen



d) Funktions - und Arbeitstabelle

S Summenterme

Nr.	A	B	Q	P	S
0	0	0	0		$S_0 = A + B$
1	0	1	1	$P_1 = A \wedge \neg B$	
2	1	0	1	$P_2 = \neg A \wedge B$	
3	1	1	1	$P_3 = A \wedge B$	

$Q = P_1 + P_2 + P_3$ bzw.
$Q = S_0 = A + B$

e) Realisation durch elektrische Spannung
alle Partien der Spannungen der Eingänge ergeben eine Ausgangsspannung

Stephan Tost
st_tost@hotmail.com

3. NICHT-Verknüpfung (NOT)

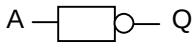
Das Eingangssignal wird negiert; die NICHT-Verknüpfung wird auch **Inverter** genannt.

a) Die NICHT-Verknüpfung wird durch einen Öffnerschalter repräsentiert.

b) Schaltalgebraischer Ausdruck

$Q = \overline{A}$ bzw. **Q = ¬ A** oder **Q = !A**

c) Schaltzeichen



d) Funktions - und Arbeitstabelle (Wahrheitstabelle)

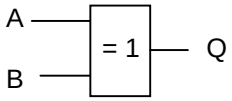
Nr.	A	Q
0	0	1
1	1	0

e) Realisation durch elektrische Spannung
Die Partien der Spannungen des Eingangs ergeben umgekehrt die Ausgangsspannung

4. XOR

a ←|→ b = a ⊕ b = !a !b + a b → Antivalenz

Schaltzeichen:



Schaltalgebra

∧	∨
1 ∧ 0 = 0	1 ∨ 0 = 1
1 ∧ 1 = 1	1 ∨ 1 = 1
0 ∧ 0 = 0	0 ∨ 0 = 0
a ∧ 1 = a	a ∨ 1 = 1
a ∧ 0 = 0	a ∨ 0 = a
a ∧ a = a	a ∨ a = a
a ∧ ¬a = 0	a ∨ ¬a = 1

Stephan Tost
st_tost@hotmail.com

Rechenregeln

Operationen innerhalb einer Art

Kommuativgesetz

$$a \wedge b = b \wedge a$$

Assoziativgesetz

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

gemischte Operationen

1. Distributivgesetz

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

2. Distributivgesetz

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

Beweis: ab + ac = a (b + c)

ab + ac
= aa + ac + ba + bc | aa = a
= a (1 + c + b) + bc | c + b = d, d +1 = 1
= a (1 + 1) + bc
= a + bc

De Morgan Regeln

1) $\neg(a \wedge b \wedge c) = \neg a \vee \neg b \vee \neg c$
2) $\neg(a \vee b \vee c) = \neg a \wedge \neg b \wedge \neg c$ [ist nicht gleich $\neg(a \wedge b \wedge c)$]

Shannon – Fano Regel

f (a, b, c, …; $\wedge$ $\vee$) =
¬ (f (¬a, ¬b, ¬c, …; $\vee$ $\wedge$))

Weitere Regeln

Absorptionsregeln

1) $a \vee (a \wedge b) = a$
2) $a \wedge (a \vee b) = a$
3) $a \wedge (\neg a \vee b) = a \wedge b$
4) $a \vee (\neg a \wedge b) = a \vee b$
5) $(a \vee b) \wedge (\neg a \vee c) = (a \wedge c) \vee (\neg a \wedge b)$

Erweiterungsregeln

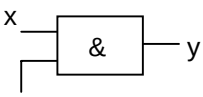
1) $a = (a \vee b) \wedge (a \vee \neg b)$
2) $a = a \wedge (b \vee \neg b)$
3) $a = (a \wedge b) \vee (a \wedge \neg b)$
4) $a = a \vee a$

Stephan Tost
st_tost@hotmail.com

Gattersonderfunktionen

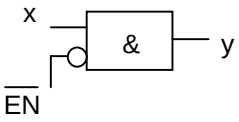
a) gesteuerter Schalter / Tor

i	s	x	y
0	0		0
1	0		0
2	1	0	0
3	1	1	1

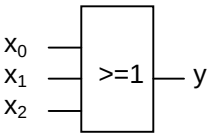


s: EN = CS (enable – chipselect)

zum Ansteuern eines Gerätes muß der Adressbus 1 und der Steuerbus 0 sein



b) Leitungszusammenführung

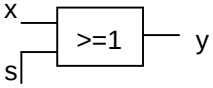


c) Einerkomplement

XOR $x_0 \leftarrow|\rightarrow x_1$

$= !x_0x_1 + !x_1x_0$

$= s!x + !sx$

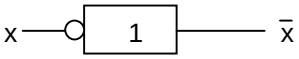


s	x	y
0	0	0
0	1	1
1	0	1
1	1	0

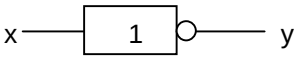
Das Ergebnis ist nur 1 wenn die Eingänge unterschiedlich sind.

