



Grundlagen der Digitaltechnik

Im Fach „Steuerungstechnik“ haben Sie bereits einen kleinen Einblick in die moderne elektronische Steuerungstechnik erhalten. Diese ist gekennzeichnet durch den Übergang von der **elektromechanischen Steuerung** zur **programmierbaren, elektronischen Steuerung**. Wie hat sich dieser Wandel in der Steuerungstechnik vollzogen – was waren seine Ursachen? Die ursprünglich verwendeten Bauelemente wie Schütz und Relais (elektromagnetisch betätigte mechanische Schalter) genügten schon bald nicht mehr den Anforderungen der halb- oder vollautomatisch gesteuerten Prozesse. Sie erwiesen sich als unzuverlässig, störanfällig und zu langsam.

Mit der Entwicklung der **Halbleiterbauelemente** wurden der Industrie schnelle, berührungslos arbeitende, elektronische Schalter zur Verfügung gestellt. Das Zeitalter der kontaktlosen, ruhenden (statischen) Steuerungen begann. Die Wünsche der Anwender an Funktionssicherheit und Unempfindlichkeit konnten erfüllt werden.

Von der Halbleiterindustrie werden komplette elektronische Bausteine mit bestimmten **logischen Funktionen** hergestellt, aus denen sich die Steuerung modular zusammensetzen läßt. Solche **statischen Steuerungssysteme** haben heute noch große technische Bedeutung: sie gehören zur Gruppe der verbindungsprogrammierten Steuerungen (VPS). Selbst komplizierte Steuerungen dieser Art erfordern nur verhältnismäßig wenige Bausteine unterschiedlicher Funktionen.

Mit dem Aufkommen des **Mikroprozessors** begann eine neue Entwicklungsstufe für elektronische Steuerungen. Die Logik der elektronischen Funktionsbausteine konnte jetzt mit einem Programm nachgebildet werden. Zur Festlegung des Steuerungsprogramms war keine Verdrahtung mehr erforderlich; es ließ sich nun in Form von Anweisungen in einem Halbleiterspeicher eingeben. Damit waren die Voraussetzungen für den Erfolg der speicherprogrammierten Steuerungen (SPS) gegeben. Der Anwendungsbereich der SPS erstreckt sich nicht nur auf umfangreiche und komplexe Steuerungsanlagen. Im Gegenteil, der Neuentwicklung von preisgünstigen Kleinststeuerungen wird sehr große Aufmerksamkeit geschenkt.

Die bei der VPS in der Verdrahtung steckende Logik muß bei der SPS in ein Programm eingebaut werden. Die Formulierung der logischen Verknüpfungen, die Beschreibung der logischen Eigenschaften von Schaltungen und die Regeln zur Minimierung von Schaltfunktionen haben für die VPS und die SPS nahezu die gleiche Bedeutung. Ihre Gesetzmäßigkeiten sind im Fachgebiet **Digitaltechnik** zusammengefaßt. Mit den Grundlagen der Digitaltechnik werden wir uns noch ausführlich beschäftigen.

Logische Grundfunktionen

Für verbindungsprogrammierte Steuerungen stellt die Industrie integrierte elektronische Bauelemente zur Verfügung, mit denen eine



Signalverarbeitung nach bestimmten logischen Gesichtspunkten erfolgt. Für die SPS sind die logischen Verknüpfungen zu programmieren. In der Signalverarbeitung besteht zwischen der Hardwarelösung (VPS) und der Softwarelösung (SPS) kein Unterschied.

Bei den **logischen Verknüpfungsfunktionen** wirkt sich der Zustand der Eingangssignale unmittelbar auf den Signalzustand am Ausgang aus. Verknüpfungsfunktionen haben keine speichernden Eigenschaften. Die Eingangssignale werden in diesem Zusammenhang auch als Eingangsvariablen, die Ausgangssignale als Ausgangsvariablen bezeichnet.

Beschreibung logischer Verknüpfungen

Für den Entwurf von Schaltungen, die Entwicklung von Programmen und für das Lesen vorgegebener Schaltungen müssen Vereinbarungen bestehen, mit denen die logische Verknüpfung der Signale beschrieben wird. Die Beschreibung kann folgendermaßen ausgeführt werden:

1. **Durch Worte**, verbale Beschreibung genannt.
2. **Durch ein Schaltzeichen**. Damit wird die logische Funktion symbolisch beschrieben. Wir verwenden die Symbole nach DIN 40700, Blatt 14.
3. **Durch die Schaltfunktion** in Form einer Funktionsgleichung. Mit Hilfe von mathematischen Zeichen wird die Art der Signalverknüpfung als Gleichung dargestellt. In der Gleichung wird die Abhängigkeit des Ausgangssignals A von den Eingangssignalen E und deren logischen Verknüpfung miteinander angegeben. Mit dieser Methode lassen sich komplette Schaltungen beschreiben. Nach der Schaltfunktion kann unmittelbar die Programmierung erfolgen.
4. **Durch die Funktionstabelle**, die auch als Wahrheitstabelle bezeichnet wird. Sie gibt den von der logischen Funktion abhängigen Zusammenhang zwischen den binären Zuständen der Eingangssignale und dem daraus resultierenden Zustand des Ausgangssignals wieder. Die Funktionstabellen eignen sich gut zur Analyse von Schaltungen und zum Test von Steuerungsprogrammen.
5. **Durch den Kontaktplan**. Der Kontaktplan hat eine große Ähnlichkeit mit dem Stromlaufplan. Mit den Kontaktsymbolen werden Eingangssignale gekennzeichnet. Andere Symbole bezeichnen **Ausgänge** und **Merker**. Die Strompfade sind waagrecht untereinander zu zeichnen.
6. **Durch das Funktionsdiagramm als Signal-Zeit-Diagramm**. Es ist ein Funktionsdiagramm. In ihm sind die Signalzustände der Eingangssignale und des davon abhängigen Ausgangssignals graphisch über der Zeit dargestellt.

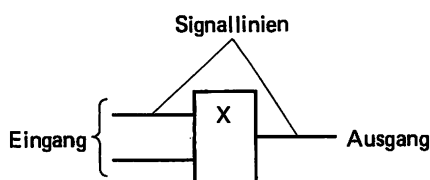


Bild B2.1
Grundform des Schaltzeichens für
Verknüpfungsglieder.

Wir wollen nun die Beschreibungsmöglichkeiten zunächst auf die signalverarbeitenden Grundfunktionen anwenden.

Tabelle B3.1: Logische Grundfunktionen zur Signalverknüpfung

	Grundfunktionsglied	Symbol	Funktions-tabelle	Verbale Beschreibung Die Variable A am Ausgang nimmt nur dann den Wert...	Funktionsdiagramm	Schaltfunktion	Kontaktplan															
		1	2	3	4	5	6															
1	UND-Glied		<table><tr><th>E2</th><th>E1</th><th>A</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	E2	E1	A	0	0	0	0	1	0	1	0	0	1	1	1	...1 an, wenn die Variablen an allen Eingängen den Wert 1 haben		$A = E1 \wedge E2$	
E2	E1	A																				
0	0	0																				
0	1	0																				
1	0	0																				
1	1	1																				
2	ODER-Glied		<table><tr><th>E2</th><th>E1</th><th>A</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	E2	E1	A	0	0	0	0	1	1	1	0	1	1	1	1	...1 an, wenn an mindestens einem Eingang die Variable den Wert 1 hat		$A = E1 \vee E2$	
E2	E1	A																				
0	0	0																				
0	1	1																				
1	0	1																				
1	1	1																				
3	NICHT-Glied		<table><tr><th>E</th><th>A</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	E	A	0	1	1	0	...0 an, wenn die Variable am Eingang den Wert 1 hat		$A = \bar{E}$										
E	A																					
0	1																					
1	0																					
4	NAND-Glied		<table><tr><th>E2</th><th>E1</th><th>A</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	E2	E1	A	0	0	1	0	1	1	1	0	1	1	1	0	...0 an, wenn die Variablen an allen Eingängen den Wert 1 haben		$A = \bar{E1} \wedge \bar{E2}$	
E2	E1	A																				
0	0	1																				
0	1	1																				
1	0	1																				
1	1	0																				
5	NOR-Glied		<table><tr><th>E2</th><th>E1</th><th>A</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	E2	E1	A	0	0	1	0	1	0	1	0	0	1	1	0	...0 an, wenn an mindestens einem Eingang die Variable den Wert 1 hat		$A = \bar{E1} \vee \bar{E2}$	
E2	E1	A																				
0	0	1																				
0	1	0																				
1	0	0																				
1	1	0																				

Grundfunktionen UND, ODER und NICHT

In Tabelle B3.1 sind die Beschreibungen für die Grundfunktionen zusammengestellt. Die angegebenen Gesetzmäßigkeiten gelten auch für Funktionen mit mehr als zwei Eingangsvariablen.

Nach Bild B2.1 ist die **Grundform des Schaltzeichens** ein Rechteck. Die Signallinien für Eingänge und Ausgang sind an den gegenüberliegenden Seiten angebracht. Für X ist in das Kästchen das **Verknüpfungssymbol** einzutragen. Mit einem Kreis am Ein- oder Ausgang des Schaltzeichens kann die Umkehrung des Signalzustands, auch **Signalumkehr** oder **Negierung** genannt, symbolisiert werden.

Bezeichnung der Grundfunktion	Funktion im Rechteck
UND-Funktion oder Konjunktion	$\&$
ODER-Funktion oder Disjunktion	≥ 1
NICHT-Funktion oder Negation	1 (mit Kreissymbol am Ein- oder Ausgang)

Mit diesen drei Grundfunktionen lassen sich alle beliebigen, auch komplizierte logische Verknüpfungen beschreiben. Schaltungen, die technisch nur aus derartigen Verknüpfungen bestehen, nennt man

Schaltnetze. Sie enthalten keine Speicherfunktionen. Schaltungen mit Verknüpfungs- und Speicherfunktionen bezeichnen wir als **Schaltwerke**.

Die Funktionstabelle und die verbale Beschreibung (Tabelle B3.1, Spalten 2 und 3) sagen Ihnen, wie die logische Funktion die Eingangsvariablen zur Ausgangsvariablen verknüpfen muß. Diese Verknüpfungsregeln sollten Sie sich unbedingt einprägen. Mit Hilfe folgender Merksätze ist dies leicht möglich:

UND-Funktion: Die Ausgangsvariable A nimmt nur dann den Zustand 1 an, wenn **alle** Eingangsvariablen den Zustand 1 haben.

ODER-Funktion: Die Ausgangsvariable nimmt nur dann den Zustand 1 an, wenn **mindestens eine** Eingangsvariable den Zustand 1 hat.

NICHT-Funktion: Die Ausgangsvariable nimmt nur dann den Zustand 1 an, wenn die Eingangsvariable den Zustand 0 hat. Es entsteht eine **Signalumkehr**.

Das Funktionsdiagramm (auch als Signal-Zeit-Diagramm bezeichnet) entspricht der grafischen Auswertung der Funktionstabelle.

Nun wollen wir die logischen Signalverknüpfungen der Grundfunktionen durch Schaltfunktionen in Form von **Funktionsgleichungen** beschreiben. Die in Tabelle B4.1 zusammengestellten Symbole sind Kennzeichen für die logischen Operationen.

Tabelle B4.1: Operationskennzeichen

Benennung der Verknüpfung	Operationskennzeichen
UND-Verknüpfung (AND-Operation)	\wedge
ODER-Verknüpfung (OR-Operation)	\vee
Signalumkehr	– (über dem Zeichen)

Mit diesen Kennzeichen sind in der Spalte 5 der Tabelle B3.1 die Schaltfunktionen aufgestellt. Wir werden diese Formulierungen zur Programm Vorbereitung und auch zur Programmierung oft anwenden.

Spalte 6 der Tabelle B3.1 enthält die **Kontaktplandarstellung**. Die Bedeutung dieser Kontaktsymbole entnehmen Sie bitte der Tabelle B5.1. Mit diesen Kontaktsymbolen werden die logischen Verknüpfungen nachgebildet. Dabei gelten die folgenden Regeln:

Die UND-Verknüpfung entspricht einer **Reihenschaltung** von Kontaktsymbolen, die ODER-Verknüpfung einer **Parallelschaltung** von Kontaktsymbolen.

Zusammenfassung

Die Grundfunktionen UND, ODER und NICHT verarbeiten binäre Signale nach bestimmten logischen Gesetzmäßigkeiten; sie haben keine **speichernden Eigenschaften**. Der Zustand der Ausgangsvariablen wird von den momentanen Signalzuständen der Eingangsvariablen bestimmt.

Alle logischen Eigenschaften der Verknüpfungsfunktionen kann man als verbale Beschreibung, Schaltsymbol, Schaltfunktion, Funktionstabelle, Kontaktplan und Signal-Zeit-Diagramm wiedergeben.

Beispiel B5.1

Drei Meldesignale sind mit **UND** zu verknüpfen. Bei $E1=1$ UND $E2=0$ UND $E3=1$ soll das Ausgangssignal $A=1$ sein. Führen Sie dieses Beispiel auch praktisch mit Ihrem System durch.

Lösung:

Die UND-Verknüpfung erzeugt nur dann $A=1$, wenn alle Eingangsvariablen den Zustand 1 haben. Signal E2 muß deshalb durch eine

Tabelle B5.1: Symbole für die Kontaktplandarstellung

Bezeichnung	Stromlaufplan DIN 40713	Kontaktplan DIN 19239	Bemerkung
Eingang als Schließer			Der Kontakt ist "geschlossen", wenn der dazugehörige Operand E Signalzustand 1 führt
Eingang als Öffner			Der Kontakt ist "geschlossen", wenn der dazugehörige Operand E den Signalzustand 0 führt
Erregerspule bzw. Ausgang			Der zur Spule gehörende Operand A führt Signalzustand 1, wenn im Strompfad Strom fließt
Ausgang			Der zur Spule gehörende Operand A führt Signalzustand 0, wenn im Strompfad Strom fließt

Signalumkehr vom Zustand 0 in Zustand 1 gebracht werden. Für das Ausgangssignal gilt deshalb die Schaltfunktion:

$$A = E1 \wedge \overline{E2} \wedge E3$$

Man kann die Lösung durch Einsetzen der Signalzustände überprüfen:

$$1 = 1 \wedge \overline{0} \wedge 1 = 1 \wedge 1 \wedge 1$$

Ergebnis: Nur die Signalkombination $E1=1, E2=0, E3=1$ oder vereinfacht ausgedrückt $E1, E2, E3=1, 0, 1$ ergibt $A=1$. Bei allen anderen Kombinationen bleibt A im Zustand 0. Schaltung und Kontaktplan zeigen Bild B 6.1.

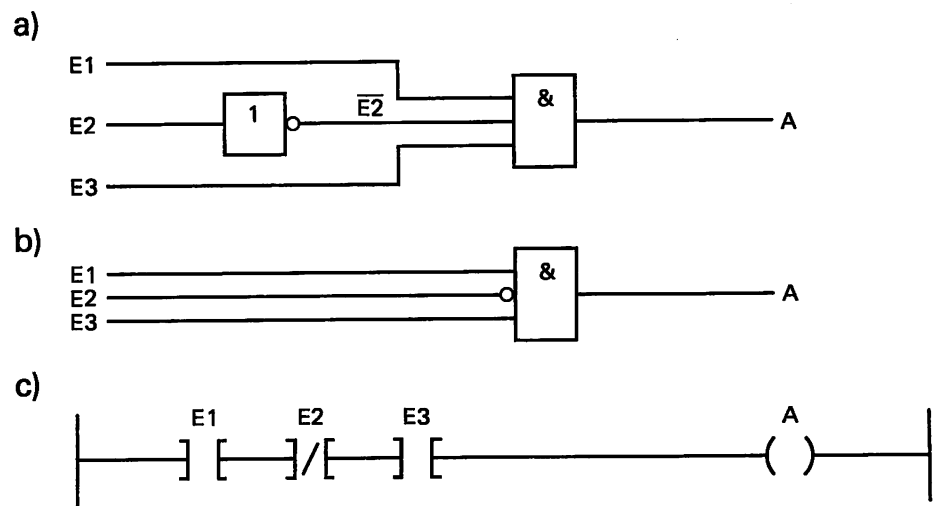


Bild B 6.1
Schaltung zu Beispiel B 6.1
a) ausführliche Schaltung
b) vereinfachtes Schaltzeichen
von a)
c) Kontaktplan.

Beispiel B 6.1

Die beiden Variablen $E1, E2$ sind mit ODER zu verknüpfen. Bei $E1, E2=0, 1$ soll die Ausgangsvariable $A=0$ sein.

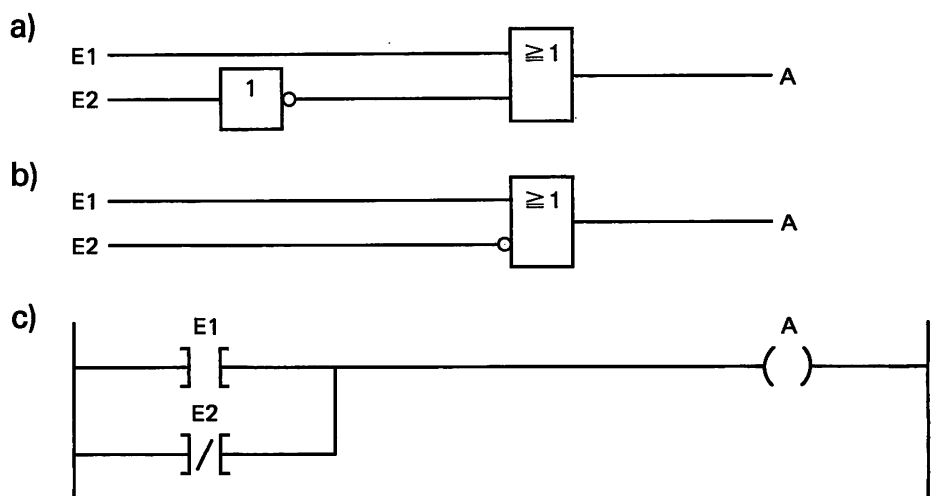


Bild B 6.2
Schaltung zu Beispiel B 6.1.
a) ausführliche Schaltung
b) vereinfachtes Schaltzeichen
von a)
c) Kontaktplan.

Lösung:

Sie haben gelernt, daß bei einer ODER-Verknüpfung nur dann $A=0$ ist, wenn alle Eingangsvariablen den Zustand 0 haben. Das Signal E2, das ja den Signalzustand 1 hat, ist deshalb vor der Verknüpfung zu negieren. Es ist nun eine Schaltung bzw. ein Kontaktplan nach Bild B 6.2 erforderlich. Die Schaltfunktion lautet: $A=E1\overline{VE2}$

Damit folgt: $0 = 0 \vee \bar{1} = 0 \vee 0$

Standardfunktionen NAND und NOR

Diese Funktionen sind in Tabelle B 3.1 in den Zeilen 4 und 5 aufgeführt. Mit jeder dieser beiden Standardfunktionen kann man die logischen Grundverknüpfungen UND, ODER und NICHT nachbilden. Die Schaltnetze enthalten dann nur noch einen Verknüpfungstyp.

NAND-Verknüpfung: Bild B 7.1 zeigt das Entstehen der NAND-Verknüpfung. Zunächst werden die Eingangsvariablen E1 und E2 UND-verknüpft und zur Ausgangsvariablen A1 zusammengefaßt: $A1=E1\wedge E2$.

Die Ausgangsvariable A entsteht durch **Negation** von A1: $A=\overline{A1}$. Wir erhalten damit eine Schaltfunktion: $A=\overline{E1\wedge E2}$ (siehe auch Tabelle B 3.1, Zeile 4, Spalte 5).

Merken Sie sich bitte:

Die NAND-Funktion verknüpft zuerst die Eingangsvariablen mit UND. Anschließend erfolgt die **Negation** des Ergebnisses der UND-Verknüpfung.

NOR-Verknüpfung: Das Entstehen dieser Verknüpfung zeigt Ihnen Bild B 7.2. Erst verknüpfen wir die Eingangsvariablen mit ODER: $A1=E1\vee E2$. Anschließend erfolgt die Negation des Zwischenergebnisses zu $A=\overline{A1}$. Damit lautet die Schaltfunktion: $A=\overline{E1\vee E2}$. (Siehe auch Tabelle B 3.1, Zeile 5, Spalte 5).

Es gilt der Merksatz:

Die NOR-Funktion verknüpft zuerst die Eingangsvariablen mit ODER. Anschließend erfolgt die **Negation** des Ergebnisses der ODER-Verknüpfung.

Mit der NAND- oder mit der NOR-Funktion kann man, wie bereits angeführt, die **Grundfunktionen** nachbilden. Die erforderlichen Schaltungen sind in Tabelle B 8.1 zusammengestellt. Wenn Sie die Funktionstabellen aufstellen, können Sie selbst die Schaltungen überprüfen.

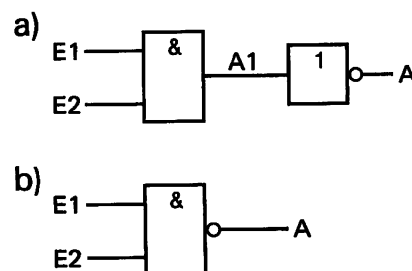


Bild B 7.1
NAND-Funktion
a) aufgelöste Schaltung
b) Schaltzeichen.

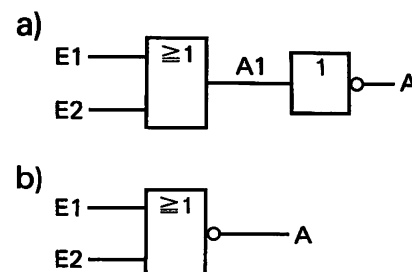


Bild B 7.2
NOR-Funktion
a) aufgelöste Schaltung
b) Schaltzeichen.

Tabelle B 8.1: Grundverknüpfungen mit NAND- und NOR-Funktionen

	Funktion	Schaltungstechnik	
		NAND	NOR
1	NICHT		
2	UND		
3	ODER		

Zusammenfassung

Für manche Anwendungen ist es vorteilhaft, die Grundfunktionen mit einer Standardfunktion nachzubilden. Hierfür eignen sich die NAND- und NOR-Verknüpfungen.

Beispiel B 8.1

Nach Tabelle B 8.1 müßte die Schaltung in Bild B 8.1 eine ODER-Funktion darstellen, die mit NAND-Verknüpfungen ausgeführt ist. Den Nachweis wollen wir durch Aufstellen der Funktionstabelle führen.

Hinweis: X und Y an den Ausgängen von N1 und N2 sind Hilfs-signale. Die Eingänge von N1 und N2 sind zusammengeschaltet. Für N1 gilt beispielsweise: $X = \overline{E1} \wedge E1 = \overline{E1}$. Da an beiden Eingängen die gleiche Eingangsvariable liegt, ist $E1 \wedge E1 = E1$. N1 und N2 bewirken nur eine Umkehr des jeweiligen Eingangssignals.

Lösung:

Die Signale innerhalb der Schaltung sind für alle Kombinationen der Eingangssignale zu ermitteln. Zwei Eingangssignale ergeben $2^2 = 4$ Signalkombinationen.

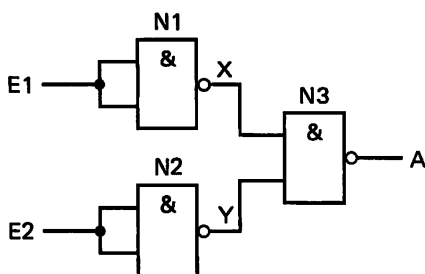


Bild B 8.1
Schaltung zu Beispiel B 8.1.

Tabelle B 8.2: Funktionstabelle zu Beispiel B 8.1

E2	E1	X	Y	A
0	0	1	1	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	1

Für die Ausgangsvariable gilt die Funktion $A = \overline{XAY}$. Vorstehendes Ergebnis entspricht der Wirkung einer ODER-Verknüpfung.

Hat eine Schaltfunktion oder ein Schaltnetz n verschiedene Eingangsvariablen, so lassen sich mit diesen 2^n verschiedene Signalkombinationen bilden.

Z. B. ergeben drei Eingangsvariable $2^3 = 8$ verschiedene Signalkombinationen.

B**9**

Aufgaben B9.1

1. Zeile 2 der Tabelle B 8.1 zeigt die UND-Nachbildung mit NOR-Funktionen. Füllen Sie für diese Schaltung nachstehende Funktionstabelle aus. Wie im letzten Beispiel gezeigt, kann man wieder Hilfssignale vorsehen, X ist E1 und Y ist E2 zugeordnet.

Tabelle B9.1: Funktionstabelle zur Aufgabe B9.1

E2	E1	X	Y	A
0	0			
0	1			
1	0			
1	1			

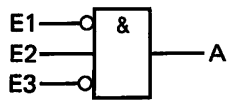
2. Die drei Eingangsvariablen E1, E2 und E3 ergeben die in folgender Funktionstabelle aufgeführten Signalkombinationen; sie werden verschiedenen Verknüpfungsfunktionen zugeführt. Die Schaltfunktionen lauten:

$$A1 = E1 \vee \overline{E2} \quad A2 = \overline{E2} \wedge E3 \quad A3 = \overline{E1} \vee E3 \quad A4 = E1 \vee \overline{E2} \vee E3$$

Tabelle B9.2: Funktionstabelle zur Aufgabe B9.1

E3	E2	E1	A1	A2	A3	A4
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Tragen Sie bitte in die vorbereitete Funktionstabelle die Signalezustände der Ausgangsvariablen ein.