d) Inverter

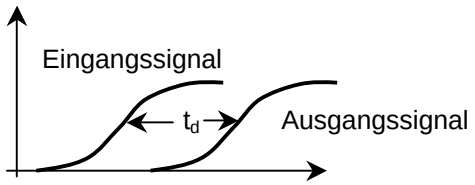
1) Inverter



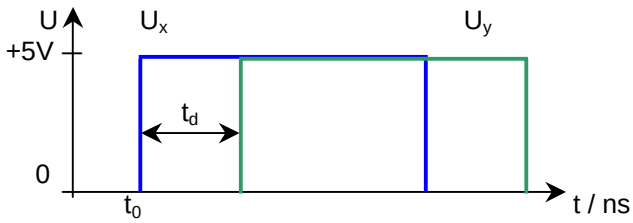
2) Delay (Verzögerungsglied)



Die Signaldurchlaufzeit wird von
50% des Eingangssignals zu
50% des Ausgangssignals gemessen.

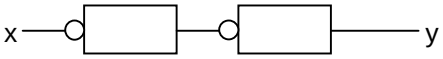


t_d – Verzögerungszeit (auch: t_p = Pulse Propagation Time)
(propagation = Übertragung)

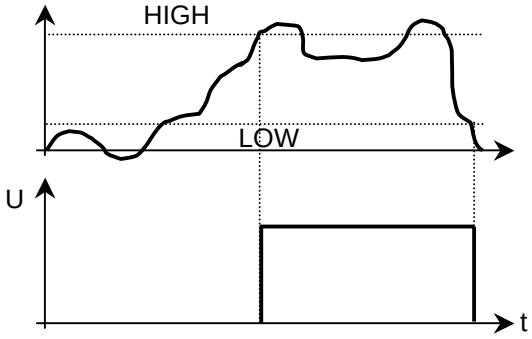


Stephan Tost
st_tost@hotmail.com

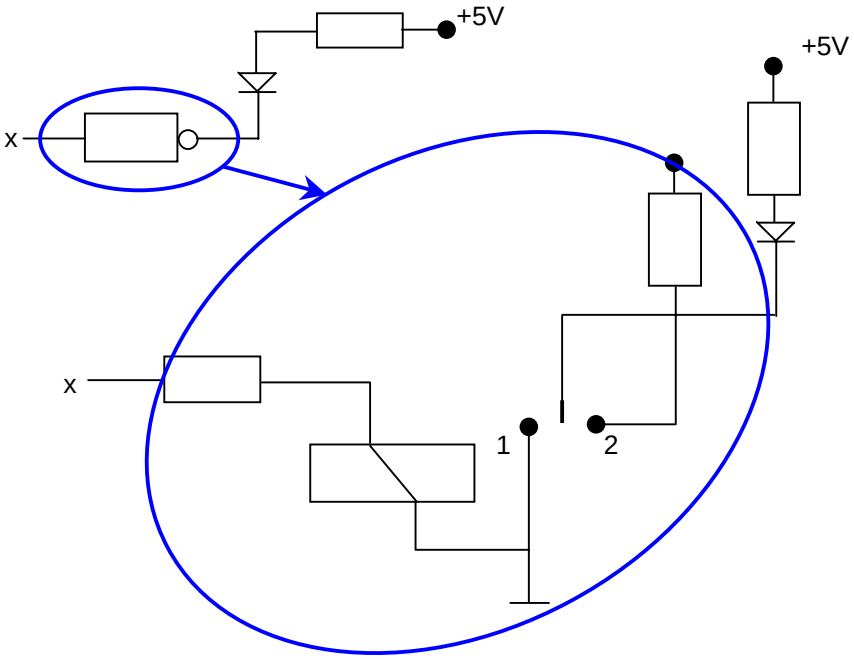
3) Signalregeneration



Wenn die HIGH - bzw. LOW - Schwelle
über- bzw. unterschritten wird,
wird umgeschaltet.