B**10**

E3	E2	E1	A
0	1	0	
0	1	1	
1	1	0	
1	1	1	
0	1	0	

Bild B10.1
Schaltung und Funktionstabelle
zur Aufgabe B9.1, Frage 7.

3. Für die Schaltfunktion $A0 = \overline{E0} \vee \overline{E1} \vee E2$ sind Schaltzeichen und Kontaktplan aufzuzeichnen.
4. Was bedeutet doppelte Negation?
5. Für die logische Aussage $A=1$, wenn E3 Zustand 1 und E1 und E2 Zustand 0 haben, sind Schaltzeichen und Schaltfunktion anzugeben.
6. Eine Logik verknüpft vier voneinander unabhängige Eingangsvariable. Wieviel verschiedene Signalkombinationen können mit diesen gebildet werden?
7. Gegeben ist eine Verknüpfungsfunktion nach Bild B 10.1. Ergänzen Sie die hier nur teilweise ausgefüllte Funktionstabelle.

Die Lösungen dieser Aufgaben finden Sie auf den Seiten F4 und F5.

Analyse und Synthese logischer Schaltnetze

Mit den im ersten Lehrbrief beschriebenen Grund- und Standardverknüpfungen lassen sich alle Schaltungen realisieren, die die Eingangsvariablen nach bestimmten Vorschriften zu Ausgangsvariablen verknüpfen. Zur Programmierung von Schaltungen sind die bisher erarbeiteten Programme der Einzelverknüpfungen nach bestimmten Regeln miteinander zu kombinieren. Die dabei zu beachtenden Regeln stammen größtenteils aus dem Fachgebiet Digitaltechnik. Wir müssen deshalb die bisher bekannten Grundlagen erweitern.

Schaltungen, die nur die Grund- bzw. Standardverknüpfungen enthalten, nennt man **Schaltnetze**. Bei der Bearbeitung logischer Schaltnetze ist zwischen der Analyse und der Synthese zu unterscheiden.

Bei der **Analyse eines Schaltnetzes** ist uns die zu untersuchende bzw. zu programmierende Schaltung vorgegeben. U.a. kann die Aufgabe darin bestehen, für diese Schaltung

- die Schaltfunktion aufzustellen und
- die Funktionstabelle anzufertigen.

Zur Programmerstellung wird die Schaltfunktion benötigt, und zum Programmtest ist die Funktionstabelle erforderlich. Außerdem lassen sich mit Schaltfunktion und Funktionstabelle die Eigenschaften einer vorgegebenen Schaltung ermitteln und interpretieren.

Die **Synthese** eines Schaltnetzes umfaßt dagegen den Entwurf logischer Schaltnetze. Der Entwurf setzt eine konkrete Aufgabenstellung voraus. Sie kann erfolgen durch

- die verbale Beschreibung,
- eine Vorgabe der Funktionstabelle oder
- die Angabe einer Schaltfunktion.

Das Ziel der Synthese ist ein **Schaltungsentwurf**, der in unserem Fall wieder zu einem Programm führt. Schaltung bzw. Programm müssen die Vorgaben der Aufgabenstellung erfüllen.

Analyse logischer Schaltnetze

Den Vorgang einer Analyse zeigt Ihnen der Ablaufplan in Bild B 12.1. In Industriebetrieben ist oftmals ein „verdrahtetes“ Schaltnetz vorgegeben. Unter dieser Bezeichnung fassen wir alle technischen Schaltungsausführungen, auch gedruckte Schaltungen, zusammen. Zunächst müssen Sie diese Schaltung aufnehmen und anschließend mit genormten Symbolen als kontaktlose Schaltung darstellen. Nunmehr kann die Analyse erfolgen. Dazu tragen Sie in die Schaltung die Ein- und Ausgangsvariablen ein. Falls erforderlich, können noch Hilfsvariablen vorgesehen werden. Ein Schaltnetz kann eine oder auch mehrere Ausgangsvariablen haben. Den Ablauf der Analyse erläutern wir an einem einfachen Beispiel.

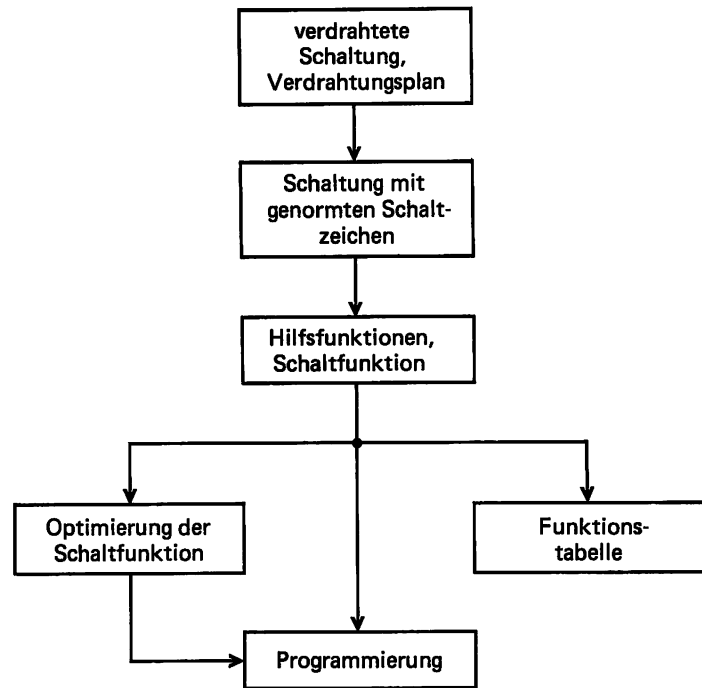
B**12**

Bild B 12.1
Ablaufplan für die Analyse logischer Schaltnetze.

Beispiel B 12.1

Stellen Sie für die Schaltung in Bild B 13.1

- die Schaltfunktion und
- die Funktionstabelle auf.

Die Schaltung enthält nur Grundfunktionen. Binäre Geber (nicht eingezeichnet) erzeugen die Signale E1, E2 und E3, die über Signalleitungen dem Schaltnetz als **Eingangsvariablen** zugeführt werden. Für diese und alle folgenden Darstellungen wird folgendes Bezeichnungsschema vereinbart:

Signalleitung E1 führt das binäre Signal E1, das die beiden Zustände 1 bzw. 0 annehmen kann. Signalleitung E2 führt das binäre Signal E2 usw. Signalleitung und Signal, also binäre Variable, werden gleich bezeichnet. Die gleiche Bezeichnung erhält auch der Eingang des Schaltzeichens, das an die betreffende Signalleitung angeschlossen ist.

Lösung:

- Aufstellen der Schaltfunktion: Wir gehen schrittweise vor.

1. Schritt: Die noch nicht bezeichneten Ausgangsvariablen der Verknüpfungsfunktionen bezeichnen wir mit X. Es sind **Zwischen- oder Hilfsvariablen**, die das Aufstellen der Schaltfunktion erleichtern sollen. Von den Signalleitungen ausgehend haben wir die Hilfsvariablen X1, X2 und X3 in Bild B 13.1 eingezeichnet.

2. Schritt: Für die Hilfsvariablen sind die Schaltfunktionen aufzustellen. Es ist zu ermitteln, wie die Ausgangsvariablen X bzw. A vom Signalzustand der Eingangsvariablen abhängen. Aus Bild B 13.1 können Sie unmittelbar ablesen:

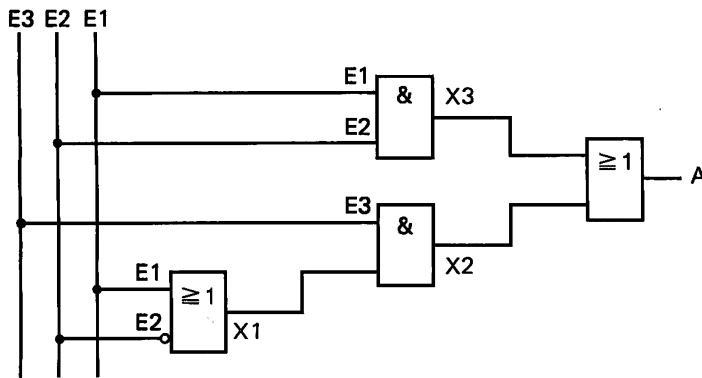


Bild B 13.1
Schaltung zu Beispiel B 12.1.

$$\begin{aligned} X1 &= E1 \vee \overline{E2} & \text{Gl.B 13.1} \\ X2 &= X1 \wedge E3 & \text{Gl.B 13.2} \\ X3 &= E1 \wedge E2 & \text{Gl.B 13.3} \\ A &= X2 \vee X3 & \text{Gl.B 13.4} \end{aligned}$$

In den Bezeichnungen der einzelnen Schaltfunktionen steht „Gl.“ für „Gleichung“.

3. Schritt: Die Hilfsvariablen sind schrittweise durch die zugehörigen Schaltfunktionen zu ersetzen.

In Gleichung B 13.2 wird X1 durch die Funktion von Gl.B 13.1 ersetzt:

$$X2 = (E1 \vee \overline{E2}) \wedge E3 \quad \text{Gl.B 13.5}$$

Wir vereinbaren, daß grundsätzlich jede **Hilfsfunktion**, die anstelle eines Hilfs- oder Zwischensignals steht, in Klammern gesetzt wird. Mathematisch gesehen entstehen zwar manchmal überflüssige Klammerausdrücke. Die Klammern kennzeichnen aber sichtbar das Zusammenwirken der Variablen. Später lernen Sie noch sogenannte **Vorrangregeln** kennen, die das Setzen von Klammern erübrigen.

Nunmehr sind noch in Gleichung B 13.4 die Hilfsvariablen X2 und X3 zu ersetzen. Die Lösung lautet:

$$A = \underbrace{[(E1 \vee \overline{E2}) \wedge E3]}_{X2} \vee \underbrace{(E1 \wedge E2)}_{X3} \quad \text{Gl.B 13.6}$$

Mit dieser Schaltfunktion ist die Schaltung von Bild B 13.1 eindeutig beschrieben. Wie wir Ihnen an einfachen Beispielen schon gezeigt haben (Seite E7), kann eine solche Schaltfunktion direkt in ein Programm umgesetzt werden.

b) Aufstellen der vollständigen Funktionstabelle: Wie schon erwähnt, muß ein Programm daraufhin untersucht werden, ob es auch tatsächlich alle Vorgaben der Aufgabenstellung erfüllt. Für alle

Schaltungen, die **binäre Signale** verarbeiten, ermöglicht die Funktionstabelle einen einfachen Test. Sie läßt sich an Hand der zu programmierenden Schaltung aufstellen und anschließend mit dem Programm experimentell aufnehmen. Beide Tabellen müssen übereinstimmen. Wir werden diesen Vergleich oft anwenden.

Die Funktionstabelle soll unmittelbar nach der Schaltung in Bild B 13.1 aufgestellt werden. Es wird angenommen, daß die Schaltfunktion noch nicht bekannt ist.

B**14**

1. Schritt: Dieser ist identisch mit dem unter a) genannten 1. Schritt. In die Schaltung sind die Hilfsvariablen einzutragen.

2. Schritt: Die Funktionstabelle (Tabelle B 14.1) ist aufzustellen. Alle möglichen Signalkombinationen für die Eingangsvariablen E sind einzutragen.

3. Schritt: Die Schaltfunktionen für die Hilfsvariablen und die Ausgangsvariable sind aufzustellen. Dieser Schritt entspricht dem 2. Schritt unter a). Alle Signalzustände der Hilfsvariablen, die diese Funktionen ergeben, sind in Tabelle B 14.1 einzutragen.

4. Schritt: Aus den Werten der Hilfsvariablen ist der Zustand der Ausgangsvariablen zu berechnen und in die entsprechende Spalte der Tabelle B 14.1 einzutragen.

Die **Hilfsfunktionen** bzw. die **Hilfsvariablen** kennzeichnen logische Zustände innerhalb eines Schaltnetzes. Sie erleichtern die Untersuchung und Analyse von vorgegebenen Schaltnetzen.

Tabelle B 14.1: Funktionstabelle zu Beispiel B 13.1

E3	E2	E1	X1= E1VE2	X2= X1AE3	X3= E1AE2	A= X2VX3
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	1	1
1	0	0	1	1	0	1
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	1	1	1	1	1

Zusammenfassung

Unter der Analyse eines Schaltnetzes versteht man die Untersuchung einer vorgegebenen Schaltung. Eine Untersuchung ist deshalb notwendig, um die logischen Eigenschaften eines Schaltnetzes feststellen zu können. Das Ergebnis der Analyse kann eine Schaltfunktion, die vollständige Funktionstabelle oder ein Programm sein.

Aufgabe B 15.1

Das Bild B 16.1 zeigt eine **Vergleicherschaltung**. Der Ausgang des Gebers I liefert die binären Signale E1 und E2, am Ausgang von Geber II entstehen die binären Signale E3 und E4. Die Ausgangssignale der Geber sind von den Eingangssignalen I und II abhängig. In der praktischen Anwendung könnten die Signale I und II prozeßabhängige Zustände melden. Die Schaltung vergleicht die Signalkombination E1,E2 mit der Kombination E3,E4. Das Ausgangssignal kann ein Melde- oder Stellsignal sein.

Füllen Sie bitte nach dem gelernten Schema die Funktionstabelle B 15.1 aus. Benutzen Sie die in die Schaltung eingetragenen Hilfs-signale. Stellen Sie zunächst die Hilfsfunktionen für X1 bis X6 auf. Welche Bedingungen müssen bei den Eingangssignalpaaren E1,E2 und E3,E4 erfüllt sein, damit das Ausgangssignal $A = 1$ wird?

Tabelle B 15.1: Funktionstabelle zur Aufgabe B 15.1

E4	E3	E2	E1	X1	X2	X3	X4	X5	X6	A
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Die Lösung dieser Aufgabe finden Sie auf Seite F11.

Beispiel B 15.1

Für die Vergleicherschaltung in Bild B 16.1 ist die Schaltfunktion aufzustellen. Wir lösen diese Aufgabe noch einmal gemeinsam, da die Schaltung etwas aufwendig ist. Das Aufstellen der Schaltfunktion erfolgt wieder schrittweise.

Lösung:

1. Schritt: Tragen Sie in die Schaltung die Hilfssignale ein. Wir verwenden die bereits in der Schaltung von Bild B 16.1 enthaltenen X-Signale.

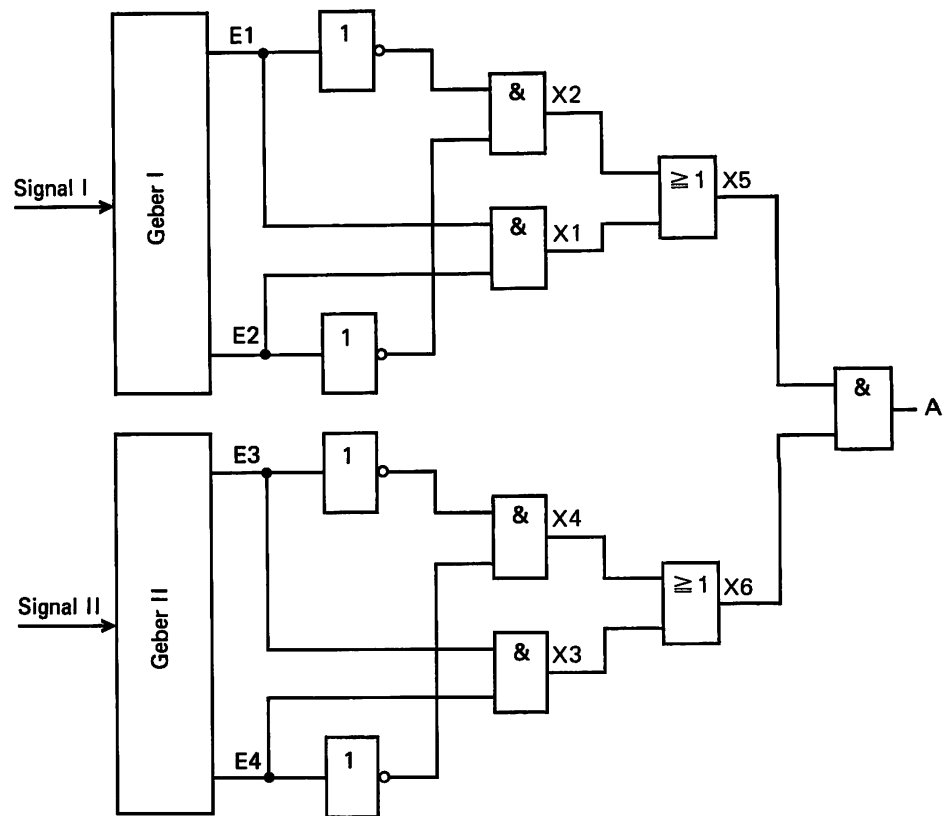
B**16**

Bild B 16.1
Vergleicherschaltung, Schaltung
zu Aufgabe B 15.1 und Beispiel
B 15.1.

2. Schritt: Aufstellen der Schaltfunktionen für die Hilfssignale X und für die Ausgangsvariable A. Aus der Schaltung lassen sich folgende Beziehungen ablesen:

$$X1 = E1 \wedge E2$$

$$X3 = E3 \wedge E4$$

$$X2 = \overline{E1} \wedge \overline{E2}$$

$$X4 = \overline{E3} \wedge \overline{E4}$$

$$X5 = X1 \vee X2$$

$$X6 = X3 \vee X4$$

$$A = X5 \wedge X6$$

3. Schritt: Die Hilfsfunktionen sind schrittweise zu ersetzen. In die Funktion für X5 sind für X1 und X2 die zugehörigen Schaltfunktionen einzusetzen.

$$X5 = \underbrace{(E1 \wedge E2)}_{X1} \vee \underbrace{(\overline{E1} \wedge \overline{E2})}_{X2}$$

In der Funktion für X6 sind X3 und X4 zu ersetzen.

$$X6 = \underbrace{(E3 \wedge E4)}_{X3} \vee \underbrace{(\overline{E3} \wedge \overline{E4})}_{X4}$$

Damit lautet die Schaltfunktion für die Ausgangsvariable:

$$A = \underbrace{[(E1 \wedge E2) \vee (\overline{E1} \wedge \overline{E2})]}_{X5} \wedge \underbrace{[(E3 \wedge E4) \vee (\overline{E3} \wedge \overline{E4})]}_{X6}$$

Denken Sie bitte an das Setzen von Klammern. Nur dann bleibt die Schaltfunktion übersichtlich. Die Schaltfunktion ist immer die Darstellung der Ausgangsvariablen in Abhängigkeit von den Eingangsvariablen. Die aufgestellte Schaltfunktion kann man unmittelbar programmieren.

Aufgabe B 17.1

Das Bild B 17.1 zeigt eine Schaltung zur Überwachung des Betriebszustands von zwei Kühlanlagen. Signal E1 meldet den Betriebszustand von Anlage 1, Signal E2 den von Anlage 2.

Es bedeuten:

Signalzustand E=1 „Keine Störung“

Signalzustand E=0 „Störung“

A1=1 Hupe EIN A2=1 Meldelampe EIN

A1=0 Hupe AUS A2=0 Meldelampe AUS

- Für die Ausgangsvariablen A1 und A2 sind die Schaltfunktionen aufzustellen.
- Füllen Sie bitte die vorbereitete Funktionstabelle B 17.1 aus.
- Welche Betriebszustände werden durch das Hup- bzw. Lichtsignal gemeldet?
- Ersetzen Sie die Schaltung in Bild B 17.1 durch einen Kontaktplan.

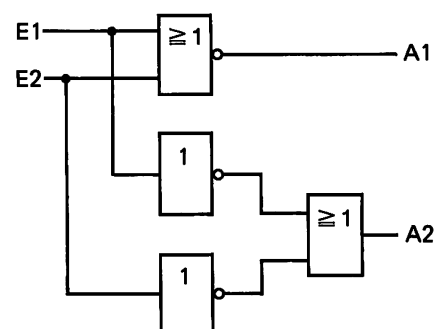


Bild B 17.1
Überwachungsschaltung, Schalt-
netz zur Aufgabe B 17.1.

Tabelle B 17.1: Funktionstabelle zur Aufgabe B 17.1

E2	E1	$\overline{E2}$	$\overline{E1}$	A1	A2
0	0				
0	1				
1	0				
1	1				

Die Lösung dieser Aufgabe finden Sie auf Seite F11.

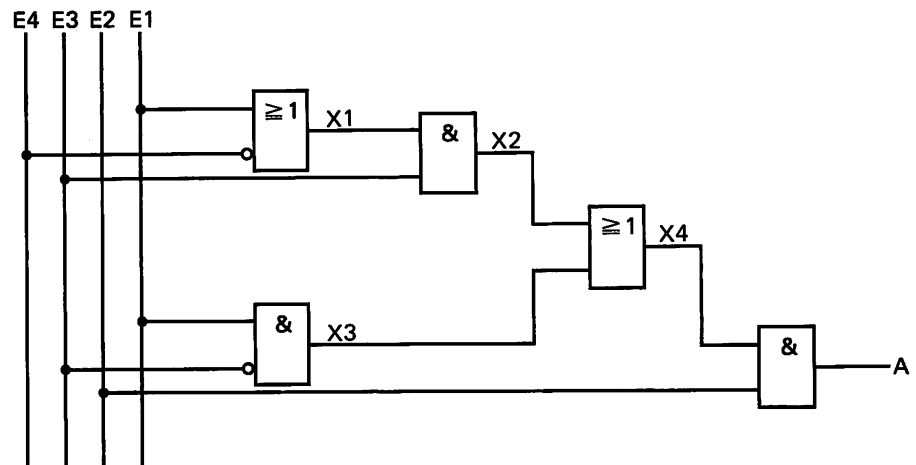
Aufgabe B 17.2

Das Bild B 18.1 enthält einen Schaltungsauszug. Zur leichteren Lösung der folgenden Aufgabe sind in die Schaltung bereits die Hilfs-signale X eingetragen.

- Stellen Sie zunächst für die Hilfsvariablen X1 bis X4 die Schaltfunktionen auf. Mit deren Hilfe ist die Schaltfunktion für die Ausgangsvariable A1 zu entwickeln.

B**18**

Bild B 18.1
Schaltung zur Aufgabe B 17.1.



- b) Bei welchen Kombinationen der Eingangsvariablen nimmt die Ausgangsvariable A den Wert 1 an? Sie können die Lösung mit Hilfe der Schaltfunktion oder durch Aufstellen der Funktionstabelle finden. Wählen Sie selbst einen Lösungsweg.

Die Lösung dieser Aufgabe finden Sie auf Seite F12.

Grundgesetze der Schaltalgebra

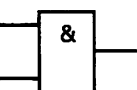
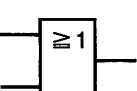

Die Schaltalgebra, auch als **Boolesche Algebra** bezeichnet, ist ein wichtiges Hilfsmittel der Digitaltechnik und damit auch der binären und digitalen Steuerungstechnik. Obwohl die Boolesche Algebra schon lange bekannt war, wurde sie erst in jüngster Zeit in der Digitaltechnik genutzt, um Schaltungen zu beschreiben und zu vereinfachen. Der englische Mathematiker Boole (1815 bis 1864) schuf hierfür die Voraussetzungen.

Eine ausführliche Abhandlung der Schaltalgebra ist nicht das Ziel des Lehrgangs. Wir wollen Ihnen nur einige Grundlagen vermitteln, die zum Entwurf und der Programmierung von Schaltungen nützlich sind. Der Beweis für verschiedene Gesetzmäßigkeiten wird deshalb auch nur experimentell geführt.

Postulate der Schaltalgebra

In der Schaltalgebra gelten bestimmte Grundgesetze. Dazu gehören die **Postulate**. Es sind Regeln für die **logische Verknüpfung von Konstanten**. Wir haben nur zwei konstante Werte; es sind die Zustände 1 und 0. Sie haben bereits mit diesen Gesetzmäßigkeiten bei der Besprechung der Grundverknüpfungen gearbeitet, indem für die Ein- oder Ausgangsvariablen deren Signalzustände in die Funktion eingesetzt wurden. Die Tabelle B 19.1 enthält eine Zusammenstellung der Postulate für die Grundverknüpfungen UND, ODER und NICHT.

Tabelle B 19.1: Zusammenstellung der Postulate

UND-Verknüpfung	$\begin{array}{cc cc} 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \end{array}$ 	$\begin{array}{l} 0 \wedge 0 = 0 \\ 0 \wedge 1 = 0 \\ 1 \wedge 0 = 0 \\ 1 \wedge 1 = 1 \end{array}$
ODER-Verknüpfung	$\begin{array}{cc cc} 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \end{array}$ 	$\begin{array}{l} 0 \vee 0 = 0 \\ 0 \vee 1 = 1 \\ 1 \vee 0 = 1 \\ 1 \vee 1 = 1 \end{array}$
NICHT-Funktion	$0 \mid 1$ 	$\begin{array}{l} \bar{0} = 1 \\ \bar{1} = 0 \end{array}$

Theoreme der Schaltalgebra

Theoreme sind **Rechenregeln**. Wir zeigen Ihnen zunächst die Verknüpfungsmöglichkeiten einer Variablen (E1) mit sich selbst und mit einer Konstanten (0 bzw. 1). Diese einfachen Beziehungen ermöglichen bereits Schaltungsvereinfachungen. Mit diesen theoretischen

Überlegungen verfolgen wir das Ziel, mit einem Minimum an Aufwand eine Schaltung technisch zu realisieren. Eine Zusammenstellung wichtiger Theoreme enthält die Tabelle B 20.1.