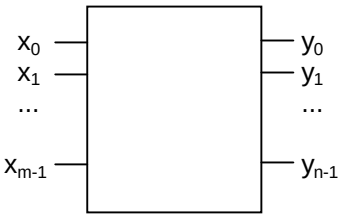


4) Treiberverstärker für kleine Lasten / Treiber für Leistungverstärker
Beispiel: Ansteuerung für LED's



Schaltnetze (ohne Speicher)

m – Eingangvariablen
n – Ausgangsvariablen



F – Funktionsbündel mit n Funktionen

$$\begin{cases} y_0 = y_0\,(x_0, x_1, x_2, \ldots, x_{m-1}) \\ y_1 = y_1\,(x_0, x_1, x_2, \ldots, x_{m-1}) \\ y_2 = y_2\,(x_0, x_1, x_2, \ldots, x_{m-1}) \\ \ldots \end{cases}$$

Stephan Tost
st_tost@hotmail.com

Funktionstabelle mit m = 3 und n = 4

i	x_2	x_1	x_0	y_0	y_1	y_2	y_3	Produktterme	Summenterme	y_0	y_0
0	0	0	0	1	0	0	1	$p_0 = !x_2\ !x_1\ !x_0$	$s_0 = x_2 + x_1 + x_0$	p_0	
1	0	0	1	0	1	0	0	$p_1 = !x_2\ !x_1\ x_0$	$s_1 = x_2 + x_1 + !x_0$		s_1
2	0	1	0	0	1	0	1	$p_2 = !x_2\ x_1\ !x_0$	$s_2 = x_2 + !x_1 + x_0$		s_2
3	0	1	1	0	1	0	0	$p_3 = !x_2\ x_1\ x_0$	$s_3 = x_2 + !x_1 + !x_0$		s_3
4	1	0	0	1	0	1	1	$p_4 = x_2\ !x_1\ !x_0$	$s_4 = !x_2 + x_1 + x_0$	p_4	
5	1	0	1	1	0	1	0	$p_5 = x_2\ !x_1\ x_0$	$s_5 = !x_2 + x_1 + !x_0$	p_5	
6	1	1	0	1	0	0	1	$p_6 = x_2\ x_1\ !x_0$	$s_6 = !x_2 + !x_1 + x_0$	p_6	
7	1	1	1	0	0	1	0	$p_7 = x_2\ x_1\ x_0$	$s_7 = !x_2 + !x_1 + !x_0$		s_7

$y_0 = p_0 + p_4 + p_5 + p_6$
$= !x_2\ !x_1\ !x_0 + x_2\ !x_1\ !x_0 + x_2\ !x_1\ x_0 + x_2\ x_1\ !x_0$

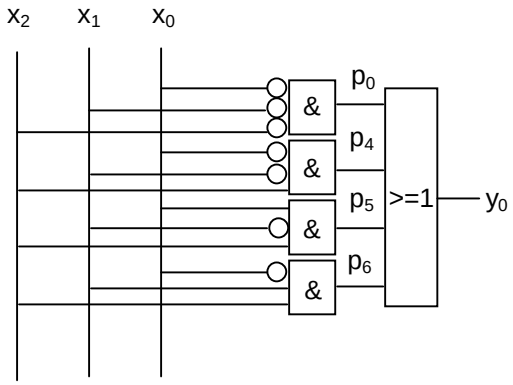
Dies ist die **disjunktive Normalenform (DNF)** für y_0.
Außerdem wird diese Form als **kanonisch** bezeichnet, d. h. alle Terme sind in ihr enthalten.

Die kanonische **konjuktive Normalenform (KNF)** wird aus den Summentermen gebildet.

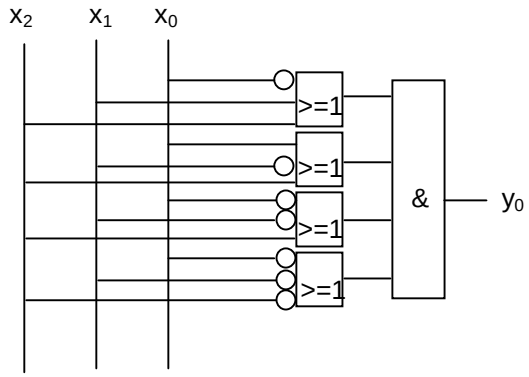
$y_0 = s_1 + s_2 + s_3 + s_7$
$= (x_2 + x_1 + !x_0) * (x_2 + !x_1 + x_0) * (x_2 + !x_1 + !x_0) * (!x_2 + !x_1 + !x_0)$

Schaltbild:

DNF



KNF



Schaltungssysthese

Pflichtenheft → Funktionstabelle

Produktterme
disjunktive Normalenform

Summenterme
konjunktive Normalenform

Schaltplan
Minimalisierung
Schaltbild

Stephan Tost
st_tost@hotmail.com

Funktionstabellenaufstellung:

Mit Produkttermen:
wenn für a **0** steht → **!a**
wenn für a **1** steht → **a**