Tabelle B 20.1: Zusammenstellung wichtiger Theoreme

Theoreme der UND-Verknüpfung			
① $E1 \wedge E1$			$E1 \wedge E1 = E1$
② $E1 \wedge \bar{E1}$			$E1 \wedge \bar{E1} = 0$
③ $E1 \wedge 1$			$E1 \wedge 1 = E1$
④ $E1 \wedge 0$			$E1 \wedge 0 = 0$
Theoreme der ODER-Verknüpfung			
⑤ $E1 \vee E1$			$E1 \vee E1 = E1$
⑥ $E1 \vee \bar{E1}$			$E1 \vee \bar{E1} = 1$
⑦ $E1 \vee 1$			$E1 \vee 1 = 1$
⑧ $E1 \vee 0$			$E1 \vee 0 = E1$

Theoreme der UND-Verknüpfung

1. Theorem: An den beiden Eingängen liegt das gleiche Signal. Man kann demnach die beiden Eingänge parallelschalten. Der Signalzustand am Ausgang entspricht dem Signalzustand an den Eingängen.

2. Theorem: Am zweiten Eingang liegt das invertierte Signal gegenüber dem ersten Eingang. Einer der beiden Eingänge hat demnach immer Zustand 0. Die UND-Bedingung kann nicht erfüllt werden, der Ausgang hat immer Zustand 0.

3. Theorem: Da ein Eingang den Zustand 1 hat, bestimmt die Variable am anderen Eingang den Zustand der Ausgangsvariablen. Ein- und Ausgangsvariable haben den gleichen Wert.

4. Theorem: Das Signal an einem Eingang hat immer den Wert 0. Die UND-Bedingung kann nicht erfüllt werden, der Ausgang hat ebenfalls den Zustand 0.

Theoreme der ODER-Verknüpfung

5. Theorem: Beide Eingänge führen das gleiche Signal, sie können deshalb parallel geschaltet werden. Somit ist der Signalzustand an Eingängen und Ausgang gleich.

6. Theorem: Da am zweiten Eingang das invertierte Signal des anderen Eingangs liegt, führt ein Eingang immer Zustand 1. Damit ist die ODER-Bedingung erfüllt, der Ausgang hat immer Zustand 1.

7. Theorem: Ein Eingang hat dauernd Zustand 1. Damit ist die ODER-Bedingung erfüllt. Unabhängig vom Wert des anderen Eingangssignals hat der Ausgang immer den Wert 1.

8. Theorem: Da ein Eingang immer den Wert 0 hat, ist der Zustand des Ausgangs vom Wert der Eingangsvariablen abhängig. Ein- und Ausgangsvariable haben den gleichen Wert.

Zusammenfassung

Mit Hilfe der Schaltalgebra, auch als Boolesche Algebra bezeichnet, können Schaltungen der Digital- und Steuerungstechnik durch Schaltfunktionen beschrieben werden. Ähnlich wie in der Algebra gibt es Regeln, nach denen die Gleichungen umgestellt und vereinfacht werden können. Die **Postulate** beinhalten Regeln zur logischen Verknüpfung der binären Konstanten 1 und 0. Die **Theoreme** umfassen Rechenregeln für Verknüpfungen einer Variablen mit sich selbst und mit binären Konstanten.

Aufgaben B 21.1

- 1) In der Schaltung in Bild B 21.1 sind die Signalzustände A1 und A2 für $E1=1$ und $E1=0$ zu ermitteln. Stellen Sie eine Beziehung zwischen E1 und A2 her.
- 2) Tragen Sie bitte in die Schaltung von Bild B 21.2 für A1, A2 und A3 die Signalzustände ein. E1 kann die Werte 0 und 1 annehmen.
- 3) Für $E1=0$ und $E1=1$ sind die Signalzustände von A1 und A2 in der Schaltung von Bild B 21.3 zu bestimmen.

Die Lösungen dieser Aufgaben finden Sie auf Seite F 14.

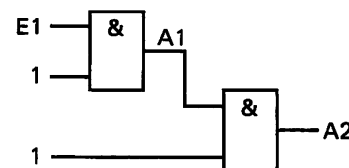


Bild B 21.1
Schaltung zur Aufgabe B 21.1,
Frage 1)

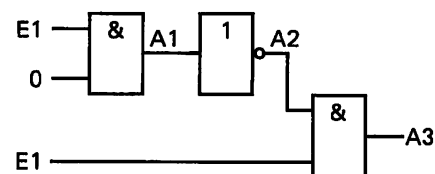


Bild B 21.2
Schaltung zur Aufgabe B 21.1,
Frage 2)

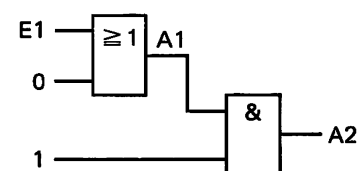


Bild B 21.3
Schaltung zur Aufgabe B 21.1,
Frage 3)

Synthese logischer Schaltnetze

Bisher wurden nur vorgegebene Schaltnetze analysiert, d.h. ihr Schaltverhalten wurde untersucht. Wir haben aber noch kein Schaltnetz entworfen und den Entwurf programmiert. Diese Tätigkeiten bezeichnet man als **Schaltnetzsynthese**.

Es ist Ziel der Schaltnetzsynthese, ein Schaltnetz zu erstellen, das die in der Aufgabe gestellten Forderungen erfüllt.

Die Synthese setzt deshalb eine exakt formulierte Aufgabenstellung voraus. Dazu können folgende Beschreibungsmethoden angewendet werden:

- Verbale Beschreibung
- Funktionstabelle
- Schaltfunktion

Betrachten wir die verschiedenen Beschreibungsmethoden etwas näher: Bei der **verbalen Beschreibung** der Aufgabenstellung muß Schritt für Schritt mit Worten dargelegt werden, wie die Variablen von der Schaltung zu verarbeiten sind. Für kleine Aufgaben ist die Beschreibung noch überschaubar, bei umfangreichen Aufgaben geht der Überblick bald verloren. Die verbale Beschreibung kann dann nur noch ergänzend mit einer anderen Beschreibungsart angewendet werden.

Große Bedeutung hat auch hier die **Funktionstabelle**. In ihr ist eindeutig die Abhängigkeit der Ausgangsvariablen von den Eingangsvariablen festgelegt. Damit sind die Bedingungen, bei denen die Ausgangsvariablen den Zustand 1 annehmen müssen, bekannt. An Hand der Funktionstabelle läßt sich auch die **Schaltfunktion** aufstellen. Vor deren Programmierung erfolgt noch eine Untersuchung, ob sich die Schaltfunktion nach den Regeln der Schaltalgebra vereinfachen läßt.

Einen stark vereinfachten Ablaufplan für die Synthese logischer Schaltnetze zeigt das Bild B 22.1. Es zeigt die verschiedenen Wege, die zum Entwurf eines Programms führen. Der Entwurf soll zunächst mit logischen Grundfunktionen erfolgen. In einem späteren Abschnitt lernen Sie dann deren Umformung in die NOR- bzw. NAND-Technik kennen.

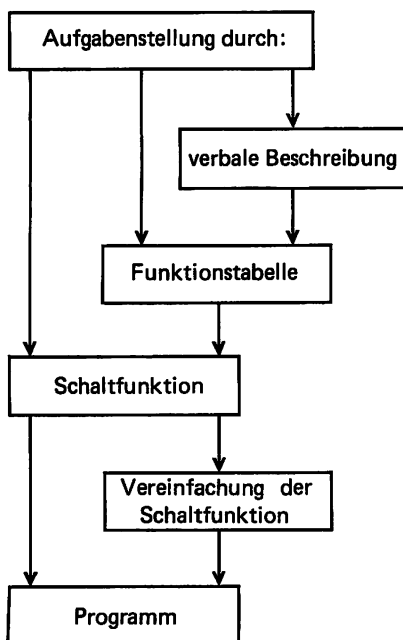


Bild B 22.1
Ablaufplan zur Synthese eines logischen Schaltnetzes.

Entwurf eines Programms nach verbaler Aufgabenstellung

Beispiel B 22.1

Die verbale Aufgabenstellung lautet:

Drei Signalgeber überwachen eine Anlage. Eine Meldeleuchte soll signalisieren, wenn von diesen drei Gebern zwei Geber ansprechen.

Sie soll aber kein Signal geben, wenn alle drei Geber aktiviert sind. Die Bearbeitung der Aufgabe erfolgt schrittweise. Sie ist eine Musterlösung für spätere umfangreichere Aufgaben, deren Lösung wesentlich straffer erfolgen muß.

Lösung:

1. Schritt: Festlegung der Eingangsvariablen und deren Signalzustände. Die Signalgeber und ihre Signale werden mit E1, E2 und E3 bezeichnet.

Es bedeuten:

Gebersignal vorhanden = Zustand 1

Gebersignal nicht vorhanden = Zustand 0

2. Schritt: Festlegung der Ausgangsvariablen und deren Signalzustände. Die Meldeleuchte und ihr Signal wird mit A1 bezeichnet.

Es bedeuten:

Lampe leuchtet = Zustand 1

Lampe leuchtet nicht = Zustand 0

3. Schritt: Aufstellen der vollständigen Funktionstabelle.

Für alle möglichen Kombinationen der Gebersignale ist der Signalzustand der Meldeleuchte gemäß der Aufgabenstellung anzugeben. Es entsteht die Funktionstabelle B 23.1. Zur besseren Kennzeichnung sind die verschiedenen Kombinationen K mit einer Dezimalziffer bezeichnet. Die Ausgangsvariable A1 nimmt nur dann den Zustand 1 an, wenn genau zwei von den drei Eingangsvariablen den Wert 1 haben.

Tabelle B 23.1: Funktionstabelle zu Beispiel B 22.1

K	E3	E2	E1	A1
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1 X3
4	1	0	0	0
5	1	0	1	1 X5
6	1	1	0	1 X6
7	1	1	1	0

4. Schritt: Konjunktive Zusammenfassung der Signalkombinationen.

Die Zeilen, deren Signalkombinationen $A1=1$ ergeben, führen zum Ansprechen der Meldeleuchte. Diese Kombinationen sind in der Tabelle B 23.1 als Hilfsvariablen X gekennzeichnet. Die zur jeweiligen Hilfsvariablen gehörenden Zustände der Eingangsvariablen werden konjunktiv (mit UND) zusammengefaßt:

$$X3 = E1 \wedge E2 \wedge \overline{E3}$$

$$X5 = E1 \wedge \overline{E2} \wedge E3$$

$$X6 = \overline{E1} \wedge E2 \wedge E3$$

5. Schritt: Aufstellen der Schaltfunktion.

Laut Funktionstabelle muß die Meldeleuchte ansprechen, wenn die Hilfsvariablen X3 ODER X5 ODER X6 Zustand 1 führen. Wir erhalten:

$$A1 = X3 \vee X5 \vee X6$$

$$A1 = (E1 E2 \overline{E3}) \vee (E1 \overline{E2} E3) \vee (\overline{E1} E2 E3)$$

6. Schritt: Vereinfachung der Schaltfunktion.

Oft wird dieser Vorgang auch als **Minimierung** oder **Optimierung** bezeichnet. Für die Schaltfunktion unseres Beispiels gibt es keine sinnvolle Zusammenfassung der Variablen, die eine Vereinfachung der Schaltfunktion ergeben würden.

7. Schritt: Programmierung der Schaltfunktion in Form von

- a) Kontaktplan,
- b) Funktionsplan,
- c) Anweisungsliste und
- d) Anweisungen.

Im Beispiel sollen die Signalzustände der Hilfsfunktionen X3, X5 und X6 noch für andere Aufgaben ausgewertet werden. Wir setzen sie deshalb in die Merker M3, M5 und M6.

Den Kontaktplan sehen Sie in Bild B 25.1a). Die Ergebnisse der UND-Verknüpfungen sind in Merkern gespeichert. Die Merkerinhalte werden anschließend abgefragt und mit ODER verknüpft. Das gilt sinngemäß auch für den Funktionsplan in Bild B 25.1b).

In der Anweisungsliste stehen die einzelnen Anweisungen in der Reihenfolge, wie sie von der SPS zu bearbeiten sind. Wir verwenden hier wieder den genormten Mnemo-Code.

```

U   E 1
U   E 2
UN  E 3
=   M 3   Hilfsvariable X3
U   E 1
UN  E 2
U   E 3
=   M 5   Hilfsvariable X5
UN  E 1
U   E 2
U   E 3
=   M 6   Hilfsvariable X6
O   M 3
O   M 5
O   M 6
=   A 1   Meldeleuchte
=   BE

```

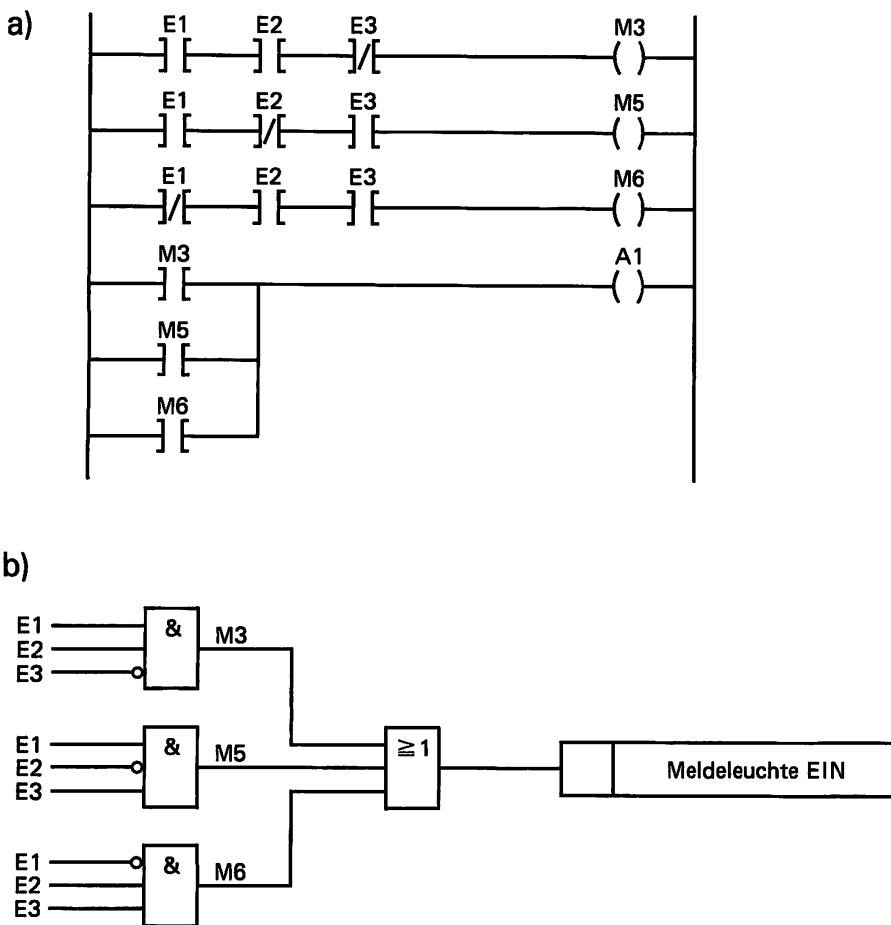


Bild B 25.1
 Pläne zum Beispiel B 22.1
 a) Kontaktplan,
 b) Funktionsplan.

Aus der Anweisungsliste werden nun die Anweisungen für das Programm zur Eingabe in den NDR-Computer gebildet. Hierzu verwenden wir wieder die mathematische Schreibweise.

$\neg E1 \& E2 \& \neg E3 = M3$
 $\neg E1 \& \neg E2 \& E3 = M5$
 $\neg E1 \& E2 \& E3 = M6$
 $M3 / M5 / M6 = A1$
 $\neg PE$

Geben Sie dieses Programm nun in den NDR-Computer ein. Lassen Sie sich den Kontaktplan anzeigen und vergleichen Sie ihn mit dem Bild B 25.1a. Testen Sie das Programm anhand der Funktionstabelle B 23.1. Kontrollieren Sie anschließend, ob die Lampe des Ausgangs A1 immer dann aufleuchtet, wenn von den drei Eingängen E1, E2, E3 zwei eingeschaltet sind.

Wir haben Ihnen gezeigt, wie man ein Problem in kleinen Schritten auflöst, um als Ergebnis ein Programm zu erhalten. Die folgenden Beispiele werden wir nicht mehr so ausführlich wie das Einführungsbeispiel behandeln. Lesen Sie im Zweifelsfall das eben beschriebene Beispiel nochmals durch.

Zusammenfassung

Schaltnetzsynthese bedeutet, daß für eine gestellte Aufgabe das Programm des erforderlichen Schaltnetzes zu entwickeln ist. Das Programm muß alle in der Aufgabenstellung geforderten Bedingungen erfüllen. Voraussetzung dafür ist eine exakte Formulierung des Problems. Die Aufgabenstellung erfolgt oft durch eine verbale Beschreibung oder die Funktionstabelle bzw. die Schaltfunktion. Vor der Programmierung ist die Schaltfunktion zu minimieren.

Anwendung der Schaltalgebra zur Minimierung von Schaltfunktionen

Einige Grundgesetze der Schaltalgebra kennen Sie bereits — gemeint sind die **Postulate und Theoreme**, mit deren Hilfe sich vorgegebene oder aufgestellte Schaltfunktionen vereinfachen lassen. Unter Vereinfachung verstehen wir die Beseitigung von überflüssigen Termen und Variablen. Eine Schaltfunktion mit einfacherer Struktur läßt sich leichter programmieren, es sind weniger Anweisungen erforderlich, das Programm hat eine geringere Zykluszeit.

Auflösung von Klammern

Oftmals sind innerhalb der Schaltfunktion Verknüpfungen durch Klammern zusammengefaßt. Man kann erst nach Beseitigung der Klammern feststellen, ob die Möglichkeit der Vereinfachung besteht. An einigen typischen Beispielen wollen wir Ihnen das Auflösen von Klammern zeigen.

Vorher geben wir Ihnen noch einen Hinweis zur vereinfachten Schreibweise von Schaltfunktionen. Bisher haben wir sie wie folgt geschrieben:

$$A = (E1 \wedge \overline{E2} \wedge \overline{E3}) \vee (E1 \wedge \overline{E2} \wedge E3) \vee (\overline{E1} \wedge \overline{E2} \wedge E3)$$

Falls keine falsche Auslegung möglich ist, darf das Verknüpfungszeichen für UND (\wedge) auch weggelassen werden:

$$A = (E1 \overline{E2} \overline{E3}) \vee (E1 \overline{E2} E3) \vee (\overline{E1} \overline{E2} E3)$$

Diese Schreibweise ist schon wesentlich einfacher. Auch die Klammern werden noch überflüssig, wenn man zwischen den UND-Verknüpfungen und dem Operationszeichen \vee einen Abstand läßt und die Regel „UND vor ODER“ beachtet. In vereinfachter Schreibweise erhält man schließlich:

$$A = E1 \overline{E2} \overline{E3} \vee E1 \overline{E2} E3 \vee \overline{E1} \overline{E2} E3$$

Die kommenden Schaltfunktionen wollen wir Ihnen in dieser Form darstellen.

Beispiel B27.1

In nachstehender Schaltfunktion ist die Klammer aufzulösen.

$$A1 = E1(E2E3 \vee \overline{E2}\overline{E3} \vee E4)$$

Lösung:

Jeder Term innerhalb der Klammer ist konjunktiv (mit UND) mit der Variablen E1 vor der Klammer zu verknüpfen. Für die Auflösung der Klammer gilt folgendes Schema:

$$A1 = E1(E2E3 \vee \overline{E2}\overline{E3} \vee E4)$$

$$A1 = E1E2E3 \vee E1\overline{E2}\overline{E3} \vee E1E4$$

Es kann auch möglich sein, daß eine Schaltfunktion mehrere durch Klammern getrennte, ineinander verschachtelte Ausdrücke enthält. Die Klammern werden nach dem Schema des letzten Beispiels aufgelöst.

Beispiel B27.2

In der folgenden Schaltfunktion sind die Klammern aufzulösen:

$$A2 = E1[E2(\overline{E3}E4 \vee E3\overline{E4})]$$

Lösung:

Wir beginnen mit der Auflösung der inneren Klammer:

$$A2 = E1[E2(\overline{E3}E4 \vee E3\overline{E4})]$$

$$A2 = E1[E2\overline{E3}E4 \vee E2E3\overline{E4}]$$

Nun werden die äußeren Klammern aufgelöst:

$$A2 = E1[E2\overline{E3}E4 \vee E2E3\overline{E4}]$$

$$A2 = E1E2\overline{E3}E4 \vee E1E2E3\overline{E4}$$

Zwecks besserer Übersicht sollte man die Variablen innerhalb der Terme immer in numerischer Folge anordnen, also in der Reihenfolge E1E2E3E4, nicht etwa in der Folge E3E2E4E1.

Das kommende Beispiel enthält die konjunktive Verknüpfung von zwei Klammerausdrücken. Die Klammern sind wieder zu entfernen.

Beispiel B28.1

In der Schaltfunktion $A3 = (E1E3 \vee \overline{E3})(E2E4 \vee \overline{E4})$ sind die Klammerausdrücke aufzulösen.

Lösung:

Grundsätzlich hat auch hier das bisher angewendete Schema Gültigkeit. Alle Ausdrücke in den Klammern sind miteinander konjunktiv zu verknüpfen. Wir erhalten:

$$A3 = (E1E3 \vee \overline{E3})(E2E4 \vee \overline{E4})$$

$$A3 = E1E2E3E4 \vee E1E3\overline{E4} \vee E2\overline{E3}E4 \vee \overline{E3}\overline{E4}$$

Vereinfachung von Schaltfunktionen

Durch die Anwendung der Postulate, Theoreme und Regeln zum Auflösen von Klammern lassen sich oft kompliziert aussehende Schaltfunktionen vereinfachen. An einigen Beispielen werden wir Ihnen verschiedene Lösungswege für Vereinfachungen zeigen.

Anfangs wird bei Ihnen sicher die Frage auftauchen, ob die Regeln zur Schaltungsvereinfachung auch richtig angewendet wurden. Diese Frage lässt sich mit einem Test leicht beantworten. Der Test erfolgt durch Aufnahme der Funktionstabelle. Vereinfachte und nicht vereinfachte Schaltfunktion werden programmiert und nacheinander in die SPS eingegeben. Beide Programme müssen die gleichen Verknüpfungsergebnisse liefern.

Beispiel B28.2

Die Abhängigkeit der Ausgangsvariablen A1 von den Eingangsvariablen E1, E2 und E3 ist durch folgende Schaltfunktion gegeben:

$$A1 = \overline{E2}(\overline{E1}(E3 \vee E1E2) \vee \overline{E1}\overline{E2}\overline{E3})$$

- Die Klammern sind nach den besprochenen Regeln aufzulösen.
- Es ist zu untersuchen, auf welche Weise sich die Schaltfunktion vereinfachen lässt.
- Stellen Sie das Programm für die gegebene Schaltfunktion auf.
- Die unter b) vereinfachte Schaltfunktion ist ebenfalls zu programmieren.
- Nacheinander sind die Programme von c) und d) in die SPS einzugeben und zu testen. Vergleichen Sie die Verknüpfungsergebnisse der in Tabelle B29.1 aufgelisteten Signalkombinationen.

Lösung:

- a) Zuerst wird die innere Klammer, dann die äußere Klammer aufgelöst. Wir erhalten nach dem **Ausmultiplizieren**:

$$A1 = \underbrace{\overline{E1} \overline{E2} E3}_I \vee \underbrace{E1 \overline{E1} E2 \overline{E2}}_{II} \vee \underbrace{\overline{E1} E2 \overline{E2} E3}_{III}$$

- b) **Term I:** Er lässt sich nicht vereinfachen, da jede Variable nur einmal in negierter oder nichtnegierter Form vorhanden ist.

Term II: Nach Tabelle B 20.1, Regel 2 ist

$$E1 \overline{E1} = 0 \quad \text{und} \quad E2 \overline{E2} = 0.$$

Term II kann gestrichen werden, da die Verknüpfung $E1 \overline{E1} E2 \overline{E2}$ immer den Wert 0 ergibt.

Term III: Bei sinngemäßer Anwendung von Regel 1 in Tabelle B 20.1 ist $\overline{E2} E2 = \overline{E2}$. Eine der beiden Variablen $\overline{E2}$ ist redundant (überflüssig) und kann entfallen. Es gilt:

$$\overline{E1} E2 \overline{E2} E3 = \overline{E1} E2 E3$$

Nunmehr lautet die Schaltfunktion:

$$A1 = \overline{E1} \overline{E2} E3 \vee \overline{E1} E2 E3$$

- c) Das Programm der Schaltfunktion laut Aufgabenstellung:
Im Gegensatz zur Schaltfunktion dürfen Sie in den Anweisungen das Operationszeichen „&“ nicht weglassen!

$$!NE2 \& (NE1 \& (E3 / E1 \& E2) / NE1 \& NE2 \& NE3) = A1$$

!PE

- d) Das Programm der vereinfachten Schaltfunktion:

$$!NE1 \& NE2 \& E3 / NE1 \& NE2 \& NE3 = A1$$

!PE

- e) Tabelle B 29.1: Funktionstabelle für Beispiel B 28.2

E3	E2	E1	A1	A1,c	A1,d
0	0	0	1		
0	0	1	0		
0	1	0	0		
0	1	1	0		
1	0	0	1		
1	0	1	0		
1	1	0	0		
1	1	1	0		

Spalte A1 enthält die theoretisch ermittelten Werte. Die A1-Werte nach Punkt c) und d) müssen mit diesen übereinstimmen. Damit ist die Richtigkeit der Vereinfachung nachgewiesen. Die Umkehrung des Auflöserns von Klammern durch das Ausmultiplizieren ist das **Setzen von Klammern**. Hierbei werden Variablen oder Terme von Schalt-

funktionen ausgeklammert. Die allen Termen gemeinsame Variable wird als **Faktor** vor die Klammer gesetzt. Auch dadurch können sich vereinfachte Schaltfunktionen ergeben. Einige typische Schulbeispiele sind nachstehend aufgeführt.

B

30

Vereinfachung von Schaltfunktionen durch Ausklammern

Beispiel B 30.1

Gegeben ist die Schaltfunktion: $A1 = E0E1\overline{E2} \vee E0E3 \vee E0E4$

Die allen Termen gemeinsame Variable ist $E0$. Sie soll ausgeklammert werden.

Lösung:

Die Variable $E0$ ist konjunktiv mit den Variablen der drei Terme verknüpft. Diese grundsätzliche Beziehung muß bestehen bleiben. Variable $E0$ wird deshalb vor die Klammer gesetzt und konjunktiv mit dem Klammerausdruck verknüpft.

$$A1 = E0(E1\overline{E2} \vee E3 \vee E4)$$

Die Anweisung des Programms lautet:

```
!E0&(E1&NE2/E3/E4)=A1
!PE
```

Beispiel B 30.2

In der Schaltfunktion $A2 = E1 \vee E1E2 \vee E1\overline{E2}E3$ ist die Variable $E1$ auszuklammern.

Lösung:

Wenn wir $E1$ vor die Klammer setzen, entsteht in der Klammer eine Lücke, die ausgefüllt werden muß.

$$A2 = E1(? \vee E2 \vee \overline{E2}E3)$$

Nach Tabelle B 20.1, Regel 3 kann man schreiben: $E1 = E1 \wedge 1$.

Dieser Ausdruck wird in die Schaltfunktion eingesetzt. Nach dem Ausklammern der Variablen $E1$ ist die „Lücke“ mit der Konstanten 1 besetzt.

$$A2 = E1(1 \vee E2 \vee \overline{E2}E3)$$

Nach Regel 7 in Tabelle B 20.1 hat der Klammerausdruck den Wert 1. Hat nämlich in einer ODER-Verknüpfung **eine** Eingangsvariable den Wert 1, dann ist die Ausgangsvariable immer 1, gleichgültig, welche Zustände die anderen Eingangsvariablen aufweisen. Als Lösung erhalten wir also mit Regel 3 in Tabelle B 20.1:

$$A2 = E1$$

Beispiel B 31.1

Die Schaltung in Bild B 31.1 ist zu untersuchen.

- Stellen Sie durch eine Schaltungsanalyse die Schaltfunktion fest.
- In der Schaltfunktion a) sind die Klammern aufzulösen. Vereinfachen Sie die Schaltfunktion.
- Programmieren Sie die Schaltfunktion a). Für die Zwischenergebnisse (das sind die Klammerausdrücke) sind Merker zu verwenden. Der Kontaktplan ist aufzuzeichnen — stellen Sie für das Programm die Anweisungen auf.
- Programmieren Sie direkt die Schaltfunktion von b). Der Kontaktplan ist aufzuzeichnen, das Programm ist in Form von Anweisungen aufzustellen.
- Speichern und Testen Sie nacheinander die Programme von c) und d). Nehmen Sie durch die Betätigung der Eingabeschalter die Funktionstabelle B 33.1 auf. A1 beinhaltet die theoretisch ermittelten Werte der Ausgangsvariablen. A1c und A1d sind die durch den Test ermittelten Verknüpfungsergebnisse der Programme c) und d).

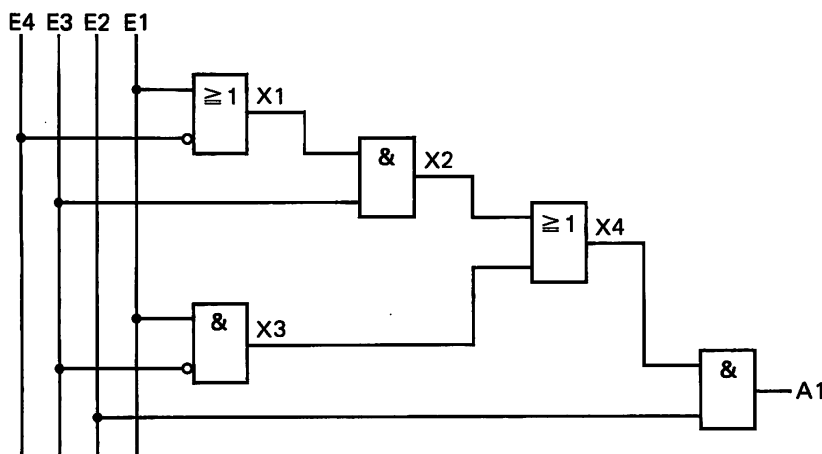


Bild B 31.1
Schaltung zum Beispiel B 31.1.

Lösung:

- Bei der Analyse gehen wir nach dem gelernten Schema vor. Erst sind die Schaltfunktionen für die Hilfsvariablen aufzustellen, anschließend wird die Schaltfunktion für die Ausgangsvariable gebildet. Wir benutzen die in Bild B 31.1 eingetragenen Hilfsvariablen X. Verfolgen Sie die nachstehenden Schaltfunktionen bitte anhand der Schaltung.

$$X1 = E1 \vee \overline{E4}$$

$$X2 = E3 X1$$

Für die Hilfsvariable X1 wird die vorstehende Funktion eingesetzt:

$$X2 = E3(E1\overline{V}E4)$$

$$X3 = E1\overline{E3}$$

$$X4 = X2 \vee X3$$

Die Hilfsvariablen $X2$ und $X3$ werden durch die abgeleiteten Funktionen ersetzt:

$$X4 = E3(E1\overline{V}E4) \vee E1\overline{E3}$$

Schaltfunktion der Ausgangsvariablen:

$$A1 = E2X4$$

$$A1 = E2(\underbrace{E3(E1\overline{V}E4)}_{M1} \vee \underbrace{E1\overline{E3}}_{M2})$$

- b) Auflösung der Klammern. Nach dem gelernten Schema wird erst die innere Klammer durch Ausmultiplizieren aufgelöst:

$$A1 = E2(E1E3 \vee E3\overline{V}E4 \vee E1\overline{E3})$$

Die Auflösung der äußeren Klammer ergibt:

$$A1 = \underbrace{E1E2E3}_{I} \vee \underbrace{E2E3\overline{V}E4}_{II} \vee \underbrace{E1E2\overline{E3}}_{III}$$

Nun betrachten wir die Terme und untersuchen, ob sich die Schaltfunktion vereinfachen läßt.

Die Terme I und III enthalten die gleiche Verknüpfung $E1E2$. Diese wird ausgeklammert. Also ist:

$$E1E2E3 \vee E1E2\overline{E3} = E1E2(E3 \vee \overline{E3}) = E1E2$$

Nach der Regel 6, Tabelle B 20.1 ist $(E3 \vee \overline{E3}) = 1$, der Klammerausdruck hat also nach Regel 3, Tabelle B 20.1 keinen Einfluß auf die Signalverknüpfung.

Damit lautet die Schaltfunktion für die Ausgangsvariable:

$$A1 = E1E2 \vee E2E3\overline{V}E4$$

Nach dem Ausklammern der gemeinsamen Variablen $E2$ entsteht folgende Lösung:

$$A1 = E2(E1 \vee E3\overline{V}E4)$$

Diese Schaltfunktion erfüllt die gleiche Aufgabe wie die unter a) aufgestellte Funktion.

- c) Den Kontaktplan für die unter a) aufgestellte Schaltfunktion zeigt das Bild B 33.1. Die Ergebnisse der Klammerausdrücke sind in Merker gesetzt. Nach dem Kontaktplan lauten die Anweisungen des Programms:

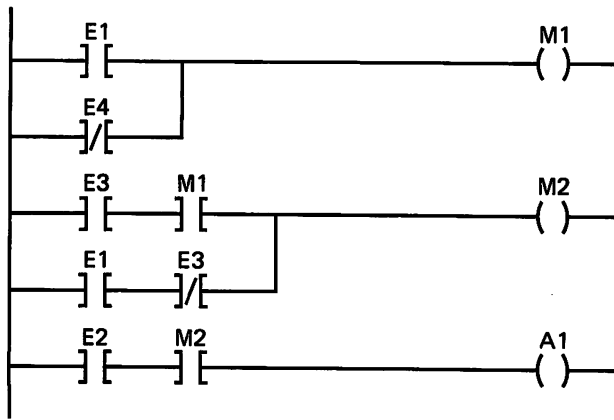


Bild B 33.1
Kontaktplan zum Beispiel B 31.1,
Frage c.

!E1/NE4=M1
!E3&M1/E1&NE3=M2
!E2&M2=A1
!PE

d) In Bild B 33.2 ist der Kontaktplan der vereinfachten Schaltfunktion wiedergegeben.

Das Programm lautet:

!E2&(E1/E3&NE4)=A1
!PE

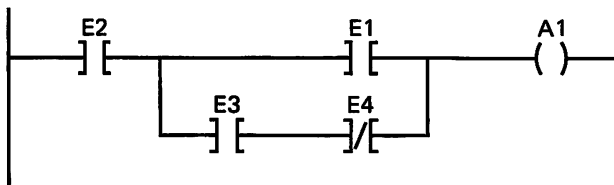


Bild B 33.2
Kontaktplan der vereinfachten
Schaltung im Beispiel B 31.1,
Frage d.

Tabelle B 33.1: Funktionstabelle zu Beispiel B 31.1

E4	E3	E2	E1	A1	A1c	A1d
0	0	0	0	0		
0	0	0	1	0		
0	0	1	0	0		
0	0	1	1	1		
0	1	0	0	0		
0	1	0	1	0		
0	1	1	0	1		
0	1	1	1	1		
1	0	0	0	0		
1	0	0	1	0		
1	0	1	0	0		
1	0	1	1	1		
1	1	0	0	0		
1	1	0	1	0		
1	1	1	0	0		
1	1	1	1	1		

- e) In der gezeigten Funktionstabelle B 33.1 enthält Spalte A1 die theoretisch ermittelten Werte. Durch Betätigung der Eingabeschalter E1 bis E4 sind experimentell die Signalwerte für A1c und A1d zu ermitteln und in die Tabelle einzutragen. Die experimentell und theoretisch ermittelten Signalzustände müssen für gleiche Kombinationen der Eingangsvariablen übereinstimmen.

B**34**

Speicherglieder

In den bisher besprochenen Schaltnetzen ist ein bestimmter Zustand der Ausgangsvariablen nur solange vorhanden, wie die zugehörige **Bitkombination** der Eingangsvariablen an den Eingängen der Schaltung anliegt. Ändern sich die Signalzustände am Eingang, so ändert sich auch der Zustand des Ausgangssignals.

In der Signalverarbeitung, zu der ja auch die **Steuerungstechnik** zählt, ist es sehr oft erforderlich, Signalzustände über die Dauer ihres Auftretens hinaus festzuhalten und wirksam werden zu lassen. In diesem Fall sind bestimmte Signalzustände, die nicht dauernd zur Verfügung stehen, zu speichern.

Folgende Beispiele machen dies deutlich.

Denken Sie an die Betätigung eines Aufzugs. Das Startsignal, das auch das Ziel enthält, wird gespeichert. Nach Erreichen des Ziels (z.B. 3. Etage) muß das gespeicherte Signal wieder gelöscht werden.

Bei Inbetriebnahme einer elektrischen Anlage betätigen Sie nur kurzzeitig den Taster „EIN“. Ein Bauelement innerhalb der Steuerung **merkt** sich den gewählten Betriebszustand. Erst mit Betätigung des Tasters „AUS“ wird die Anlage stillgesetzt.

In den vorstehenden Beispielen wird über ein Eingabesignal der Speicher gesetzt und über ein zweites Eingabesignal wieder gelöscht. Die Anlage kann sich somit einen Signalzustand über eine beliebig lange Zeit merken; es ist ein „**Langzeitspeicher**“. Es gibt auch „**Kurzzeitspeicher**“. Diese merken sich einen Schaltzustand nur über eine begrenzte, einstellbare Zeit. Denken Sie dabei an die Zeitschalter für die Hausflurbeleuchtung. Diese Speicher benötigen Zeitglieder, die Sie erst in einem späteren Abschnitt kennenlernen.

Eine elektronische Schaltung, die ein Binärzeichen (also 1 Bit) speichern kann, ist ein **Speicherglied**. Das Speicherglied hat zwei stabile, sogenannte **statische Zustände**. Der Signalausgang führt den Wert 1 oder 0. Nur bestimmte Steuersignale kippen die Schaltung aus dem einen stabilen in den anderen stabilen Zustand. Nach Wegfall der Steuersignale bleibt der zuletzt eingenommene Zustand der Ausgangsvariablen bestehen. Die Speicherkapazität eines Speicherglieds beträgt somit **1 Bit**.

Die Grundform des Schaltzeichens zeigt das Bild B 35.1. Zwischen Schaltzeichen und Funktionssymbol besteht beim Speicherglied kein Unterschied. Nach DIN 40 700, Teil 14 gelten folgende allgemeine Festlegungen:

- Das Symbol ist ein Rechteck, das durch eine gestrichelte Linie in zwei Felder aufgeteilt ist. Der Informationsfluß verläuft im Symbol längs der gestrichelten Linie. Wenn die Variable eines Eingangs den Wert 1 hat, nimmt die Variable an dem Ausgang, der am gleichen Feld des Schaltzeichens liegt, ebenfalls den Wert 1 an. (Z.B. $E1=1$ bewirkt $A=1$.)

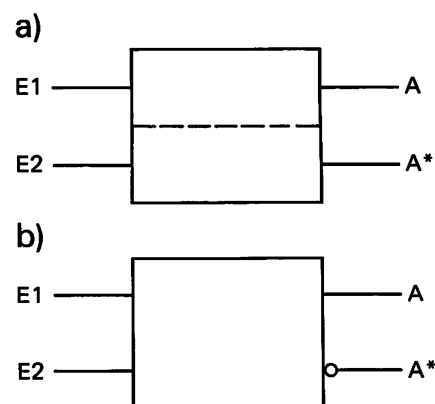


Bild B 35.1
Grundformen der Schaltzeichen
von Speichergliedern.

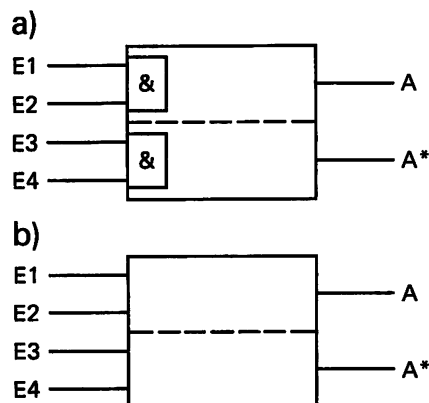
B**36**

Bild B 36.1
Speicherglieder mit mehreren
Steuereingängen.
Steuernd wirken in
Schaltung a) (E1AE2) bzw.
(E3AE4),
Schaltung b) (E1VE2) bzw.
(E3VE4).

- Die Variablen von zwei Ausgängen, die durch die gestrichelte Linie getrennt sind, haben komplementäre (gegensätzliche) Signalwerte (z. B. $A=1$ ergibt $A^*=0$; $A=0$ ergibt $A^*=1$). Es ist also $A^*=\bar{A}$.
- Zwecks einfacherer Darstellung kann die gestrichelte Linie entfallen. In diesem Fall ist der Komplementärausgang mit einem Negationszeichen zu versehen (Bild B 35.1b).
- Hat ein Feld mehrere gleichartige Eingänge, so ist mit einem Schaltzeichen (UND, Exklusiv-ODER) anzugeben, wie die Eingänge miteinander verknüpft sind. Das Verknüpfungsergebnis wirkt auf das Speicherglied ein. Falls kein Schaltzeichen eingetragen ist, besteht eine ODER-Verknüpfung. Das Bild B 36.1 zeigt zwei Beispiele.

Ursprung der heutigen elektronischen Speicherglieder sind die **Kippschaltungen**, die z. B. in verbindungsprogrammierten Steueranlagen (VPS) vielfach eingesetzt werden. Bezüglich des Kippverhaltens wird zwischen **astabilen**, **monostabilen** und **bistabilen** Schaltungen unterschieden. Auf diese drei Schaltungsarten wollen wir nun kurz eingehen.

Anmerkung: Die Ausgänge sind mit den Buchstaben A und A* bezeichnet. Im Setzzustand (hierbei hat das Speicherglied eine Information gespeichert) haben die Ausgänge die Zustände $A=1$ und $A^*=0$. Im Rücksetzzustand haben sie die Werte $A=0$ und $A^*=1$.

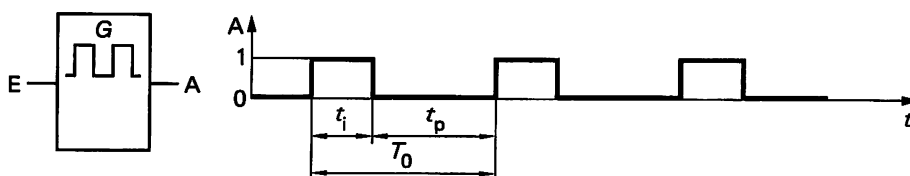
Kippschaltungen

Astabile Kippschaltung

Eine Schaltung ist stabil, wenn sich nach dem Einschalten der Spannung ein bestimmter konstanter Strom einstellt. Eine Schaltung ist astabil, wenn sich nach Inbetriebnahme die Ausgangsspannung mit einer bestimmten Frequenz sprunghaft zwischen zwei Spannungswerten ändert. Bei den Grundsaltungen entsprechen die Spannungswerte den logischen Zuständen 1 und 0. Dieses Verhalten wird als „Kippen“ bezeichnet. Die Frequenz der Spannungsänderung nennt man **Kippfrequenz**.

Die Steuerung des Kippens erfolgt über das Auf- oder Entladen eines Kondensators. Mit einem vorgeschalteten Widerstand ist die Kippfrequenz einstellbar. Das Schaltzeichen und das Signal-Zeit-Diagramm des astabilen Kippglieds gibt das Bild B 36.2 wieder.

Bild B 36.2
Astabiles Kippglied,
Schaltzeichen und Signal-Zeit-
Diagramm.
 t_i = Impulsdauer, t_p = Pausenzeit.
 $T_0 = t_i + t_p$ = Dauer einer
Schaltperiode,
 $f_s = 1/T_0$ = Schalt- bzw. Kipp-
frequenz.



Anwendung: Eine astabile Kippschaltung kann als Taktgenerator eingesetzt werden. Die rechteckförmigen Taktsignale **synchronisieren** das Zusammenarbeiten digital arbeitender Baugruppen und Geräte. Auch unser NDR-Computer benötigt einen Taktgenerator. Selbstverständlich können Sie damit auch einen Geber für ein Blinklicht aufbauen oder Produktionsabläufe steuern.

Monostabile Kippschaltung

Wie Sie dem Namen schon entnehmen können, hat die Schaltung nur einen stabilen Signalzustand. Schaltzeichen und Signal-Zeit-Diagramm zeigen das Bild B 37.1. Für das Schaltverhalten können Sie dem Diagramm folgende Wirkung entnehmen: Mit einem 1-Signal am Eingang E kippt die Schaltung von der stabilen ($A=0$) in die instabile Lage ($A=1$). Am Ausgang wechselt das Signal vom Zustand 0 in den Zustand 1. In diesem verharrt die Schaltung während der **Verweilzeit** t_v . Danach kippt sie ohne Rücksetzsignal selbständig wieder in den stabilen Ruhezustand zurück. In der Regel ist das Schaltverhalten (die Verweilzeit t_v) unabhängig von der Dauer des Eingangssignals.

Während der Verweilzeit hat die monostabile Kippschaltung, auch als Mono-FF (Abkürzung für Mono-Flip-Flop) bezeichnet, die Information 1 Bit gespeichert. Die Länge der Verweilzeit ist mit einer Widerstand-Kondensator-Schaltung (als RC-Glied bezeichnet) einstellbar.

Anwendung: Das monostabile Kippglied eignet sich

- zur Regeneration stark verformter Rechtecksignale (Impulsformer),
- zur Verkürzung oder Verlängerung von Rechtecksignalen und
- als zeitbegrenzender Signalspeicher (Zeitglied).

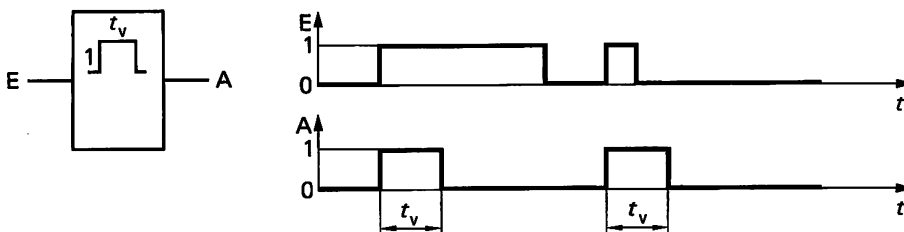


Bild B 37.1
Monostabiles Kippglied, Schaltzeichen und Signal-Zeit-Diagramm.

Bistabile Kippschaltung

Sie ist die Grundschiung des elektronischen Speicherglieds mit zwei stabilen Zuständen, die dem 0- bzw. 1-Zustand entsprechen. Die Schaltung bezeichnen wir als **Speicherglied**. Die verschiedenen Arten von Speichergliedern wollen wir im folgenden Abschnitt besprechen.

Statisches RS-Speicherglied

Die Schaltzeichen für einige Schaltungsvarianten zeigt uns das Bild B 38.1. Den Kippvorgang lösen die Eingangsvariablen aus.

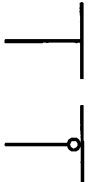
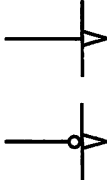
Die Bezeichnungen an den Eingängen bedeuten:

S Setzeingang (S=set)

R Rücksetzeingang (R=reset)

Bei **statischen Eingängen** (deshalb auch die Bezeichnung statisches Speicherglied) wird der Kippvorgang von den Zuständen 1 oder 0 der anliegenden Binärsignale ausgelöst. Außer den statischen Eingängen gibt es sogenannte **dynamische Eingänge**. Diese reagieren auf einen Zustandswechsel von 0 nach 1 oder von 1 nach 0. Die in der Norm festgelegten Eingangsdarstellungen enthält die Tabelle B 38.1.

Tabelle B 38.1: Eingänge für Speicherglieder

Bezeichnung	auslösendes Signal	Symbol
Eingang für Zustandssteuerung	1 0	 statischer Eingang
Eingang für Flanken-Steuerung	0 → 1 1 → 0	 dynamischer Eingang

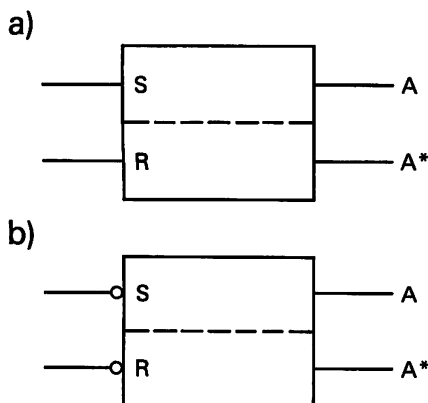


Bild B 38.1

Statisches RS-Speicherglied.

- a) steuernd wirkt Zustand 1 der Eingangsvariablen.
b) steuernd wirkt Zustand 0 der Eingangsvariablen.

Betrachten wir das RS-Speicherglied in Bild B 38.1a. Ein aktives Signal am S-Eingang ($S=1$) kippt das Speicherglied in den **Setzzustand**. In diesem Zustand haben die Ausgänge die Signalzustände $A=1$ und $A^*=0$. Nach dem Setzen des Speicherglieds bleibt dieser Signalzustand bestehen. Das **Rücksetzen** erfolgt mit einem aktiven Signal am R-Eingang ($R=1$). Nach dem Rücksetzen nehmen an den Ausgängen die Signalzustände die Werte $A=0$ und $A^*=1$ an. Ein Signalwechsel an den Ausgängen entsteht erst wieder mit einem erneuten Setzsignal. Das Speicherglied Bild B 38.1a benötigt zur Auslösung des Kippvorgangs am betreffenden Eingang ein 1-Signal, das Speicherglied in Bild B 38.1b dagegen ein 0-Signal. Wie Sie aus der Schaltung ablesen können, wirkt das invertierte Eingangssignal $S=0$ oder $R=0$ steuernd auf die Ausgänge A und A*.

Die Speicherglieder nach Bild B 38.1 werden auch als **asynchrone Speicherglieder** bezeichnet.

Zur schaltungstechnischen Anwendung muß das Schaltverhalten der Speicherglieder bekannt sein. In der praktischen Anwendung haben sich hierfür Tabellen und Diagramme bewährt. Als Tabellen haben

- die Funktionstabelle und
- die Schaltfolgetabelle

große Bedeutung. Die zeichnerische Auswertung der Funktionstabelle ergibt wieder das Signal-Zeit-Diagramm.