Mit Summentermen:
wenn für a **0** steht → **a**
wenn für a **1** steht → **!a**

Funktionsgleichung für die Schaltung aufstellen :
- über Pflichteinsen (Produktterme)
  nur wenn gefordertes Ergebnis **1** Produktterm in Summengleichung einfügen (ODER-Verknüpfung)
- über Pflichtnullen (Summenterme)
  nur wenn gefordertes Ergebnis **0** Summenterm in Produktgleichung einfügen (UND-Verknüpfung)

Produktterme werden auch MinTerme genannt
Summenterme werden auch MaxTerme genannt
gewünschtes Ergebnis (Vorgabe)

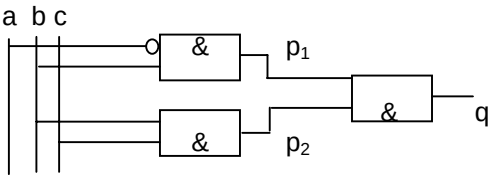
Beispiel:

i	a	b	Pi	Si	q
0	0	0	!a ∧!b	a ∨ b	1
1	0	1	!a ∧ b	a ∨ !b	0
2	1	0	a ∧!b	!a ∨ b	1
3	1	1	a ∧ b	!a ∨ !b	0

q = p_0 ∨ p_2 = (!a ∧!b) ∨ (a ∧!b) → aus Pi
q = s_1 ∧ s_3 = (a ∨ !b) ∧ (!a ∨ !b) → aus Si

Schaltungsanalyse

Beispiel:



p_1 = !a ∧ b
p_2 = b ∧ c

q = (!a ∧ b) ∨ (b ∧ c)

i	a	b	c	!a b	b c	q
0	0	0	0	0	0	0
1	0	0	1	0	0	0
2	0	1	0	1	0	**1**
3	0	1	1	1	1	**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	0	1	**1**

Lösung sind also die Zeilen 2, 3 und 7.

Um die kanonische Form zu erhalten kann man die fehlenden Glieder in der Form (x + !x) zu dem jeweiligen Term mittels UND Verknüpfung hinzufügen, da (x + !x) = 1 ist, und so den Term nicht beeinflusst.

q = !a b + b c
  = (!a b) (c + !c) + (b c) (a + !a)
  = !a b c + !a b !c + b c a + b c !a
  = 2!a b c + !a b !c + a b c
  = !a b c + !a b !c + a b c da die Information (!a b c) redundant vorhanden war

Stephan Tost
st_tost@hotmail.com

Bemessung der Länge von Schaltgliedern

Die Funktionenlänge **l** beschreibt die Gesamtanzahl der Eingänge aller Schaltelemente in einem Schaltbild.
Beispiel:

!ab + bc = q → l = 6
2 UND Verknüpfungen 2 Eingänge und eine ODER Verknüpfung mit 2 Eingängen

!abc + !ab!c + abc = q → l = 12
3 UND Verknüpfungen mit je 3 Eingängen und eine ODER Verknüpfung mit 3 Eingängen

Karnaugh Veitch Diagramm

Die Funktionstabelle wird hierbei in ein Diagramm umgewandelt, um redundante Informationen zu eliminieren und die Aufstellung der Funktionsgleichung zu vereinfachen.

i	B	A	Q1	Q2
0	0	0	1	**1**
1	0	1	0	**1**
2	1	0	0	0
3	1	1	1	0

Die einzelnen Zellen werden gemäß der x - !x Informationen mit laufenden Nummern bezeichnet.
Beispiel: erste Zelle: **!a !b** → bei i = **0**

!A	A	!A
0 1	3 2	
!B	B	

Dann werden die geforderten Ergebnisse der Schaltung in die Tabelle übertragen.
(Nur die Pflichteinsen / -nullen nötig, je nachdem ob man Summen- oder Produktgleichung haben will.)
Beispiel: für i = **0** ist **Q1 = 1**, also: Zelle 0 → **1**

Für Q1:

!A	A	!A
0 **1** 1	3 **1** 2	
!B	B	

$q_1 = p_0 + p_3 = !a\ !b + a\ b = a \leftarrow|\rightarrow b$

Für Q2:

!A	A	!A
0 **1** 1 **1**	3 2	
!B	B	

$q_2 = p_0 + p_1 = !a\ !b + a\ !b = !b\ (!a + a) = !b$

Stephan Tost
st_tost@hotmail.com

Für drei Variablen:

i	C	B	A	Q1	Q2
0	0	0	0	1	0
1	0	0	1	0	0
2	0	1	0	1	0
3	0	1	1	1	**1**
4	1	0	0	1	0
5	1	0	1	1	**1**
6	1	1	0	0	**1**
7	1	1	1	0	0

Für Q1:

	!A	A		!A
!C	0 **1**	1	3 **1**	2 **1**
C	4 **1**	5 **1**	7	6
	!B		B	

Wenn man nun die Gleichung aufstellt, enthält diese redundante Informationen. Um diese aus der Gleichung zu halten, kann man Kombinationen in denen x !x auftaucht zusammenfassen, und die negiert erscheinenden Terme weglassen (da ja (x $\vee$!x) = 1 ist).