Die Funktionstabelle eines Speicherglieds und eines Verknüpfungsglieds werden unterschiedlich dargestellt. Der durch die Anregung über die Eingangsvariablen erzielte Zustand der Ausgangsvariablen eines Speicherglieds bleibt auch dann erhalten, wenn die Anregung nicht mehr wirksam ist. Unter Anregung versteht man die Wirkung eines Signals am Setz- oder Rücksetzeingang.

Zur Vereinfachung bezeichnen wir die steuernden Eingangsvariablen mit S und R. Für die Ausgangsvariablen gilt:

A_n, A_n^* Zustand der Ausgangsvariablen **vor** einer neuen Wertekombination am Eingang.

A_{n+1}, A_{n+1}^* Zustand der Ausgangsvariablen **nach** einer neuen Wertekombination am Eingang.

Für alle RS-Speicherglieder (wie in Bild B 38.1a) hat die folgende „ausführliche“ Funktionstabelle Gültigkeit. Die RS-Speicherglieder der verschiedenen Typen sind so entwickelt, daß sie die Signalfolge der Tabelle erfüllen.

Tabelle B 39.1: Ausführliche Funktionstabelle des RS-Speicherglieds nach Bild B 38.1a)

S	R	A_n	A_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

1-Signal rücksetzen

1-Signal setzen

Undefinierter Zustand der Ausgangssignale, verbotene Wertekombination für R und S am Eingang.

Die letzten beiden Zeilen der Tabelle B 39.1 bezeichnen einen nicht erlaubten Schaltzustand. Setz- und Rücksetzeingang dürfen nie gleichzeitig den Zustand 1 annehmen. In diesem Fall würde an den Ausgängen ein **undefinierter Signalzustand** entstehen. Mit schaltungstechnischen Maßnahmen ist dieser Betriebszustand zu verhindern. Es muß immer die Bedingung $SAR=0$ erfüllt sein.

In der vereinfachten Funktionstabelle, mit der man in der Praxis meist arbeitet, ist nur noch die unmittelbare Abhängigkeit der Ausgangsvariablen von den Eingangsvariablen aufgeführt.

Tabelle B 40.1: Vereinfachte Funktionstabelle des RS-Speicherglieds nach Bild B 38.1a)

S	R	A_{n+1}	A_{n+1}^*
0	0	A_n	A_n^*
1	0	1	0
0	1	0	1
1	1	x	x

keine Änderung des Signalzustands der Ausgänge
Speicherglied setzen
Speicherglied rücksetzen
undefinierter Signalzustand

Die Funktionstabelle ist so gestaltet, daß für jede Eingangskonfiguration die daraus folgenden Ausgangswerte abgelesen werden können. Eine **Zustandsfolgetabelle** enthält die umgekehrte Darstellung. Für eine gewünschte Zustandsfolge der Ausgangsvariablen kann man die erforderlichen Zustände der Eingangsvariablen ablesen. Für das RS-Speicherglied nach Bild B 38.1a gilt:

Tabelle B 40.2: Folgezustandstabelle des RS-Speicherglieds nach Bild B 38.1a)
x bedeutet, daß der Zustand der betreffenden Eingangsvariablen beliebig, also 1 oder 0 sein kann.

Zustandsfolge der Ausgangs- variablen $A_n \rightarrow A_{n+1}$	Signale an den Eingängen	
	S	R
0 \rightarrow 0	0	x
0 \rightarrow 1	1	0
1 \rightarrow 0	0	1
1 \rightarrow 1	x	0

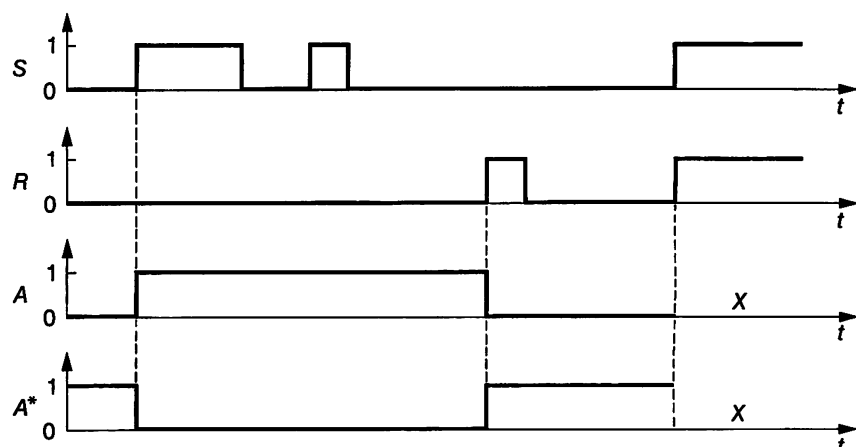


Bild B 40.1
Signal-Zeit-Diagramm des statischen RS-Speicherglieds, x kennzeichnet einen undefinierten Signalzustand der Ausgangsvariablen A bzw. A*. Die Signalkombination SAR=1 ist nicht erlaubt.

Die in der Folgezustandstabelle ausgedrückten Zusammenhänge kann man auch der ausführlichen Funktionstabelle entnehmen. Nach der Funktionstabelle ist das **Signal-Zeit-Diagramm** in Bild B 40.1 entwickelt. In diesem Diagramm sind die Zustände der Eingangsvariablen S und R als Beispiel vorgegeben. Aus der Tabelle B 40.2 können Sie die entsprechenden Zustände der Ausgangsvariablen ablesen und in das Diagramm eintragen.

Zusammenfassung

Die meisten Schaltungen in der Steuerungstechnik benötigen außer Verknüpfungsgliedern noch Funktionsglieder mit einem Speicherverhalten. Elektronische Speicherglieder sind aus den Kippschaltungen entwickelt worden. Ein Speicherglied hat die Speicherkapazität 1 Bit. Statisch gesteuerte bistabile Kippschaltungen besitzen zwei Steuerungseingänge. Ein Eingang dient zur Steuerung in die Arbeitslage (Setzustand), der andere zur Steuerung in die Ruhelage (Rücksetzustand). Beide Eingänge dürfen normalerweise nicht gleichzeitig angesteuert werden ($SAR=0$).

Die folgenden Beispiele und Aufgaben haben wir zunächst mit elektronischen Bauelementen gelöst. Nach Erweiterung unserer Programmierkenntnisse werden wir die Schaltungen zu einem späteren Zeitpunkt auch noch programmieren.

Beispiel B 41.1

Überwachungsschaltung

In der Schaltung nach Bild B 41.1 meldet das Signal E1 den Betriebszustand einer Anlage.

Zustand $E1=1$ bedeutet: ungestörter Betrieb,
Zustand $E1=0$ bedeutet: gestörter Betrieb

Signalgeber (Meldelampen und Hupe) und Signale sind gleich bezeichnet. Bei einer Störung soll Meldelampe A3 aufleuchten (Signal $A3=1$) und Hupe A2 ertönen (Signal $A2=1$). Die Meldung der Störung wird durch Betätigung des Signaltasters E2 bestätigt. Bei Tasterbetätigung entsteht ein Kurzzeitsignal $E2=1$. Dadurch sind folgende Funktionen auszulösen:

- Abschalten von Hupe A2 und Meldelampe A3,
- Einschalten von Meldelampe A4.

Nach Beseitigung der Störung ($E1=1$) soll die Meldelampe A4 löschen und Meldelampe A1 aufleuchten.

Festlegung der Signalzustände in der Schaltung:

- A1, A3, A4 haben Zustand 1: Meldelampe leuchtet.
- A1, A3, A4 haben Zustand 0: Meldelampe leuchtet nicht.

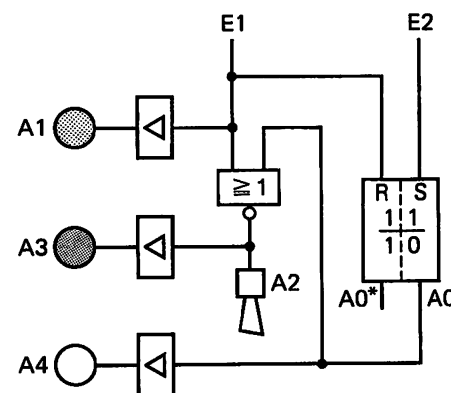


Bild B 41.1
Überwachungsschaltung zum Beispiel B 41.1. Schaltverstärker übernehmen die erforderliche Leistungsverstärkung der Ausgangssignale.

A2 hat Zustand 1: Hupe ist eingeschaltet.
 A2 hat Zustand 0: Hupe ist ausgeschaltet.

Wir wollen uns die Schaltung näher ansehen und die Funktionstabelle B42.1 überprüfen.

B**42**

Lösung:

Die Zahlen im Symbol des RS-Speicherglieds bedeuten:

Führen beide Eingänge gleichzeitig den Signalzustand 1 (verbotene Signalkombination), dann nehmen die Ausgänge folgende Signalzustände an: $A0=0$ und $A0^*=1$. Ein Speicherglied mit diesen Eigenschaften hat Rücksetzdominanz, d.h. das Signal am Rücksetzeingang bestimmt bei $S=R=1$ die Signalzustände am Ausgang.

Bei ungestörtem Betrieb ($E1=1$) ist das RS-Speicherglied rückgesetzt ($R=1$, $A0=0$). Mit $E1=1$ ist auch $A1=1$; die Ausgangssignale $A2$ und $A3$ sind Null, weil sie am negierten ODER-Glied-Ausgang anliegen. Bei einer Störung werden mit $E1=0$ die Ausgangssignale $A1=0$, $A3=1$ (Meldelampe leuchtet) und $A2=1$ (Hupe ertönt) erzeugt.

In der Funktionstabelle B 42.1 gehen wir vom ungestörten Betrieb aus. Nacheinander sind dann die Signalzustände in der Schaltung für folgende Betriebszustände eingetragen: Keine Störung – Störung – Meldung der Störung bestätigt – Störung noch vorhanden – Störung beseitigt. Die Schaltungsanalyse ergibt die eingetragenen Werte.

Überprüfen Sie bitte die in Tabelle B 42.1 eingetragenen Signalwerte.

Tabelle B 42.1: Funktionstabelle der Überwachungsschaltung von Beispiel B 41.1.

Betriebszustand	E1	E2	R	S	A0	A1	A2	A3	A4	Bemerkung
1	1	0	1	0	0	1	0	0	0	keine Störung
2	0	0	0	0	0	0	1	1	0	Störung
3	0	1	0	1	1	0	0	0	1	Meldung der Störung bestätigt
4	0	0	0	0	1	0	0	0	1	Störung noch vorhanden
5	1	0	1	0	0	1	0	0	0	Störung beseitigt

Zur Tabelle B 42.1 geben wir Ihnen einige Erläuterungen:

Zustand 3: Der Taster zur Bestätigung der Störungsmeldung wird nur kurzzeitig betätigt. Das Speicherglied geht dabei in den gesetzten Zustand, am Ausgang entsteht das Signal $A0=1$.

Zustand 4: Obwohl E2 wieder den Wert 0 angenommen hat, bleibt das Speicherglied im gesetzten Zustand.

Eine etwas einfachere Schaltung sollen Sie nun selbst untersuchen.

Aufgabe B 43.1

Meldeschtung

Der Signalzustand eines binären Signalgebers ist zu überwachen. Die Überwachungsschaltung zeigt das Bild B 43.1. E1 ist das Gebersignal. Bei $E1=0$ ist keine Störung vorhanden, mit $E1=1$ wird eine Störung des Signalgebers gemeldet. Die Störungsmeldung wird wieder durch Betätigung eines Tasters bestätigt, dabei nimmt E2 kurzzeitig den Wert 1 an. Im Störfall sollen Meldelampe A1 und Hupe A2 ansprechen. Die Hupe soll abschalten, wenn die Störung bestätigt ist.

- Ergänzen Sie die Funktionstabelle (Tabelle B 43.1)
- Prüfen Sie, ob die Signalzustände den eingetragenen Bemerkungen entsprechen.

Tabelle B 43.1: Funktionstabelle zur Aufgabe B 43.1

Zustand	E1	E2	R	S	A0	A0*	A1	A2	Bemerkung
1	0	0							keine Störung
2	1	0							Störung
3	1	1							Meldung quittiert
4	1	0							Störung noch vorhanden
5	0	0							Störung beseitigt

Die Lösung dieser Aufgabe finden Sie auf Seite F16.

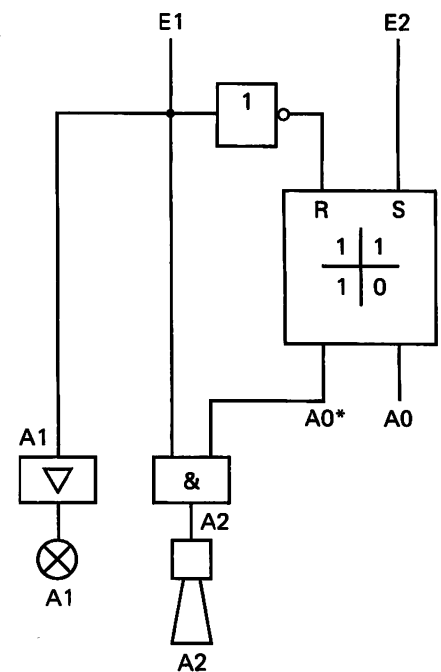


Bild B 43.1
Meldeschtung zur Aufgabe
B 43.1.

Statische und dynamische RS-Speicherglieder mit Taktsteuerung

Bei den bisher beschriebenen RS-Speichergliedern wird der Kippvorgang von einem Binärsignal (1 oder 0) ausgelöst. Das Speicherglied wird zum Kippen angeregt, solange der betreffende Signalzustand wirksam ist. Man bezeichnet diese Anregung deshalb auch als **Zustandssteuerung**. Speicherglieder mit Zustandssteuerung werden **statische Speicherglieder** genannt.

Bezogen auf dynamische RAM-Bausteine (DRAM) ist die Information „1Bit“ als elektrische Ladung in einer integrierten Kapazität gespeichert. Wegen des unvermeidbaren Entladestroms geht die Ladung verloren. Sie muß deshalb in bestimmten Zeitabständen (ca. 2ms) aufgefrischt werden.

Größere Bedeutung hat bei einem Speicherglied die Flankensteuerung. Bei ihr erfolgt die Anregung zum Kippen nur während des $0 \rightarrow 1$ oder $1 \rightarrow 0$ Sprungs des Steuersignals. Die hierfür gültigen Eingangssymbole enthält Tabelle B 38.1. Speicherglieder mit Flankensteuerung bezeichnet man als **Speicherglieder mit dynamischem Eingang**.

Bei diesen Speichergliedern wird die Eingangsinformation zum Kippen erst zu einem bestimmten Zeitpunkt, den ein **Taktimpuls** festlegt,

übernommen. Diese Speicherglieder erhalten einen zusätzlichen **Takteingang** (Clock-Eingang, C-Eingang).

Die Schaltzeichen der Speicherglieder mit Takteingang sind in Bild B 44.1 wiedergegeben. Beide Speicherglieder (b und c) enthalten sogenannte **Vorbereitungseingänge** und einen Takteingang. Erst mit dem Eintreffen des Taktimpulses wird die an den Vorbereitungseingängen liegende Information vom Speicherglied übernommen. Diese Steuerungsabhängigkeit zwischen mehreren Eingängen ist im Schaltzeichen wie folgt gekennzeichnet:

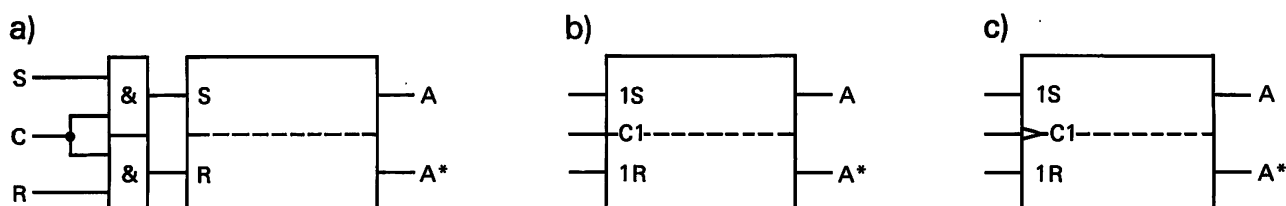


Bild B 44.1

RS-Speicherglied mit Taktsteuerung, Speicherglied mit

a) Taktzustandssteuerung, ausführliche Eingangserschaltung,

b) Taktzustandssteuerung, Schaltzeichen für a),

c) Taktflankensteuerung.

Der steuernde (das Kippen auslösende) Takteingang ist mit C bezeichnet, zusätzlich erhält er eine Zählnummer, z. B. 1 (C1). Die gesteuerten Eingänge, die Vorbereitungseingänge, haben die Kennbuchstaben S und R. Zusätzlich erhalten sie die gleiche Zählnummer wie der steuernde Takteingang (1S und 1R). Die Vorbereitungseingänge werden auch als **Informationseingänge** bezeichnet.

Nicht taktgesteuerte RS-FF nennt man auch **asynchrone Speicherglieder**, taktgesteuerte RS-FF werden **synchrone Speicherglieder** genannt.

Für das Schaltverhalten haben die bereits aufgestellten Tabellen weiter Gültigkeit. Der einzige Unterschied zum asynchronen RS-FF besteht darin, daß die an den Informationseingängen liegenden Werte erst mit dem Taktsignal übernommen werden. Einen derartigen Vorgang zeigt das Diagramm in Bild B 44.2.

Das taktgesteuerte RS-FF mit Zustandssteuerung nach Bild B 44.1b wird auch als „latch“ oder „Auffang-FF“ bezeichnet. Die beschriebenen Bausteine gehören, wie bereits erwähnt, zur Gruppe der „synchrone Speicherglieder“. Schaltungen mit diesen Speichergliedern

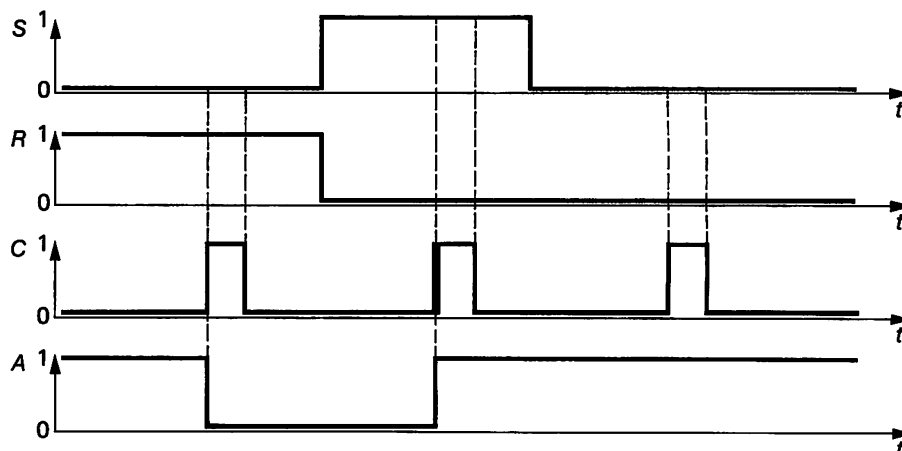


Bild B 44.2

Informationssteuerung mit zustandsgesteuertem Takteingang.

lassen sich über das Taktsignal synchronisieren. Sie werden derartige Schaltungen noch kennenlernen.

Falls Sie der Meinung sein sollten, daß wir nun alle RS-FFs besprochen hätten, so müssen wir Sie leider enttäuschen. Das beste Stück, also das wichtigste RS-Speicherglied kommt noch. Mit diesem Bauelement, das wir nachstehend besprechen werden, sind die meisten synchronisierten Schaltungen, wie z.B. Schieberegister, Zähler und Frequenzteiler aufgebaut.

RS-Master-Slave-FF

Dieses Flipflop kann mit Zustands- oder Flankensteuerung getaktet werden. Sein Schaltschema mit Zustandssteuerung zeigt das Bild B 45.1. Hauptmerkmal des Bausteins ist die Anordnung zweier hinter-

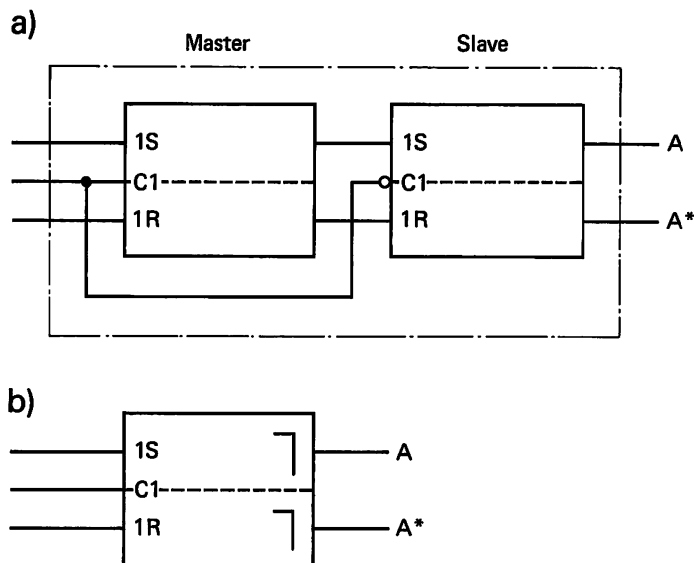


Bild B 45.1
Master-Slave-RS-Speicherglied
mit Taktzustandssteuerung.
a) ausführliche Schaltung,
b) Schaltzeichen.

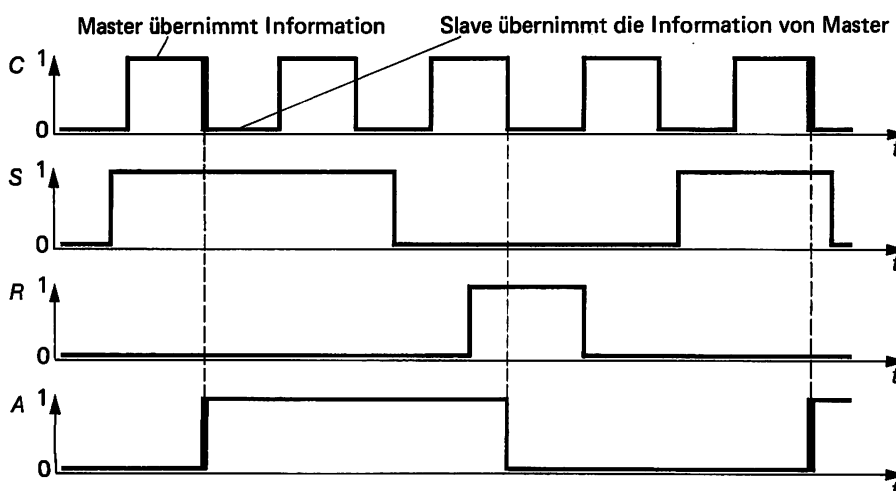


Bild B 45.2
Signal-Zeit-Diagramm des MS-RS-
FF mit Taktzustandssteuerung.

einander liegender RS-Speicherglieder. Das an erster Stelle liegende Speicherglied, der **Master** (master = Meister) nimmt bei $C1=1$ die an den Informationseingängen liegende Information auf und speichert sie. Das an zweiter Stelle liegende Speicherglied, der **Slave** (slave = Sklave), kann die Information vom Master nicht übernehmen, da dessen Takteingang nicht auslösend angesteuert ist (Negation am Takteingang). Beim Übergang des Eingangstaktes von 1 auf 0 wird der Master gesperrt und der Slave angesteuert. Der Slave übernimmt die Information vom Master und speichert sie. Sie steht jetzt am Ausgang A zur Verfügung.

Wie beschrieben, läuft der Speichervorgang in zwei Phasen ab. In der ersten Phase ($C1=1$) wird die zu speichernde Information aufgenommen, sie gelangt jedoch erst in der zweiten Phase ($C1=0$) an die Ausgänge. Zwischen Signalaufnahme und Signalausgabe besteht eine Zeitdifferenz die aufgenommene Information wird verzögert auf den Ausgang geschaltet. Der in das Schaltzeichen (Bild B 31.1b) eingetragene „Haken“ kennzeichnet das beschriebene Verhalten. Man bezeichnet solche Ausgänge als **retardierte** Ausgänge.

Im Signal-Zeit-Diagramm von Bild B 45.2 können Sie die verzögernde Wirkung gut erkennen. Diese komplexen Speicherbausteine in diskreter Bauweise erfordern einen erheblichen Schaltungsaufwand. Ihre technische Realisierung mit wirtschaftlichem Aufwand ermöglichte die Einführung der integrierten Schaltungstechnik. Für das Schaltverhalten dieses RS-Speicherbausteins haben die bisher betrachteten Tabellen ebenfalls Gültigkeit.

Zusammenfassung

Das RS-FF mit Takteingang und Informationseingängen ermöglicht die Herstellung synchronisierbarer Schaltungen. Der Taktimpuls bestimmt den Zeitpunkt der Informationsübernahme. Synchron arbeitende Schaltungen benötigen Speicherbausteine, bei denen zwischen Informationsaufnahme und Informationsausgabe eine Verzögerung besteht. Diese Schaltungen sind nur mit Master-Slave-FFs realisierbar. Durch Änderung der Eingangsschaltung und Einbau einer Rückkopplung vom Ausgang zum Eingang kann man dem Speicherglied noch andere Schalteigenschaften geben. Auf Seite G7 (Tafel 6) finden Sie die wichtigsten Speicherglieder zusammengestellt und kommentiert. Das RS-Speicherglied ist die Ausgangsschaltung für die Programmierung von Speicherfunktionen. Wir haben deshalb diese Schaltung ausgewählt und ausführlich besprochen.

Theoreme von de Morgan

Die Gesetze der NAND- und NOR-Verknüpfungen wurden schon auf Seite B7, Lehrbrief 1 aufgeführt. Wir müssen auf diese Gesetzmäßigkeit noch einmal näher eingehen. In den VPS finden Sie viele elektronische Schaltkreise mit NAND- bzw. mit NOR-Gattern ausgeführt. Diese Gatter lassen sich z.B. in TTL-Technik einfach und wirtschaftlich produzieren, außerdem sind sie universell einsetzbar. Mit einem Gattertyp (NAND oder NOR) kann man die Funktionstabellen aller übrigen Gatter (UND, ODER und NICHT) nachbilden (vgl. auch Tabelle B 8.1).

Die rechnerische Umsetzung der Schaltung, bestehend aus UND, ODER und NICHT in NAND bzw. NOR erfolgt mit den de Morganschen Theoremen. Bei der Umsetzung von Schaltnetzen in ein Programm können auch wir mit der NAND- bzw. der NOR-Technik konfrontiert werden. Die nachstehend erläuterten Gesetze erleichtern das Umsetzen von Hardware in Software erheblich. Die de Morganschen Theoreme umfassen die nachstehend erläuterten und in Beispielen angewendeten Gesetze.

Erstes de Morgansches Theorem

$$A1 = \overline{E1 \wedge E2} = \overline{E1} \vee \overline{E2}$$

Das Gesetz gibt die Umwandlung einer NAND-Verknüpfung wieder:

Die NAND-Verknüpfung der Variablen E1 und E2 kann durch eine ODER-Verknüpfung ersetzt werden. Vor der ODER-Verknüpfung müssen jedoch die beiden Variablen negiert werden.

Die Richtigkeit der Aussage beweist folgende Funktionstabelle:

Tabelle B 47.1: Funktionstabelle zum Ersten de Morganschen Theorem

$$\overline{E1 \wedge E2} = \overline{E1} \vee \overline{E2}$$

1	2	3	4	5	6
E2	E1	$\overline{E1 \wedge E2}$	$\overline{E2}$	$\overline{E1}$	$\overline{E1} \vee \overline{E2}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

Verknüpft werden die Variablen der Spalten 1 und 2. Spalte 3 enthält das Ergebnis der NAND-Verknüpfung. Nach der Negation der Eingangsvariablen (Spalten 4 und 5) zeigt Spalte 6 das Ergebnis der ODER-Verknüpfung. Die Ergebnisse der Spalten 3 und 6 stimmen überein. Somit hat das Erste de Morgansche Theorem Gültigkeit.

Kontaktplan der NAND-Verknüpfung

Für die NAND-Verknüpfung gibt es verschiedene Darstellungsformen. Die in Bild B 48.1 dargestellten Kontaktpläne sind gleichwertig.

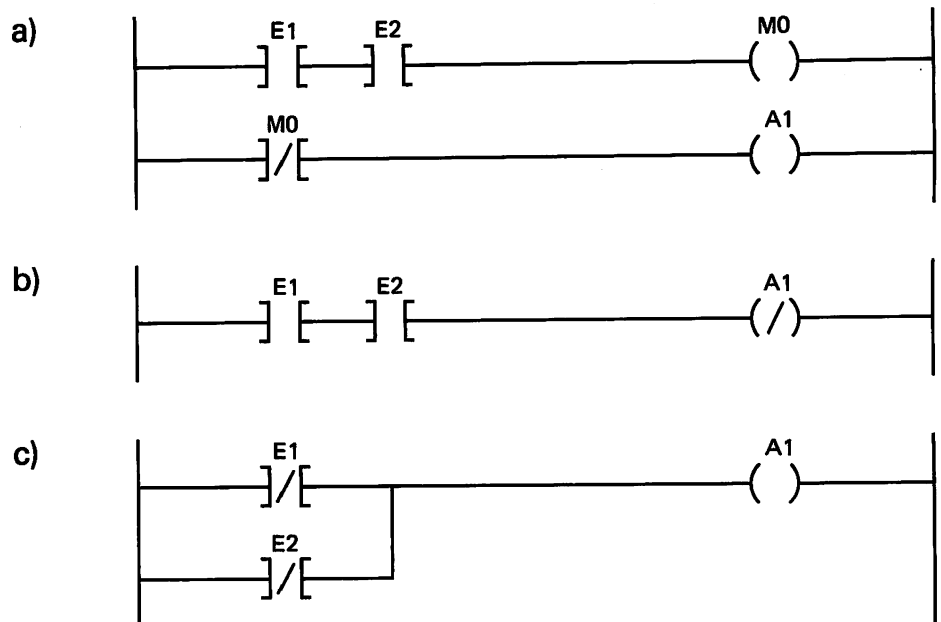
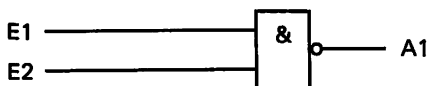


Bild B 48.1
Möglichkeiten der Kontaktplan-
darstellung einer NAND-Ver-
knüpfung mit zwei Eingangsvari-
ablen, $A1 = \overline{E1 \wedge E2}$.

KOP a) Das Ergebnis der UND-Verknüpfung wird in einen Merker gesetzt. Anschließend erfolgt die invertierte Merkerabfrage und die Zuweisung auf einen Ausgang.

$$E1 \wedge E2 = M0$$

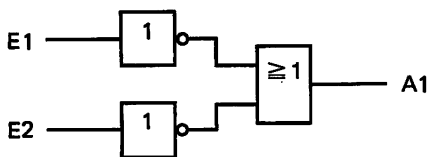
$$\overline{M0} = A1, \text{ somit } \overline{E1 \wedge E2} = A1$$



KOP b) Das Ergebnis der UND-Verknüpfung wird dem Ausgang negiert zugewiesen.

$$E1 \wedge E2 = \overline{A1}$$

$$\overline{E1 \wedge E2} = \overline{\overline{A1}}, \text{ somit } \overline{E1 \wedge E2} = A1$$



KOP c) Anwendung des Ersten de Morganschen Theorems:

$$\overline{E1} \vee \overline{E2} = A1, \text{ somit}$$

$$\overline{E1 \wedge E2} = A1$$

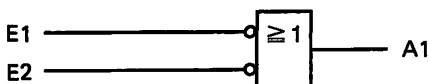


Bild B 48.2
Möglichkeiten der Funktionsplan-
darstellung einer NAND-Ver-
knüpfung mit zwei Eingangsvari-
ablen.

Die Symbole für den FUP der NAND-Verknüpfung enthält das Bild B 48.2.

Aufgabe B 48.1

Welche Verknüpfung erfolgt mit

a) $\overline{\overline{E1} \wedge \overline{E2}} =$ und b) $\overline{\overline{\overline{E1} \wedge \overline{E2}}} = ?$

Die Lösung dieser Aufgabe finden Sie auf Seite F 27.

Zweites de Morgansches Theorem

$$A1 = \overline{E1 \vee E2} = \overline{E1} \wedge \overline{E2}$$

Dieses Gesetz gibt die Umwandlung der NOR-Verknüpfung wieder, es besagt:

Die NOR-Verknüpfung der Variablen E1 und E2 kann durch eine UND-Verknüpfung der beiden Variablen ersetzt werden, sofern diese den UND-Eingängen negiert zugeführt werden.

Den Nachweis für die Gültigkeit des Gesetzes sollen Sie selbst erbringen.

Aufgabe B49.1

Vervollständigen Sie die Funktionstabelle für die NOR-Verknüpfung.

- In die Funktionstabelle B49.1 sind die Werte der Variablen und der Verknüpfungen einzutragen.
- Vergleichen Sie die Spalten 4 und 7 miteinander.

Tabelle B49.1: Funktionstabelle zur Aufgabe B49.1

1	2	3	4	5	6	7
E2	E1	$E1 \vee E2$	$\overline{E1 \vee E2}$	$\overline{E2}$	$\overline{E1}$	$\overline{E1} \wedge \overline{E2}$
0	0					
0	1					
1	0					
1	1					

Die Lösung dieser Aufgabe finden Sie auf Seite F27.

Kontaktplan der NOR-Verknüpfung

Auch für diese Verknüpfung gibt es verschiedene, gleichwertige Darstellungen. Die Pläne sollen Sie selbst entwerfen und erhalten dafür einige Hinweise:

Aufgabe B49.2

Entwurf der Kontaktpläne für die NOR-Verknüpfung

Für folgende, schaltalgebraische Formulierungen ist der KOP aufzuzeichnen:

a) $\overline{E1 \vee E2} = A1$

$E1 \vee E2 = M0$, somit ist $\overline{M0} = A1$

$$\text{b) } \overline{\overline{E1} \vee \overline{E2}} = A1$$

$$\overline{\overline{E1} \vee \overline{E2}} = \overline{A1}, \text{ somit ist } E1 \vee E2 = \overline{A1}$$

$$\text{c) } \overline{\overline{E1} \vee \overline{E2}} = A1$$

$$\overline{E1} \wedge \overline{E2} = A1$$

Die Lösung finden Sie auf Seite F27.

B**50**

Funktionsplan der NOR-Verknüpfung

Die Symbole des Funktionsplans sollen Sie ebenfalls selbst entwickeln.

Aufgabe B 50.1

Zeichnen Sie die NOR-Funktionsplansymbole für:

$$\text{a) } \overline{\overline{E1} \vee \overline{E2}} = A1 \quad \text{und} \quad \text{b) } \overline{E1} \wedge \overline{E2} = A1$$

Die Lösung finden Sie auf Seite F28.

Für die besprochenen Gesetzmäßigkeiten gelten folgende Programme:

Erstes de Morgansches Theorem:

$$\begin{array}{lll} \text{a) } !E1 \& E2 = M0 & \text{b) } !E1 \& E2 = NA1 & \text{c) } !NE1 / NE2 = A1 \\ !NM0 = A1 & !PE & !PE \\ !PE & & \end{array}$$

Diese Programme entsprechen den Kontaktplänen in Bild B 48.1.

Zweites de Morgansches Theorem:

$$\begin{array}{lll} \text{a) } !E1 / E2 = M0 & \text{b) } !E1 / E2 = NA1 & \text{c) } !NE1 \& NE2 = A1 \\ !NM0 = A1 & !PE & !PE \\ !PE & & \end{array}$$

Die Programme entsprechen den Formulierungen in Aufgabe B 49.2.

Anwendungen der de Morganschen Theoreme auf Verknüpfungen mit mehr als zwei Eingangsvariablen

Die eben besprochenen Theoreme haben nicht nur für Verknüpfungen von zwei Variablen Gültigkeit, sondern lassen sich auch auf Verknüpfungen von beliebig vielen Variablen anwenden.

Allgemein gilt:

$$\overline{E1 \wedge E2 \wedge E3 \wedge E4 \wedge \dots} = \overline{E1} \vee \overline{E2} \vee \overline{E3} \vee \overline{E4} \vee \dots$$

$$\overline{E1 \vee E2 \vee E3 \vee E4 \vee \dots} = \overline{E1} \wedge \overline{E2} \wedge \overline{E3} \wedge \overline{E4} \wedge \dots$$

Analyse und Programmierung von Schaltnetzen mit NAND- und NOR-Verknüpfungen

Im folgenden Kapitel werden vorgegebene Schaltnetze analysiert, untersucht und programmiert. Ausgewählt sind wieder Grundschaltungen, die in der Steuerungstechnik und Digitaltechnik vielfältige Anwendung finden. Die vorgegebene Hardwarelösung ist in ein Programm umzusetzen. Die Beispiele enthalten nicht immer optimale Lösungen. Es wurde Wert auf eine anschauliche Umsetzung von Hardware in Software gelegt. Betrachten Sie bitte von diesem Standpunkt aus die folgenden Beispiele und Lösungsvorschläge für die gestellten Aufgaben.

Die Digitaltechnik erscheint jetzt nicht mehr als eigenständiges Fachgebiet. In unseren Anwendungen erfüllt sie eine Hilfsfunktion. Mit ihr werden die gestellten Aufgaben soweit als möglich zur Programmierung aufbereitet. Die folgenden Beispiele werden Ihnen wieder zeigen, daß das erfolgreiche Arbeiten mit der SPS fundierte Kenntnisse der Digitaltechnik voraussetzt. Dieser Hinweis kann nicht oft genug wiederholt werden — er ist ein Erfahrungswert aus der Praxis.

Beispiel B51.1

Analyse eines Schaltnetzes mit NOR-Verknüpfungen

- Für das Schaltnetz in Bild B51.1 sind die Schaltfunktionen für die Ausgangsvariablen A0 bis A2 aufzustellen.
- Die Funktionstabelle B51.1 ist mit Hilfe der Schaltfunktionen auszufüllen.

Tabelle B51.1: Funktionstabelle zum Beispiel B51.1

E2	E1	A0	A1	A2
0	0			
0	1			
1	0			
1	1			

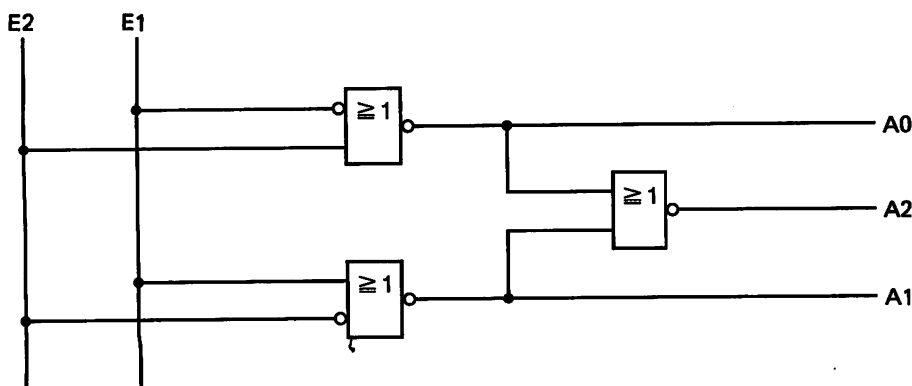


Bild B51.1
Schaltnetz zum Beispiel B51.1.

Lösung:

a) Schaltfunktionen für die Ausgangsvariablen:

Aus der Schaltung in Bild B 51.1 können Sie unmittelbar die folgenden Schaltfunktionen ablesen:

$$A0 = \overline{\overline{E1} \vee \overline{E2}} = \overline{\overline{E1}} \wedge \overline{\overline{E2}} = E1 \wedge \overline{E2}$$

$$A1 = \overline{E1 \vee \overline{E2}} = \overline{E1} \wedge \overline{\overline{E2}} = \overline{E1} \wedge E2$$

$$A2 = \overline{A0 \vee A1} = \overline{A0} \wedge \overline{A1}$$

In der letzten Schaltfunktion sind A0 und A1 durch die aufgestellten Ausdrücke zu ersetzen und zu negieren.

Wir erhalten:

$$A0 = E1 \wedge \overline{E2}, \text{ die Umkehrung ergibt:}$$

$$\overline{A0} = \overline{E1 \wedge \overline{E2}} = \overline{E1} \vee E2$$

$$A1 = \overline{E1} \wedge E2, \text{ die Umkehrung ergibt:}$$

$$\overline{A1} = \overline{\overline{E1} \wedge E2} = E1 \vee \overline{E2}$$

Diese Ausdrücke sind nun in die Funktion für A2 einzusetzen.

$$A2 = (\overline{E1} \vee E2) \wedge (E1 \vee \overline{E2})$$

Nach dem Auflösen der Klammern wenden Sie die bereits bekannten Regeln an und erhalten:

$$A2 = \underbrace{\overline{E1}E1}_0 \vee \overline{E1}\overline{E2} \vee E2E1 \vee \underbrace{E2\overline{E2}}_0$$

Das Ergebnis lautet:

$$A2 = \overline{E1}\overline{E2} \vee E1E2$$

b) Die abgeleiteten Schaltfunktionen ergeben folgende Funktionstabelle:

Tabelle B 52.1: Funktionstabelle zum Beispiel B 51.1

E2	E1	A0	A1	A2
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Die untersuchte Schaltung wird als **Vergleicherschaltung** bezeichnet. Verglichen werden die binären Zustände der Variablen E1 und E2.

Wir legen fest:

Zustand 1 ist größer als Zustand 0. Bei A0=1 ist E1 > E2, bei A1=1 ist E1 < E2 und bei A2=1 ist E1=E2.

Beispiel B53.1**Programmierung einer Vergleicherschaltung**

Programmieren Sie bitte die Schaltung von Beispiel B 51.1 (Bild B 51.1).

- Zeichnen Sie den FUP für die Schaltung in Bild B 51.1.
- Programmieren Sie den FUP. Die Ergebnisse der ODER-Verknüpfungen sind in Merker zu setzen.
- Mit Hilfe des Zweiten de Morganschen Theorems sind die NOR-Verknüpfungen umzuformen. Zeichnen Sie den FUP mit den sich daraus ergebenden Verknüpfungen.
- Der FUP nach c) ist zu programmieren.
- Testen Sie die Programme von b) und d) und überprüfen Sie experimentell die Funktionstabelle B 52.1.

Lösung:

- Die Schaltung lässt sich direkt in den FUP (Bild B 53.1) umsetzen. Eingetragen sind die zur Programmierung erforderlichen Merker.

- Für den FUP a) erhält man folgendes Programm:

```
!NE1/E2=M0
!NM0=A0
!E1/NE2=M1
!NM1=A1
!A0/A1=M2
!NM2=A2
!PE
```

- Die Anwendung des Zweiten de Morganschen Theorems ergibt für die Verknüpfungen folgende Funktionen:

$$\overline{\overline{E1} \vee E2} = A0 \rightarrow E1 \wedge \overline{E2} = A0$$

$$\overline{E1 \vee \overline{E2}} = A1 \rightarrow \overline{E1} \wedge E2 = A1$$

$$\overline{A0 \vee A1} = A2 \rightarrow \overline{A0} \wedge \overline{A1} = A2$$

Damit entsteht ein FUP nach Bild B 53.2.

- Für den FUP von Bild B 53.2 lautet das Programm:

```
!E1&NE2=A0
!NE1&E2=A1
!NA0&NA1=A2
!PE
```

- Die Testergebnisse der Programme nach b) und d) stimmen mit den Werten der Funktionstabelle B 52.1 überein.

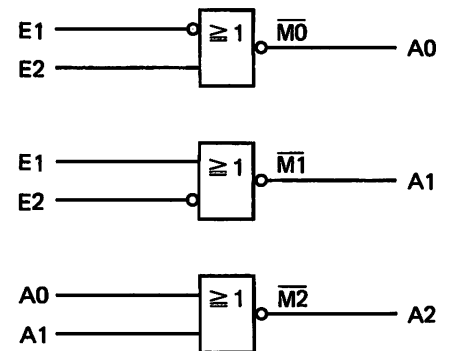


Bild B 53.1
Funktionsplan der Vergleicherschaltung, Beispiel B 53.1, Punkt a).

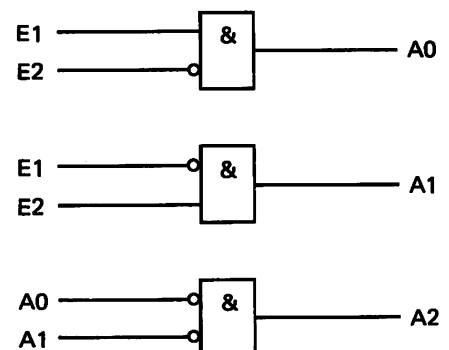


Bild B 53.2
Umgeformter Funktionsplan der Vergleicherschaltung, Beispiel B 53.1, Punkt c).

Beispiel B 54.1

Schaltungsumformung mit Hilfe der de Morganschen Theoreme

- a) Die Schaltfunktionen für die Ausgangsvariablen A1 und A2 in der Schaltung von Bild B 54.1 sind aufzustellen. Durch Anwendung der de Morganschen Theoreme lassen sich Invertierungen der Verknüpfungen einsparen.
- b) Wie bezeichnet man die Schaltfunktionen A1 und A2?

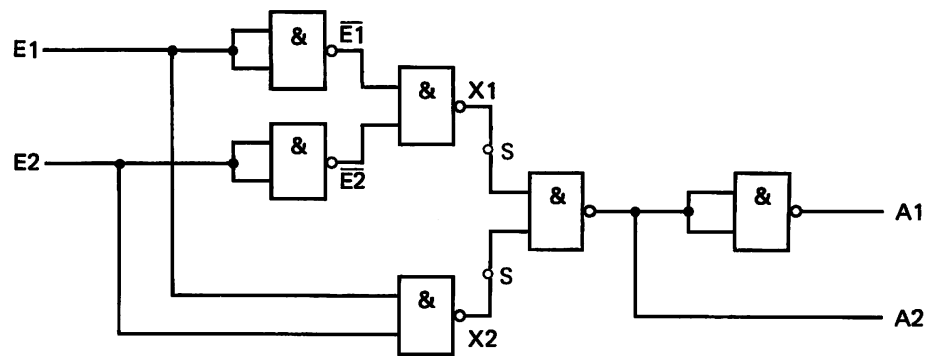


Bild B 54.1
Schaltnetz zum Beispiel B 54.1,
S Schnittstellen.

Lösung:

- a) Aufstellen und Umformen der Schaltfunktionen:

Wir stellen zuerst die Schaltfunktion für die Variable A2 auf. Die Schaltfunktion für Variable A1 erhält man durch Invertierung der Schaltfunktion A2.

Schaltfunktion A2:

Die eingangsseitigen NAND-Verknüpfungen bewirken eine Signalumkehr. Für die Hilfsvariablen X1 und X2 erhält man:

$$X1 = \overline{E1} \wedge \overline{E2} = \overline{E1} \vee \overline{E2} = E1 \vee E2$$

$$X2 = \overline{E1} \wedge E2 = \overline{E1} \vee \overline{E2}$$

Nach der Schaltung besteht für A2 folgende Beziehung:

$$A2 = \overline{X1} \wedge \overline{X2} = \overline{X1} \vee \overline{X2}$$

Die Hilfsvariablen X1 und X2 in der Schaltfunktion für A2 sind nunmehr durch die aufgestellten Funktionen zu ersetzen.

$$A2 = \overline{E1 \vee E2} \vee \overline{\overline{E1} \vee \overline{E2}} = (\overline{E1} \wedge \overline{E2}) \vee (E1 \wedge E2)$$

$$A2 = \overline{E1} \overline{E2} \vee E1 E2$$

Schaltfunktion A1:

Wir erhalten die Lösung durch Negation der Schaltfunktion für A2. Im Ansatz wird die Funktion mit Hilfsvariablen benutzt.

$$A1 = \overline{A2} = \overline{\overline{X1} \wedge \overline{X2}} = X1 \wedge X2$$

Nunmehr sind die Hilfsvariablen wieder zu ersetzen. Wir erhalten:

$$A1 = (E1 \vee E2) \wedge (\overline{E1} \vee \overline{E2})$$

Nach den Ihnen bekannten Regeln werden die Klammern beseitigt. Es entsteht folgende Lösung:

$$A1 = E1\overline{E1} \vee E1\overline{E2} \vee E2\overline{E1} \vee E2\overline{E2}$$

$$A1 = E1\overline{E2} \vee \overline{E1}E2$$

- b) Die Verknüpfung nach der Schaltfunktion A1 wird als **Antivalenz** oder **Exklusiv-ODER** bezeichnet. Nur bei ungleichen Zuständen der Eingangsvariablen nimmt die Ausgangsvariable den Zustand 1 an.

Die Schaltfunktion A2 wird als Äquivalenz bezeichnet. Sie liefert nur bei gleichen Zuständen der Eingangsvariablen den Wert A2=1.

Beispiel B55.1

Entwicklung von FUP und Programm für die Schaltung von Beispiel B54.1.

- Entwickeln Sie den FUP für die Schaltung von Bild B 54.1 in aufgelöster Darstellung. Zur Auftrennung der Schaltung sind die mit S bezeichneten Schnittstellen zu wählen.
- Programmieren Sie nach a) den FUP.
- Durch Anwendung des Ersten de Morganschen Theorems ist der FUP umzuwandeln. Stellen Sie auch für diese Darstellung das Programm auf.
- Testen Sie die Programme von b) und c) und überprüfen Sie bitte experimentell die Aussage von Punkt b) im Beispiel B 54.1.

Lösung:

- Den aufgeschnittenen FUP zeigt das Bild B 55.1. In den Plan sind die zur Aufstellung des Programms benutzten Merker eingetragen.
- Programm:
Die Ergebnisse der UND-Verknüpfungen sind in die Merker M3 bis M5, die Werte der Hilfsvariablen sind in die Merker M1 und M2 gesetzt.

$$!NE1 \& NE2 = M3$$

$$!NM3 = M1$$

$$!E1 \& E2 = M4$$

$$!NM4 = M2$$

$$!M1 \& M2 = M5$$

$$!NM5 = A2$$

$$!NA2 = A1$$

$$!PE$$

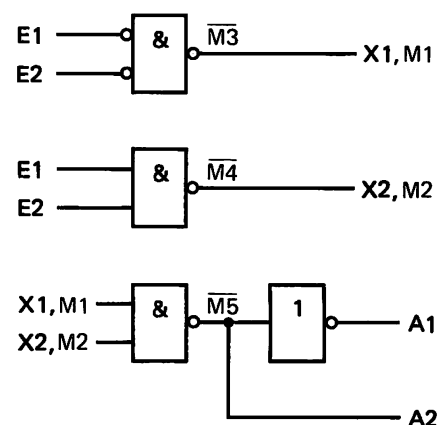


Bild B 55.1
Aufgelöster Funktionsplan vom Schaltnetz Bild B 54.1, Beispiel B 55.1, Punkt a).
M zur Programmierung benutzte Merker.