Beispiel:

$q_1 = p_0 + p_2 + p_3 + p_4 + p_5$
q_1 = !a !b !c + a b !c + !a b !c + !a !b c + a !b c

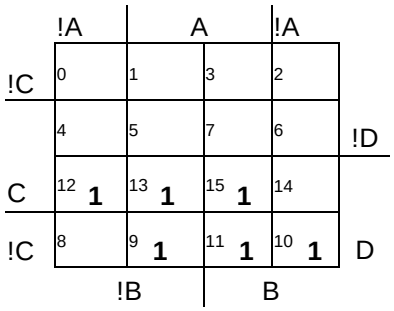
Man kann zusammenfassen: Zelle 3 **(a b !c)** und Zelle 2 **(!a b !c)** zu **(b !c)**
Man kann weiterhin zusammenfassen: Zellen (0, 2), (3, 2), (0, 4), (4, 5)

Man erhält so:
q_1 = !a !c + b !c + !b c + !a !b
= !a !c + b !c + !b c bzw.
= b !c + !b c + !a !b, da jede Zelle nur einmal in der Gleichung vorkommen muß!

Stephan Tost
st_tost@hotmail.com

Beispiel mit vier Variablen:

i	D	C	B	A	Q1
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	**1**
10	1	0	1	0	**1**
11	1	0	1	1	**1**
12	1	1	0	0	**1**
13	1	1	0	1	**1**
14	1	1	1	0	0
15	1	1	1	1	**1**

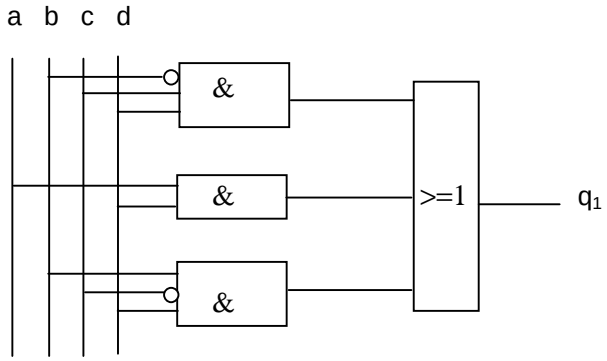


Hier kann ein Viererblock gebildet werden, der sich durch (a d) darstellen läßt.

q_1 = !b c d + a d + b !c d

Diese Darstellungsweise ist natürlich auch für die Pflichtnullen möglich.

Schaltbild:



Stephan Tost
st_tost@hotmail.com

Ampelschaltung

Ablauf:

0 A_grün F_rot
→ Taste

1 A_gelb F_rot
→ Taste

2 A_rot F_grün
→ Taste

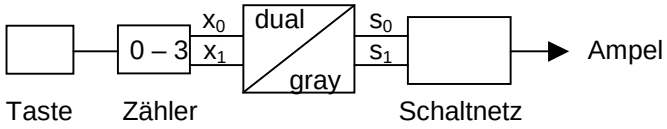
3 A_rotgelb F_rot
→ Taste

Codierung:

'1' – Lampe leuchtet
'0' – Lampe leuchtet nicht

i	x1	x0	s1	s0	A_grün	A_gelb	A_rot	F_grün	F_rot
0	0	0	0	0	1	0	0	0	1
1	0	1	0	1	0	1	0	0	1
2	1	0	1	1	0	0	1	1	0
3	1	1	1	0	0	1	1	0	1
dez.	dual		gray-code						

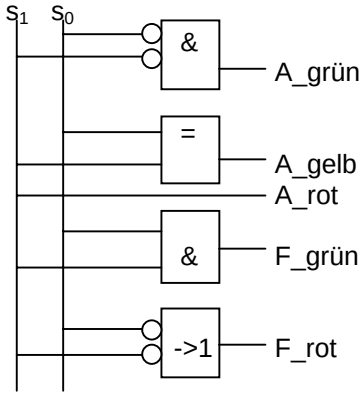
$A\_grün = !s_1\ !s_0$
$A\_gelb = !s_1\ s_0 + s_1\ !s_0 = s_1 \leftarrow|\rightarrow s_0$
$A\_rot = s_1\ s_0 + s_1\ !s_0 = s_1$
$F\_grün = s_1\ s_0$
$F\_rot = !s_1\ !s_0 + !s_1\ s_0 + s_1\ !s_0 = !s_1 + !s_0$



Die Signale aus dem Schaltnetz werden über einen Leistungsverstärker an die Lampen in der Ampel geleitet. Das Schaltnetz hat somit fünf Ausgänge.