B**56**

- c) Umformung des FUP: Nach dem Ersten de Morganschen Theorem gelten folgende Umformungen:

$$\overline{\overline{E1} \wedge \overline{E2}} = E1 \vee E2 = M1$$

$$\overline{E1 \wedge E2} = \overline{E1} \vee \overline{E2} = M2$$

$$\overline{M1 \wedge M2} = \overline{M1} \vee \overline{M2} = A2$$

$$\overline{\overline{M1} \wedge \overline{M2}} = M1 \wedge M2 = A1 = \overline{A2}$$

Den hierfür gültigen FUP zeigt das Bild B 56.1. Dafür lautet das Programm:

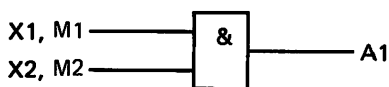
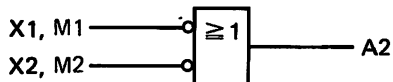
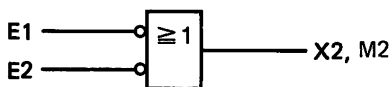
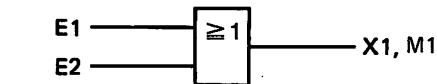
!E1/E2=M1

!NE1/NE2=M2

!NM1/NM2=A2

!M1&M2=A1

!PE



- d) Der Test beider Programme stimmt mit der Aussage von Punkt b) in Beispiel B 54.1 überein.

Bild B 56.1

Umgeformter Funktionsplan von Bild B 55.1, Beispiel B 55.1, Punkt c).
M zur Programmierung benutzte Merker.

Aufgabe B 56.1

Programmierung eines Halbaddierers

Das Bild B 56.2 zeigt eine Rechenschaltung. Sie wird in der Arithmetik mit Dualzahlen angewendet. Die Grundzüge der Dualarithmetik lernen Sie noch. Man bezeichnet die Schaltung als **Halbaddierer**.

- a) Programmieren Sie das Schaltnetz von Bild B 56.2. Formen Sie es vor der Programmierung mit dem Ersten de Morganschen Theorem um. Die Hilfsvariablen X sind in Merker zu setzen. Sie können die Operanden der Merker selbst wählen.
- b) Ergänzen Sie bitte experimentell die Funktionstabelle B 57.1.

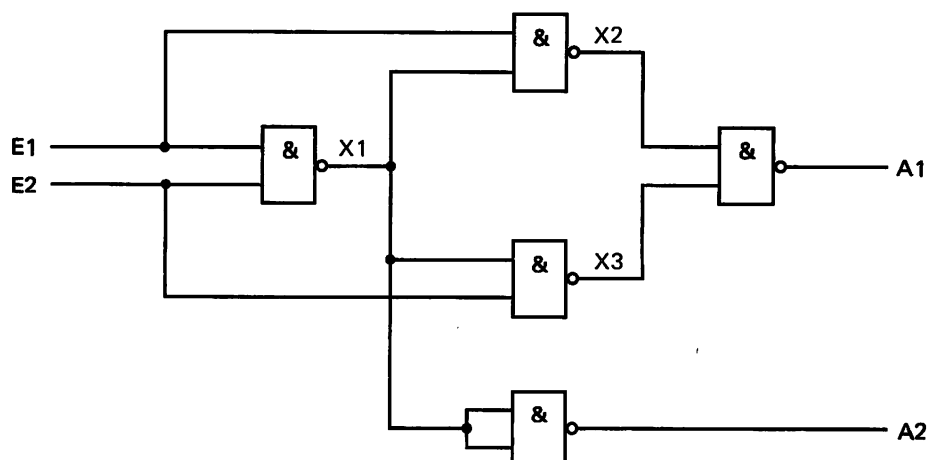


Bild B 56.2

Schaltung eines Halbaddierers mit NAND-Verknüpfungen.

Tabelle B 57.1: Funktionstabelle zur Aufgabe B 56.1

E2	E1	A1	A2
0	0		
0	1		
1	0		
1	1		

- c) Entwickeln Sie nach der experimentellen Ergänzung der Funktionstabelle B 57.1 die Schaltfunktionen für die Variablen A1 und A2. Es dürfen nur die Grundfunktionen UND, ODER und NICHT verwendet werden.
- d) Programmieren und testen Sie die Schaltfunktionen von c). Überprüfen Sie experimentell die Funktionstabelle B 57.1. Die Ergebnisse von a) und c) müssen übereinstimmen.

Die Lösung dieser Aufgabe finden Sie auf Seite F 28.

Schwellenwertschaltung

Bei einer Schwellenwertschaltung darf der Signalausgang nur dann den Zustand 1 annehmen, wenn mindestens eine vorgegebene Anzahl der Signaleingänge den Zustand 1 führt. Die vorgegebene Zahl der Eingänge heißt „Schwellenwert s “. Daraus folgt, daß die Zahl der Signaleingänge größer oder gleich „ s “ sein muß.

Das Bild B 57.1 zeigt eine Blockschaltung für $s=3$. Die Ausgangsvariable darf demnach nur dann den Wert 1 haben, wenn mindestens drei der fünf Signaleingänge Zustand 1 führen. Es ist gleichgültig, welche Eingänge diesen Wert haben.

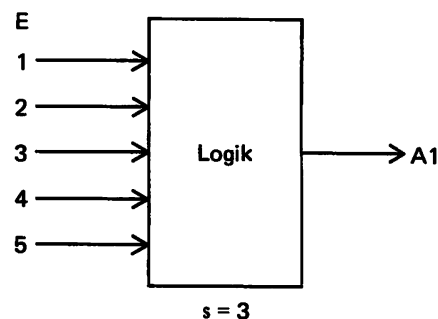


Bild B 57.1
Blockschaltung der Schwellenwertschaltung mit 5 Eingängen.

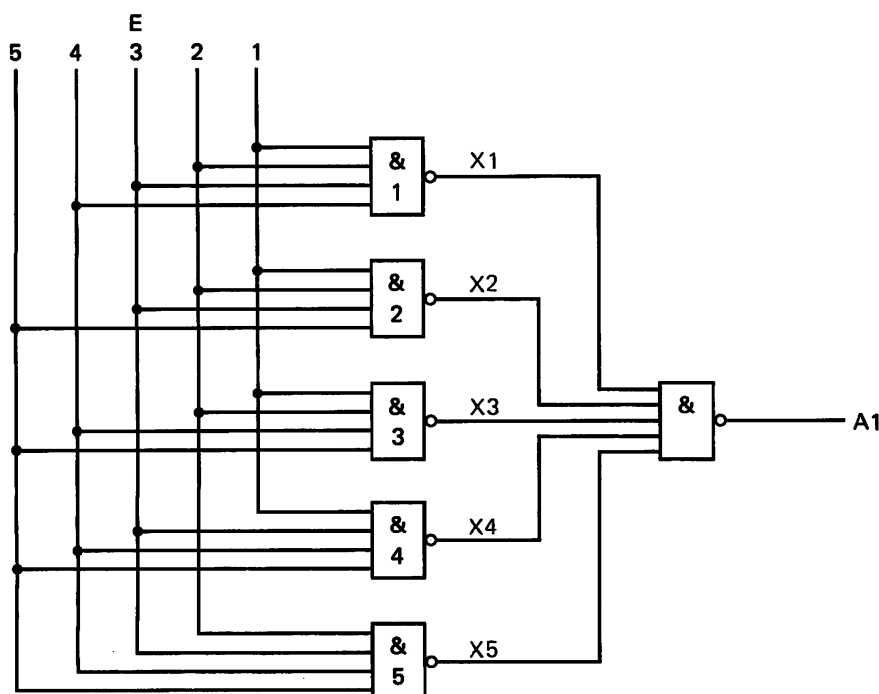


Bild B 57.2
Schwellenwertschaltung für $s=4$ in NAND-Technik (Aufgabe B 58.1).

Aufgabe B 58.1**Programmierung einer Schwellenwertschaltung**

Das Bild B 57.2 zeigt eine Schwellenwertschaltung für $s=4$ in NAND-Technik. Die Funktion der Schaltung ist durch ein Programm nachzubilden. Formen Sie vor der Programmierung die Schaltung nach dem Ersten de Morganschen Theorem um.

- Führen Sie die Schaltungsumformung durch. Zeichnen Sie danach den FUP auf und stellen Sie das Programm auf.
- Testen Sie Ihr Programm durch Annahme verschiedener Betriebszustände. Haben vier oder mehr der fünf Eingangsvariablen den Wert 1, dann muß auch $A1=1$ sein. Haben weniger als vier Variable den Zustand 1, dann muß die Ausgangsvariable A1 den Wert 0 haben.

Die Lösung finden Sie auf Seite F29.

Elektronische Schalter

In der Schaltungstechnik der VPS finden Sie elektronische Schalter für vielfältige Aufgaben eingesetzt. Mit elektronischen Bauelementen wird die Funktion der mechanischen oder elektromechanischen Schalter nachgebildet. Aus der Vielzahl der verschiedenen Schalter und Schalterfunktionen sind nachstehend nur einige wichtige aufgeführt.

- Informationsweiche: Mit einem Steuersignal kann von mehreren anliegenden Informationen eine bestimmte Information zur weiteren Verarbeitung ausgewählt werden.
- EIN/AUS-Schalter zur Freigabe oder Sperrung von Signalleitungen.
- Umschalter und Drehschalter: Sie werden zur Steuerung der Informationsauswahl und Weitergabe benötigt.

Einige typische Beispiele, die in der Hardwarelösung mit NAND- oder NOR-Bausteinen verwirklicht sind, haben wir ausgewählt. Die Funktion dieser Schalter ist zu programmieren. Auch in Programmen werden zur Steuerung des Informationsflusses derartige Verknüpfungen benötigt.

Aufgabe B 58.2**Nachbildung der Funktion einer Informationsweiche**

Die Informationsweiche in Bild B 59.1 hat die beiden Informationseingänge E1, E2 und den Steuereingang E0. Der Zustand von E0 bestimmt, ob E1 oder E2 auf den Ausgang A0 geschaltet ist. Mit der Schaltung ist es möglich, von zwei verschiedenen Informationen eine auszuwählen. Bei $E0=1$ ist die Information E1, bei $E0=0$ ist die Information E2 am Ausgang A0 wirksam.

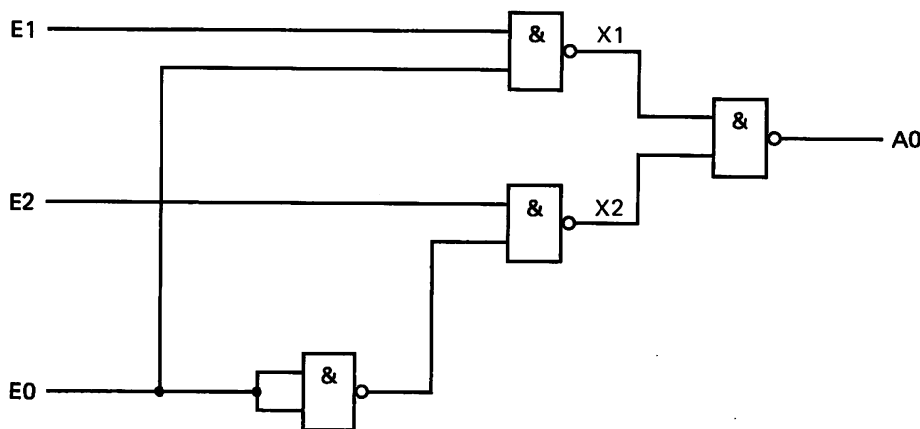


Bild B 59.1
Informationsweiche.
E1, E2 Informationen,
E0 Steuersignal.

B

59

- Entwickeln Sie nach der Schaltung in Bild B 59.1 den aufgelösten FUP.
- Der FUP ist mit Hilfe des Ersten de Morganschen Theorems umzuformen.
- Bilden Sie den FUP b) durch einen KOP nach.
- Für die Lösung b) ist das Programm aufzustellen.
- Testen Sie Ihr Programm durch experimentelle Aufnahme der Funktionstabelle B 59.1.

Tabelle B 59.1: Funktionstabellen zur Aufgabe B 58.2

E2	E1	E0	A0
0	0	0	
0	1	0	
1	0	0	
1	1	0	

E2	E1	E0	A0
0	0	1	
0	1	1	
1	0	1	
1	1	1	

Die Lösung dieser Aufgabe finden Sie auf den Seiten F29 und F30.

Die Beispiele und Aufgaben haben Ihnen gezeigt, daß man die in NAND oder NOR ausgeführten Schaltungen umformen kann. Die umgeformten Schaltungen lassen sich oft einfacher und übersichtlicher programmieren. Vielleicht fragen Sie sich, weshalb wir diesen Lehrstoff so ausführlich behandelt haben. Dazu folgende Bemerkungen:

Es wird sicher selten vorkommen, daß man zur Lösung einer Aufgabe die erforderlichen Verknüpfungen in NAND oder NOR programmiert. Dagegen wird es oft vorkommen, daß man eine in NAND oder NOR vorgegebene Hardwarelösung durch ein **Programm** nachbilden muß. Zu diesem Zweck müssen Sie die vorgegebene Schaltung analysieren können. Erst anschließend kann das Programm aufgestellt werden. Die Beispiele und auch Aufgaben wurden nach diesen Gesichtspunkten ausgewählt. Zum Abschluß wollen wir noch einen Drehschalter durch ein Programm nachbilden. Nach dem Durcharbeiten des Lehrstoffes sollten Sie in der Lage sein, selbständig derartige Aufgaben zu lösen.

Beispiel B 60.1

Drehschalter

Wir wollen ein Programm zur Nachbildung der Funktion eines Drehschalters aufstellen. Es werden ankommende Signale ausgewählt und auf eine gemeinsame Übertragungsleitung geschaltet.

Die Funktion der Schaltung in Bild B 60.1 entspricht der eines Drehschalters. Mit den binären Zuständen der Steuervariablen E5 und E6 wird die Schaltstellung ausgewählt. Die jeweilige Signalkombination von E5 und E6 bestimmt, welche Eingangsinformation (E1 bis E4) auf den Ausgang A0 geschaltet ist. Es besteht die Möglichkeit, wahlweise jeden der vier Eingänge mit der Ausgangsleitung zu verbinden. Die Steuersignale bereiten die Verknüpfungen zur Signalweitergabe vor. Anhand von Bild B 60.1 können Sie die erläuterte Funktion überprüfen.

Für die dargestellte Schaltung gilt folgende Zuordnung:

E6	E5	A0
0	0	E4
1	0	E3
0	1	E2
1	1	E1

- Zeichnen Sie einen der Schaltung entsprechenden, aufgelösten FUP auf.
- Der FUP ist mit dem de Morganschen Theorem $\overline{E1 \wedge E5 \wedge E6} = \overline{E1} \vee \overline{E5} \vee \overline{E6}$ umzuformen.
- Stellen Sie für den umgeformten FUP das Programm auf.
- Zum Programmtest sind einige ausgewählte Betriebszustände nach Tabelle B 61.1 experimentell zu überprüfen.

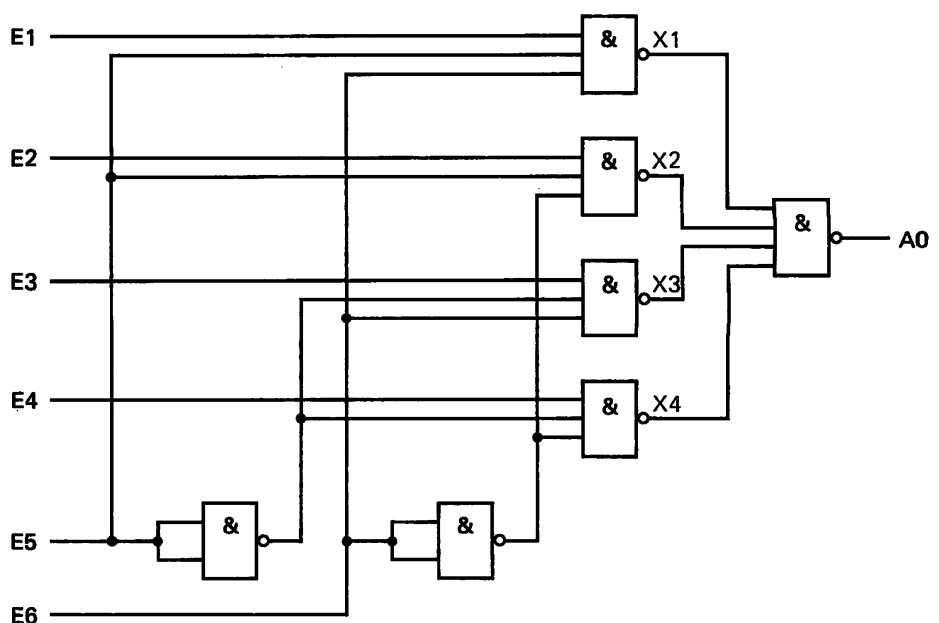


Bild B 60.1
Nachbildung der Funktion eines
Drehschalters mit vier Eingängen.
E1 bis E4 Informationen,
E5, E6 Steuersignale.

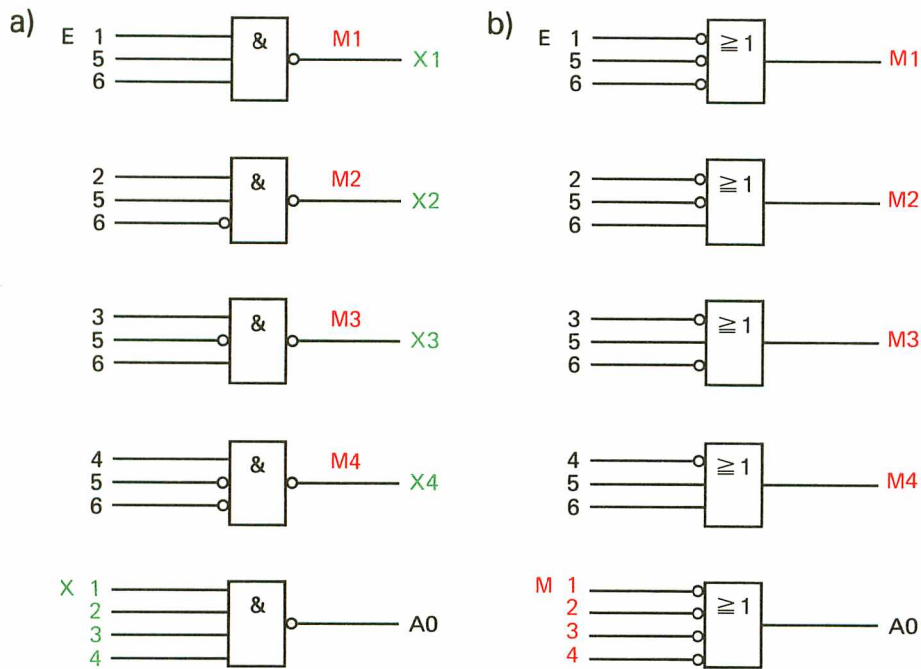


Bild B 61.1

a) Funktionsplan des Drehschalters, Beispiel B 60.1, Punkt a).

b) Umgeformter Funktionsplan des Drehschalters, Beispiel B 60.1, Punkt b).

Tabelle B 61.1: Funktionstabelle zum Programmtest

E4	E3	E2	E1	E6	E5	A0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	0	1	1	1	1
1	0	1	0	0	0	1
1	0	1	0	0	1	1
1	0	1	0	1	0	0
1	0	1	0	1	1	0

Lösung:

- a) Bild B 61.1a gibt den der Schaltung entsprechenden, aufgelösten FUP wieder.
- b) Den umgeformten FUP zeigt das Bild B 61.1b.
- c) Verwendet sind die in Bild B 61.1b eingetragenen Operanden. Aus dem FUP b) läßt sich folgendes Programm ableiten:

```

!NE1/NE5/NE6=M1
!NE2/NE5/E6=M2
!NE3/E5/NE6=M3
!NE4/E5/E6=M4
!NM1/NM2/NM3/NM4=A0
!PE

```

- d) Der Programmtest ergibt die in Tabelle B61.1 eingetragenen Werte für A0. Diese stimmen mit den Vorgaben überein.

Zusammenfassung

Die rechnerische Umformung der Verknüpfungsfunktionen UND, ODER und NICHT in NAND bzw. NOR erfolgt mit den de Morganschen Theoremen. Diese Gesetzmäßigkeit gilt für Verknüpfungen mit zwei und mehr Eingangsvariablen.

Wenn bei ungleichen Zuständen der Eingangsvariablen als Verknüpfungsergebnis die Ausgangsvariable den Zustand 1 annimmt, wird die Verknüpfung als Antivalenz bezeichnet. Wenn bei gleichen Zuständen der Eingangsvariablen die Ausgangsvariable den Zustand 1 annimmt, wird die Verknüpfung als Äquivalenz bezeichnet.

Bei einer Schwellenwertschaltung nimmt die Ausgangsvariable nur dann den Zustand 1 an, wenn mindestens eine vorgegebene Anzahl der Eingangsvariablen den Zustand 1 hat.



Zahlensysteme

Es gibt Bereiche, in denen die Benutzung des herkömmlichen Dezimalsystems sehr unzuweckmäßig und aufwendig oder aus technischen Gründen gar nicht möglich ist. Hierzu zählen die Digitaltechnik und deren Anwendungsbereiche, zu dem ja auch die Steuerungstechnik gehört.

Das im täglichen Leben benutzte Zahlensystem hat die Basis $B=10$. Die Dezimalzahl $Z=290$ ist z.B. die abgekürzte Schreibweise für:

$$Z = 200 + 90 + 0$$

$$Z = 2 \cdot 100 + 9 \cdot 10 + 0 \cdot 1$$

$$Z = 2 \cdot 10^2 + 9 \cdot 10^1 + 0 \cdot 10^0$$

Der Anteil, den eine Ziffer zum Zahlenwert Z beiträgt, hängt von der Stellung der Ziffer innerhalb der Zahl ab. Jede Stelle hat eine bestimmte **Wertigkeit**, einen bestimmten Stellenwert. Er wird durch die Potenz zur Basis $B=10$ ausgedrückt. Die Ziffer an jeder Stelle gibt an, wievielfach der Stellenwert zur Bildung des Zahlenwerts beiträgt. Das Dezimalsystem hat die Ziffern 0 bis 9. Zur direkten Nachbildung einer Dezimalziffer wären somit 10 verschiedene elektrische Zustände erforderlich.

In der Digitaltechnik wird deshalb mit einem Zahlensystem gearbeitet, das mit zwei elektrischen Zuständen auskommt. Diese Zustände lassen sich leicht und betriebssicher durch die binären Werte 0 und 1 nachbilden. Ein hierfür geeignetes Zahlensystem ist das **Dualsystem**.

Dualsystem

Dieses Zahlensystem hat die Basis $B=2$, die Stellenwerte sind eine Potenz zur Basis 2. Im Dualsystem gibt es nur die Ziffern 0 und 1. Wir kommen in unseren Anwendungen mit positiven ganzen Zahlen aus. Es gilt:

$$\text{Stellenwert} \quad \dots \quad 2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

dezimaler

$$\text{Stellenwert} \quad \dots \quad 128 \quad 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$$

Als Beispiel soll die Dezimalzahl 22 in eine Dualzahl umgesetzt werden. Zu diesem Zweck sind die dezimalen Stellenwerte im Dualsystem auszuwählen, deren Summe den Dezimalwert 22 ergibt. Ziffer 1 gibt an, daß der Stellenwert zur Bildung der Dezimalzahl benötigt wird. Ziffer 0 bezeichnet, daß er nicht benötigt wird. Wir erhalten:

$$1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 22$$

$$1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 22$$

$$16 + 0 + 4 + 2 + 0 = 22$$

In einfacher Schreibweise werden nur noch die Ziffern 0 und 1 als Dualzahl angegeben.

$$10110 = 22$$

↑
niedrigster Stellenwert

Ganz rechts steht die Dualziffer mit dem niedrigsten Stellenwert, sie beträgt in unserem Fall immer $2^0 = 1$. Ihr folgen die dezimalen Stellenwerte (nach links) in der Folge 2 4 8... . Zur Vermeidung von Verwechslungen erhalten die Zahlen einen **Index**.

Index 2 bedeutet: Dualzahl

Index 10 bedeutet: Dezimalzahl

$$22_{10} = 10110_2$$

Umgekehrt kann nach dem gleichen Schema eine Dualzahl in eine Dezimalzahl umgesetzt werden. Man geht wie folgt vor:

$$110101_2 = (32 + 16 + 0 + 4 + 0 + 1)_{10} \\ = 53_{10}$$

Die Stellen der Dualzahl mit dem Wert 1 sind durch ihre dezimalen Wertigkeiten zu ersetzen. Die Summe dieser Wertigkeiten ergibt die Dezimalzahl.

Selbstverständlich gibt es zur Umwandlung von einem Zahlensystem in das andere auch mathematische Regeln. Für unsere Anwendungen in der Steuerungstechnik sind die vorstehend behandelten Zusammenhänge jedoch ausreichend.