Gray-Code
Der Gray-Code ist hier eine Codierung bei der pro Schritt immer nur ein Bit geändert werden muß.

Ampelschaltbild

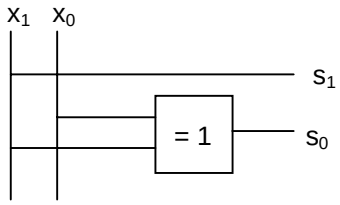


Stephan Tost
st_tost@hotmail.com

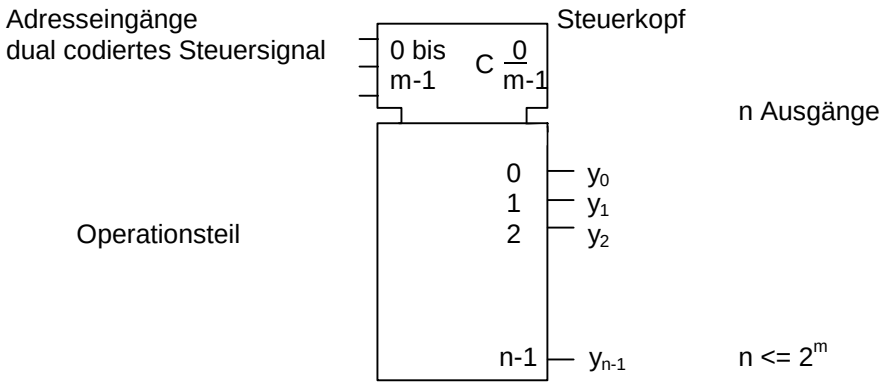
Codewandler

$s_1 = x_1 !x_0 + x_1 x_0 = x_1 (!x_0 + x_0) = x_1 * 1 = x_1$

$s_0 = !x_1 x_0 + x_1 !x_0 = x_1 \leftarrow|\rightarrow x_0$ → Antivalenz



Adressdekoder (1 aus n Dekoder)



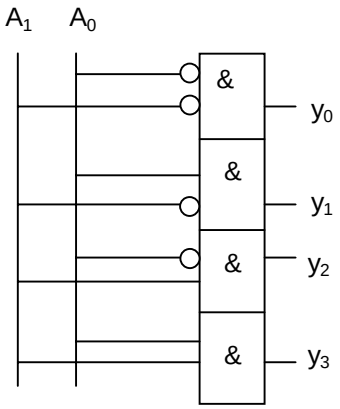
Genau eine Ausgangsleitung wird auf ‚1' gesetzt, nämlich diejenige deren Portadresse der Eingangsadresse entspricht.

Beispiel: 1 aus 4 Dekoder (dient zur Ansteuerung von (in diese Fall) vier Adressen [je immer nur eine])

i	A_1	A_0		y_0	y_1	y_2	y_3
0	0	0		1			
1	0	1			1		
2	1	0				1	
3	1	1					1

Jeweils der Ausgang ist auf ‚1' gesetzt dessen Nummer der Dualzahl des Eingangswertes entspricht.
Beispiel: Zeile 2: Eingang: 1 0 entspricht dezimal 2 → Ausgang Nr. 2 (y_2) wird angesprochen usw.

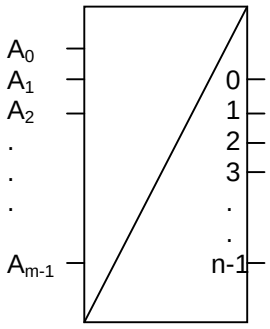
Schaltbild:



Stephan Tost
st_tost@hotmail.com

1 aus n Dekoder - setzt genau einen Ausgang auf „1“

m Adressen max. 2^m Ausgänge

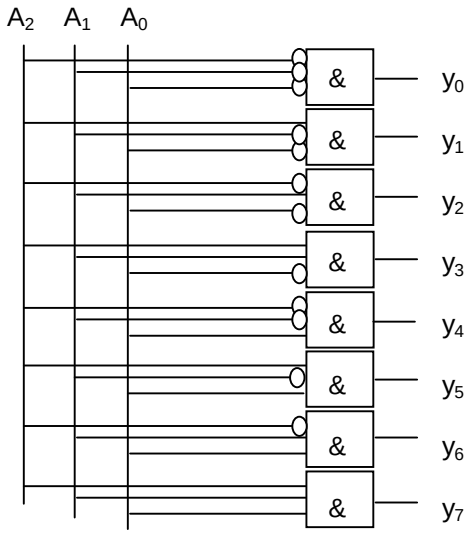


m = 3 → n = 8

i		A_2	A_1	A_0		y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0		0	0	0		1							
1		0	0	1			1						
2		0	1	0				1					
3		0	1	1					1				
4		1	0	0						1			
5		1	0	1							1		
6		1	1	0								1	
7		1	1	1									1

Dualcode

<u>Schaltbild:</u>



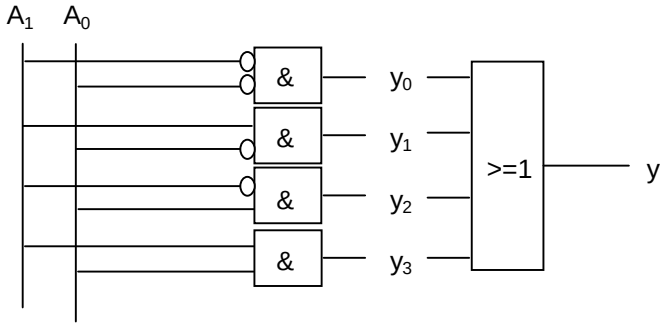
Stephan Tost
st_tost@hotmail.com

Multiplexer

- haben mehrere Dateneingänge und einen Ausgang
- dienen zur Übertragung von mehreren Signalen auf einer Leitung

Hierbei wird mit Hilfe der Adressleitungen (Zähler) ein Takt erzeugt, der jedem Signal ein Zeitfenster zur Verfügung stellt in dem dann das Signal an den Ausgang durchgeschaltet wird.

Schaltbild:

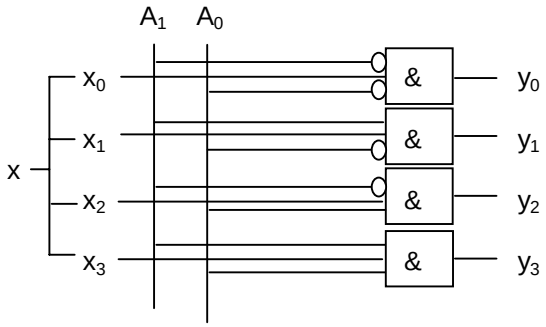


Demultiplexer

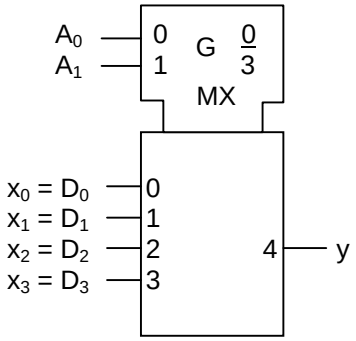
- haben einen Dateneingang und mehrere Ausgänge
- bilden die Umkehrfunktion des Multiplexers

Hierbei muß der Takt der Adressleitungen genau dem der Codierung (s. Multiplexer) entsprechen, damit auch jedes Signal wieder zu der Ausgangsleitung durchgeschaltet wird zu der es gehört.

Schaltbild:



Schaltbild Multiplexer



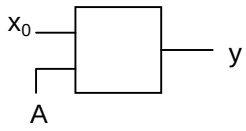
2 Adressleitungen
4 Eingänge

in G Notation (Multiplexer) mit Dateneingängen 0-3

Stephan Tost
st_tost@hotmail.com

weitere Beispiele:

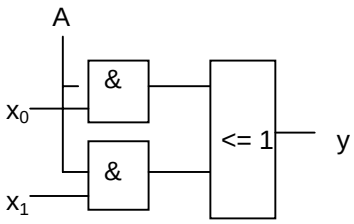
1 : 1 Multiplexer



A	x	y
0	0	0
0	1	0
1	0	0
1	1	1

Wenn an A ‚1' anliegt, das Signal also durchgeschaltet ist, ist y gleich x.

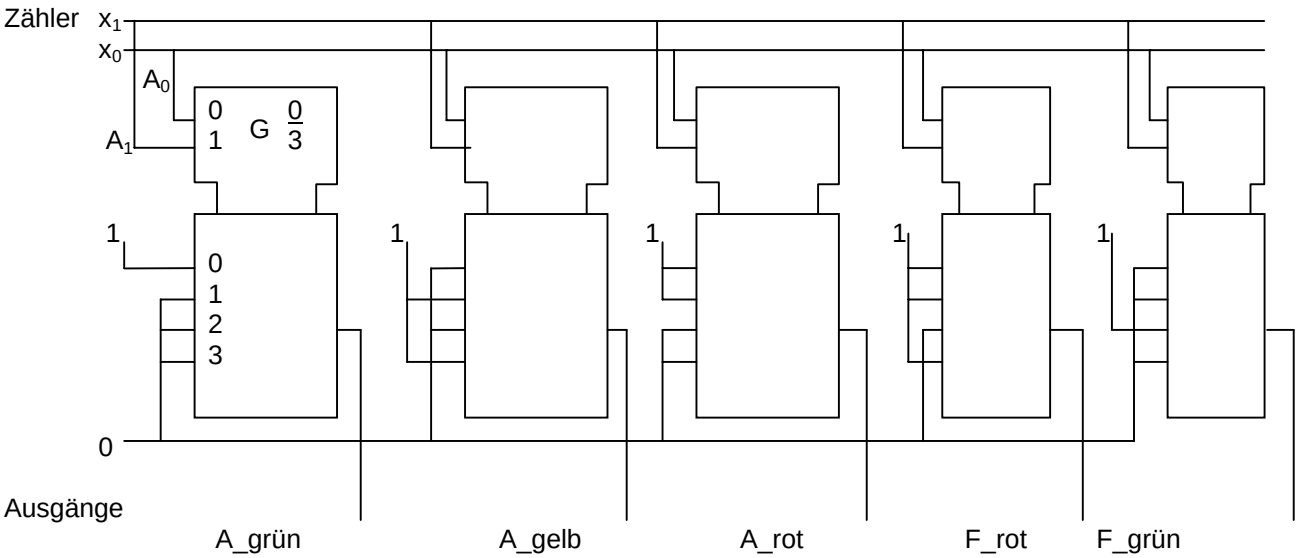
2 : 1 Multiplexer



4 : 1 Multiplexer → siehe Multiplexer

Lösung Ampelschaltung

mit fest verdrahteten Multiplexern



Ausgänge

über

Leistungsverstärker zu den Lampen der Ampel

Stephan Tost
st_tost@hotmail.com

Sieben Segment Anzeige

Für jedes Segment gibt es einen Multiplexer
- mit 4 Eingängen, da mit drei Eingängen nicht 10 Ziffern symbolisiert werden könne
- die Dateneingänge sind festverlötet, je nachdem welches Segment für jede Ziffer benötigt wird, wird bei der Durchschaltung des Dateneingangs eine ‚1' oder ‚0' an den Ausgang gelegt

i	Segment a	Segment b
0	1	1
1	0	1
2	1	1
3	1	1
4	0	1
5	1	0
6	1	0
7	1	1
8	1	1
9	1	1

Addierer

Ein-Bit-Addition

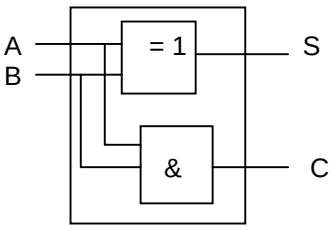
A = {0, 1 A + B = Z
B = {0, 1

S = !AB + A!B = A ←|→ B

A	B	Z	Carry	Summe
0	0	00	0	0
0	1	01	0	1
1	0	01	0	1
1	1	10	1	0

Halbaddierer [HA]

C_0 – Carry out



Zwei-Bit-Addition

A = 10
B = 11

A + B:

		1	0
		1	1
C	1	0	
E	1	0	1

C_0 Summe

Stephan Tost
st_tost@hotmail.com

Volladdierer

HA: {S = !AB + A!B = A ←|→ B, C = AB
VA: {S = !AB!C_{in} + A!B!C_{in} + !A!BC_{in} + ABC_{in} = (A ←|→ B)!C_{in} + (A ←|→ B)C_{in}
C_{out} = AB!C_{in} + !ABC_{in} A!BC_{in} + ABC_{in} = AB + (A ←|→ B)C_{in}

A	B	C_{in}	S	C_{out}
0	0		0	0
0	1		1	0
1	0		1	0
1	1		0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

A	B	C_{in}	S_1	C_{o1}	S_2	C_{o2}	C_{out}
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
1	0	0	1	0	1	0	0
1	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0
0	1	1	1	0	0	1	1
1	0	1	1	0	0	1	1
1	1	1	0	1	1	0	1

<u>Vier-Bit-Addierer</u>

A 1011
B 1111
C 1
S 11010

nötig sind: ein HA und drei VA

Stephan Tost
st_tost@hotmail.com