Aufgabe B 64.1

- Stellen Sie den Dezimalwert $Z_{10} = 43$ als Dualzahl dar.
- Wandeln Sie die Dualzahl $Z_2 = 01110$ in eine Dezimalzahl um.

Die Lösung dieser Aufgabe finden Sie auf Seite F 30.

Arithmetik im Dualsystem

Keine Angst, Sie müssen keine große Abhandlung über Arithmetik mit Dualzahlen über sich ergehen lassen. Für unser einfaches Steuerungssystem benötigen wir nur die Grundregeln der Addition und der Subtraktion. Diese Regeln sind sehr einfach, sie entsprechen den Regeln des Dezimalsystems.

Regeln für die Addition von Dualzahlen

Zu addieren sind die beiden einstelligen Dualzahlen A und B. Das Ergebnis der Addition ist eine **Summe**. Sind A und B gleich 1, dann ergibt sich ein Übertrag Ü für die nächsthöhere Stelle. Es gelten folgende Regeln:

$A + B = S$	\ddot{U}
$0 + 0 = 0$	0
$1 + 0 = 1$	0
$0 + 1 = 1$	0
$1 + 1 = 0$	1

Das Additionszeichen „+“ ist als „PLUS“ zu bezeichnen, damit keine Verwechslung mit logisch UND entsteht.

Für den Ausdruck „Übertrag“ in die nächsthöhere Stelle finden Sie in anderer Fachliteratur auch die Bezeichnung „Carry“.

Wir wenden nun die genannten Regeln auf die Addition mehrstelliger Dualzahlen an.

1. Beispiel:

Addition: $A = 7_{10} = 111_2$

$B = 6_{10} = 110_2$

$$\begin{array}{r}
 A \quad 1 \ 1 \ 1 \\
 + B \quad 1 \ 1 \ 0 \\
 \hline
 \ddot{U} \ 1 \ 1 \ 0 \\
 \hline
 S \ 1 \ 1 \ 0 \ 1_2 = 13_{10}
 \end{array}$$

Im Beispiel ist zu $A+B$ das Übertragungssignal \ddot{U} zu addieren. In der Spalte ganz rechts ist noch kein Übertrag vorhanden. Zur Kontrolle wandeln wir das Ergebnis in eine Dezimalzahl um und erhalten:

$$8+4+0+1=13.$$

2. Beispiel:

Addition: $A = 15_{10} = 1111_2$

$B = 17_{10} = 10001_2$

$$\begin{array}{r}
 A \quad 1 \ 1 \ 1 \ 1 \\
 + B \quad 1 \ 0 \ 0 \ 0 \ 1 \\
 \hline
 \ddot{U} \ 1 \ 1 \ 1 \ 1 \ 1 \\
 \hline
 S \ 1 \ 0 \ 0 \ 0 \ 0 \ 0_2 = 32_{10}
 \end{array}$$

Wie man bei der dualen Addition vorgeht, haben Sie sicher aus den Beispielen erkannt. Wir benötigen nunmehr ein Programm, das uns die Rechenarbeit abnimmt.

Nach den Regeln der Digitaltechnik stellen wir zuerst eine Funktionstabelle auf und entwickeln danach die Schaltfunktionen, die anschließend programmiert werden.

Halbaddierer

Unter einem Halbaddierer versteht man eine Schaltung bzw. ein Programm, das nur die Addition $A+B=S$ ausführt, den Übertrag \dot{U} der vorangehenden Addition aber nicht berücksichtigt. In den eben gezeigten Beispielen könnte der Halbaddierer also nur die Addition der Spalte ganz rechts ausführen.

Die Signale werden den Variablen der SPS wie folgt zugeordnet:

$$A \triangleq E1 \quad B \triangleq E2 \quad S \triangleq A1 \quad \dot{U} \triangleq A2$$

Tabelle B 66.1: Funktionstabelle des Halbaddierers

E2	E1	A1	A2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Danach erhalten wir die folgenden Schaltfunktionen

$$S = A1 = E1\bar{E2} \vee \bar{E1}E2$$

$$\dot{U} = A2 = E1E2$$

Den Funktionsplan und den Kontaktplan sehen Sie im Bild B 66.1. Für unsere SPS lauten die Anweisungen:

$$!E1\&NE2/NE1\&E2=A1$$

$$!E1\&E2=A2$$

$$!PE$$

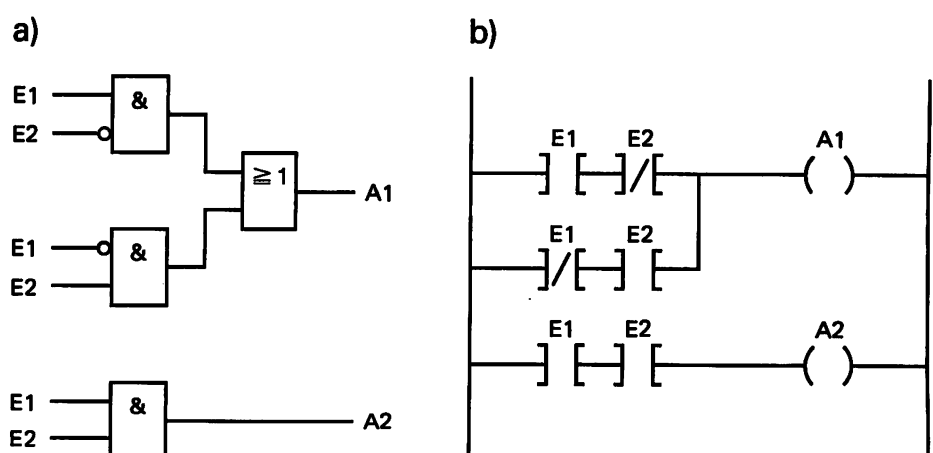


Bild B 66.1
Halbaddierer.
a) Funktionsplan,
b) Kontaktplan.

Aufgabe B 66.1

Geben Sie das eben gezeigte Programm in die SPS ein. Danach ist experimentell die Funktionstabelle B 66.1 zu überprüfen.

Die Lösung dieser Aufgabe finden Sie auf Seite F 30.

Volladdierer

Mit dem Halbaddierer kann nur die Addition der rechten Spalte in der ersten Dualstelle ausgeführt werden. Bereits bei der zweiten Dualstelle muß der Übertrag aus der vorangehenden Dualstelle berücksichtigt werden. Das Programm für die zweite Dualstelle hat somit drei Eingangsvariablen, nämlich A, B und \bar{U}_x . Wir vereinbaren, daß \bar{U}_x der Übertrag aus der Addition der vorangehenden Spalte bzw. Dualstelle ist. Eine Schaltung oder ein Programm zur Lösung dieser Aufgabe bezeichnet man als **Volladdierer**.

Die bereits festgelegten Signalbezeichnungen gelten weiter; zusätzlich wird noch der Übertrag \bar{U}_x durch die Variable E3 nachgebildet. Für den Volladdierer gilt die Funktionstabelle B67.1. Danach können die Schaltfunktionen für Summe und Übertrag aufgestellt werden.

Tabelle B67.1: Funktionstabelle des Volladdierers

E3	E2	E1	A1	A2
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

Die Schaltfunktion für die Summe lautet:

$$A1 = E1\bar{E}2\bar{E}3 \vee \bar{E}1E2\bar{E}3 \vee \bar{E}1\bar{E}2E3 \vee E1E2E3$$

Die Schaltfunktion für den Übertrag auf die nächste Dualstelle lautet:

$$A2 = \underline{E1E2\bar{E}3} \vee \underline{E1\bar{E}2E3} \vee \underline{\bar{E}1E2E3} \vee \underline{E1E2E3}$$

Mit den bis jetzt besprochenen Regeln ist für A2 folgende Vereinfachung möglich:

$$A2 = E1E3 \vee E2E3 \vee E1E2$$

Aufgabe B67.1

- Stellen Sie für den Volladdierer das Programm mit vorstehenden Operanden auf.
- Überprüfen Sie bitte experimentell, ob das Programm die Funktionen aus Tabelle B67.1 erfüllt.

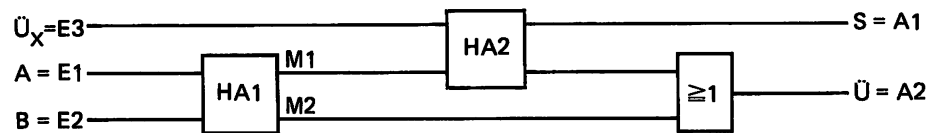
Die Lösung dieser Aufgabe finden Sie auf Seite F30.

B**68**

Bild B 68.1
Volladdierer, bestehend aus zwei
Halbaddierern (Aufgabe B 68.1).

Aufgabe B 68.1

In der Schaltungstechnik wird vielfach ein Volladdierer aus zwei Halbaddierern nach dem Schema in Bild B 68.1 aufgebaut. In unserer Aufgabe ordnen wir M1 dem S-Ausgang, M2 dem Ü-Ausgang zu.



- Programmieren Sie den Volladdierer nach dem Schema in Bild B 68.1. Benutzen Sie das bereits für den Halbaddierer aufgestellte Programm und die in das Schema eingetragenen Operanden.
- Das Programm ist wie üblich zu testen. Vergleichen Sie die experimentellen Ergebnisse mit der Funktionstabelle B 67.1.

Das Ergebnis dieser Aufgabe finden Sie auf Seite F 30.

Regeln für die Subtraktion von Dualzahlen

Das Ergebnis der Subtraktion der beiden einstelligen Dualzahlen $A-B$ ist eine Differenz D . Bei $B > A$ wird eine Entleihung E aus der nächsthöheren Stelle erforderlich. Das ist der Fall bei: $A-B=0-1$. Es gelten die folgenden Regeln:

$A - B = D$	E
$0 - 0 = 0$	0
$1 - 0 = 1$	0
$0 - 1 = 1$	1
$1 - 1 = 0$	0

A Minuend
B Subtrahend
D Differenz
E Entleihung

Wir wenden vorstehende Regeln wieder auf die Subtraktion mehrstelliger Dualzahlen an. Aufgeführt sind nur Beispiele für positive Differenzen (Zahlenwert $A > \text{Zahlenwert } B$).

1. Beispiel:

Subtrahieren Sie die Zahl $B = 3_{10} = 11_2$ von der Zahl $A = 9_{10} = 1001_2$.

$$\begin{array}{r}
 A \quad 1 \ 0 \ 0 \ 1 \\
 - B \quad \quad 1 \ 1 \\
 \hline
 E \quad 1 \ 1 \\
 \hline
 D \quad 0 \ 1 \ 1 \ 0_2 = 6_{10}
 \end{array}$$

Zur Kontrolle bilden wir die Summe der dezimalen Wertigkeiten; sie ergibt $0+4+2+0=6$ als Differenz. Das Subtraktionsergebnis ist richtig.

Das Beispiel zeigt:

Falls eine Entleihung erforderlich ist, muß sie in der folgenden, nächsthöheren Stelle berücksichtigt werden, sie muß also dort ebenfalls subtrahiert werden: $A-B-E$.

2. Beispiel:

Subtrahieren Sie die Zahl $B = 23_{10} = 10111_2$ von der Zahl $A = 30_{10} = 11110_2$.

$$\begin{array}{r}
 A \quad 1 \ 1 \ 1 \ 1 \ 0 \\
 - B \quad 1 \ 0 \ 1 \ 1 \ 1 \\
 \hline
 E \quad 0 \ 1 \ 1 \ 1 \\
 \hline
 D \quad 0 \ 0 \ 1 \ 1 \ 1_2 = 7_{10}
 \end{array}$$

Auch diese Rechenarbeit soll die SPS für uns erledigen – wir müssen wieder ein Rechenprogramm aufstellen. Es wird der gleiche Weg wie bei der Programmierung der Addition beschritten.

Halbsubtrahierer

Diese Schaltung bzw. das Programm kann nur die Subtraktion $A-B=D, E$ ausführen; es kann die Entleihung aus der vorangegangenen Subtraktion noch nicht berücksichtigen.

In den vorstehenden Beispielen könnte damit nur die Subtraktion der rechten Spalte ausgeführt werden. Es gelten folgende Vereinbarungen:

$$A \triangleq E1 \quad B \triangleq E2 \quad D \triangleq A1 \quad E \triangleq A2$$

Die Beachtung dieser Zuordnungen ist sehr wichtig, denn die Rechenoperation lautet $E1-E2=A1, A2$. $E2-E1$ würde ein falsches Ergebnis liefern.

Tabelle B69.1: Funktionstabelle des Halbsubtrahierers

E2	E1	A1	A2
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

Danach bilden folgende Schaltfunktionen den Halbsubtrahierer nach:

$$D = A1 = E1\overline{E2} \vee \overline{E1}E2 \quad E = A2 = \overline{E1}E2$$

Aufgabe B70.1

Mit den eben gezeigten Schaltfunktionen ist

- das Programm für den Halbsubtrahierer aufzustellen und
- mit Ihrer SPS experimentell zu überprüfen.

Die Lösung dieser Aufgabe finden Sie auf Seite F31.

Vollsubtrahierer

Der Halbsubtrahierer kann eine Entleihung aus der vorhergehenden Stelle nicht berücksichtigen. Dazu ist ein Vollsubtrahierer erforderlich. Die Entleihung wird mit E_x bezeichnet und durch die Variable E_3 nachgebildet. Das Programm muß das Ergebnis der Rechenoperation $A-B-E_x=D, E$ ausführen. Die auszuführende Signalverknüpfung enthält folgende Funktionstabelle:

Tabelle B70.1: Funktionstabelle des Vollsubtrahierers

E3	E2	E1	A1	A2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Mit der Funktionstabelle sind die Eigenschaften des zu entwickelnden Programms bzw. der zu entwerfenden Schaltung wieder eindeutig beschrieben.

Aufgabe B70.2

Programmierung des Vollsubtrahierers

- Nach der Funktionstabelle B70.1 sind die Schaltfunktionen des Vollsubtrahierers aufzustellen. Versuchen Sie, die Schaltfunktion für die Entlehnung E , dargestellt durch die Ausgangsvariable A_2 , zu vereinfachen.
- Programmieren Sie den Vollsubtrahierer.
- Das Programm b) ist wie üblich experimentell zu testen.

Die Lösung finden Sie auf Seite F31.

Damit schließen wir die einführende Betrachtung des Abschnitts „Arithmetik mit Dualzahlen“ ab.



Codierung

Der Z-80 Mikroprozessor in unserer SPS-Anlage ist ein zahlenverarbeitendes System. Seine Befehle erhält er in Form von Zahlen. Etwas anderes als Zahlen kann er nicht verstehen und verarbeiten. Natürlich versteht der Mikroprozessor keine gesprochenen Wörter (noch nicht!); die Zahlen müssen ihm in codierter Form mitgeteilt werden.

Wir wollen uns in diesem Kapitel mit verschiedenen **Codierungen** von Zahlen befassen. Dabei muß ausdrücklich gesagt werden, daß die Darstellung von Zahlen durch Ziffern auch nichts anderes als eine Codierung ist.

In unserer speicherprogrammierbaren Steuerung findet eine Umsetzung von Nachrichten statt. Jede Anweisung stellt eine Nachricht dar. Wir formulieren die Anweisungen beispielsweise im **Mnemonic-Code** oder als **mathematische Funktionen**. Die SPS versteht aber nur Codeworte mit Binärzeichen. Sie versteht unsere Anweisungen nur, wenn zwischen der „Programmiersprache“ und der „Maschinensprache“ eine feste Zuordnung besteht.

Denken Sie weiterhin an die Zuordnung der Dezimalzahlen zu den Dualzahlen. Der Ausdruck $5_{10} = 0101_2$ sagt, bezogen auf vorstehende Definition, folgendes aus: Die Ziffer 5 gehört zum Zeichenvorrat der Dezimalzahlen und die Kombination 0101 zum Zeichenvorrat der Dualzahlen. Nach dem vereinbarten Code sind beide Zeichen eindeutig einander zugeordnet.

Unter Codierung versteht man die **Verschlüsselung** einer Nachricht. Die Verschlüsselung muß nach einer bestimmten Vorschrift erfolgen.

Ein Code ist eine Vorschrift für die eindeutige Zuordnung der Zeichen eines Zeichenvorrats zu denjenigen eines anderen Zeichenvorrats nach DIN 44 300.

Die Codierung hat für die gesamte elektronische Steuerungstechnik große Bedeutung. Mit ihr kann man Steuerungsanweisungen und Befehle von SPS und VPS in eine Form bringen, in der sie sich gut verarbeiten lassen. Die in der Digitaltechnik und für SPS verwendeten Codes ordnen den einzelnen Nachrichten Codezeichen zu, die aus mehreren binären Elementen bestehen. Ein derartiges zweiwertiges Element ist ein **Bit** (binary digit). Codes mit zweiwertigen Codeelementen werden in der Digitaltechnik als **Binärcodes** bezeichnet. Die in der Steuerungstechnik verwendeten Codezeichen, die auch als **Codeworte** bezeichnet werden, bestehen aus einer bestimmten Anzahl von Binärzeichen. Mit einem n-stelligen Codewort lassen sich 2^n verschiedene Nachrichten (Informationen, Anweisungen) darstellen. Die für ein Mikroprozessorsystem verwendeten Codewörter haben eine Länge von 8 Bit. Sie werden als **Bytes** bezeichnet.

Binärcodes für Dezimalzahlen

Zum Verständnis des Lehrstoffes genügt die Besprechung der Verschlüsselung von Dezimalzahlen. Buchstaben und Zeichen klammern wir aus. Im Prinzip haben für deren Codierung die nachstehenden Ausführungen ebenfalls Gültigkeit.

B
72

Additive Codes

Bei diesen Codes hat jedes Binärzeichen innerhalb des Codeworts einen festen dezimalen Stellenwert. Die verschlüsselte Dezimalzahl ergibt sich aus der Summe der Wertigkeiten der Stellen, deren binäre Elemente den Zustand 1 haben.

In der Tabelle B72.1 sind einige häufig verwendete Codes aufgeführt. Die Spalten 1 bis 3 enthalten additiv bewertete Codes. Nicht additiv bewertete Codes sind Anordnungs-codes. Die Spalten 4 und 5 in Tabelle B72.1 enthalten dafür je ein Beispiel.

Tabelle B72.1: Häufig verwendete Codes

	1	2	3	4	5
Code	Dual-Code	BCD-Code	Aiken-Code	Exzeß-3-Code	Gray-Code
Stellenwert	8 4 2 1	8 4 2 1	2 4 2 1	keine	keine
Codierung von Dezimalzahlen	0 0000 1 0001 2 0010 3 0011 4 0100 5 0101 6 0110 7 0111 8 1000 9 1001 10 1010 11 1011 12 1100 13 1101 14 1110 15 1111	0 0000 1 0001 2 0010 3 0011 4 0100 5 0101 6 0110 7 0111 8 1000 9 1001 Pseudo-tetraden	0 0000 1 0001 2 0010 3 0011 4 0100 Pseudo-tetraden 5 1011 6 1100 7 1101 8 1110 9 1111	0 Pseudo-tetraden 1 0011 2 0100 3 0101 4 0110 5 0111 6 1000 7 1001 8 1010 9 1011 Pseudo-tetraden	0 0000 1 0001 2 0011 3 0010 4 0110 5 0111 6 0101 7 0100 8 1100 9 1101 10 1111 11 1110 12 1010 13 1011 14 1001 15 1000
	E4E3E2E1	E4E3E2E1	E4E3E2E1	E4E3E2E1	E4E3E2E1

Minimalcodes und Maximalcodes

Minimalcodes enthalten nur soviel Binärzeichen je Codewort, wie zur Darstellung der Information unbedingt erforderlich sind. Zur Codierung der 10 Dezimalziffern ist ein **Codewort** mit 4 Bit erforderlich. Die Tabelle B72.1 enthält somit nur Minimalcodes.

Maximalcodes verwenden im Codewort mehr Binärzeichen als zur Darstellung der Information erforderlich sind. Derartige Codes sind redundant. Mit **Redundanz R** wird ausgedrückt, daß die Zahl der binären Elemente je Codewort für eine bestimmte Information größer als beim Minimalcode ist.

Die in Tabelle B72.1 gezeigten Codes haben vier binäre Elemente je Codewort. Vier Binärzeichen bilden eine **Tetrade**. Zur Codierung der Dezimalziffern werden von den 16 möglichen Signalkombinationen nur 10 Kombinationen benötigt. Die überflüssigen 6 Kombinationen heißen **Pseudotetraden** oder **Pseudodezimale**.

Einschrittige Codes

Verfolgen Sie bitte die Entwicklung der Codeworte des Gray-Codes in Spalte 5 von Tabelle B72.1. Beim Übergang von einer Dezimalzahl zur folgenden oder vorhergehenden ändert immer nur ein Bit seinen Zustand. Diese Codierung findet in der Steuerungstechnik zur Umwandlung analoger Signale in digitale Signale Anwendung. Sie wird in digital arbeitenden Drehgebern und Längenmeßgeräten benutzt. Mit einschrittigen Codes ergeben sich im Vergleich zu anderen Codes nur sehr kleine Umsetzfehler.

Fehlererkennungs-Codes

Wir haben bereits erwähnt, daß im Codewort Binärzeichen verfälscht werden können. Oft entstehen die Fehler beim Übertragen oder Speichern der Informationen. Ohne besondere Maßnahmen lassen sich derartige Fehler nur erkennen, wenn das neu entstandene Zeichen nicht zum Zeichenvorrat des Codes gehört.

Bei der Verschlüsselung mit dem Minimalcode wird oft folgende Methode zur Fehlererkennung benutzt. Durch Hinzufügen eines Prüfbits oder **Paritätsbits** wird der Minimalcode zum Fehlererkennungscode erweitert. Betrachten Sie bitte die Tabelle B74.1. Dargestellt ist der BCD-Code. Den Binärzeichen des Codes ist ein Prüfbit beigelegt. Das Beispiel wurde so gewählt, daß die Kombination von Codewort und Prüfbit eine ungerade Anzahl von Elementen im 1-Zustand enthält. Jedes übertragene Zeichen wird auf diesen Zustand überprüft. Ergibt die Überprüfung ein geradzahliges Ergebnis, wird eine Fehlermeldung ausgelöst.

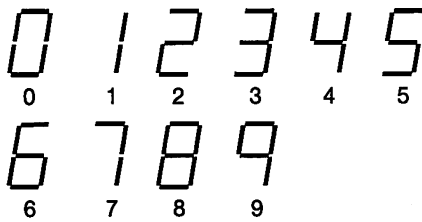
Tabelle B74.1: BCD-Code mit Prüfbit

Dezimal- ziffer	Codewort E4E3E2E1 8 4 2 1	Prüf- bit	Elemente mit Zustand 1
0	0 0 0 0	1	1
1	0 0 0 1	0	1
2	0 0 1 0	0	1
3	0 0 1 1	1	3
4	0 1 0 0	0	1
5	0 1 0 1	1	3
6	0 1 1 0	1	3
7	0 1 1 1	0	3
8	1 0 0 0	0	1
9	1 0 0 1	1	3

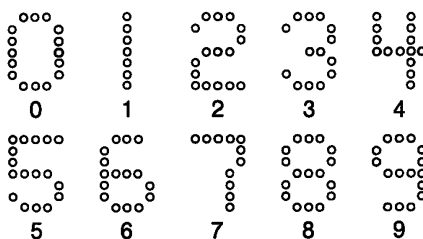
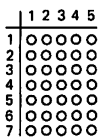
B

a)

74



b)



c)

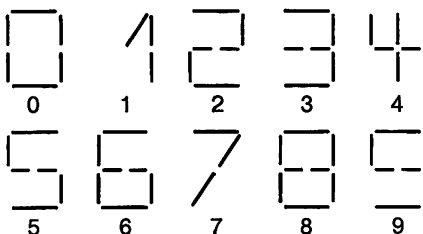
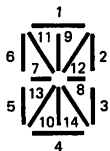


Bild B74.1

Halbleiter-Anzeigen:

- a) Sieben-Segment-Anzeige,
b) 5x7 Punkt-Matrix-Anzeige,
c) Alphanumerische Anzeige.

Alphanumerische Codes

Die folgenden Informationen sollen das Kapitel „Codierung“ abrunden. Codes, deren Zeichenvorrat mindestens die Dezimalziffern und die Buchstaben des Alphabets umfassen, nennt man **alphanumerische Codes**. Dazu gehören u. a. der Morse-Code und das internationale Fernschreibalphabet.

Für die Datenverarbeitung, zu der ja auch die SPS gehört, hat die Verschlüsselung nach ASCII (American Standard Code for Information Interchange = amerikanischer Normcode für Datenaustausch) große Bedeutung. Dieser Code ist bei uns unter dem Namen ISO-7-Bit-Code bekannt und nach DIN 66 003 genormt (ISO = Abk. für International Standards Organisation).

Nach soviel Theorie wird es wieder Zeit, deren technische Bedeutung und Anwendung zu behandeln. Schaltungen sowie Programme zur Codierung, Umcodierung und Decodierung fallen unter den Begriff **Codeumsetzer**.

Codeumsetzer

Zur Codierung der Nachrichten ist ein **Codeverschlüssler** erforderlich. Sie finden dafür auch die Bezeichnungen Encoder oder Codierer. Eine Anwendung kennen Sie schon, es ist die Tastatur mit zugehöriger Elektronik unserer SPS. Mit einem Schaltnetz, dem Codierer, erfolgt die Umsetzung der eingegebenen Zeichen in die binäre Darstellung. Das Steuerwerk der SPS versteht nur Codeworte mit binären Elementen.

Das Bild B75.1 zeigt eine einfache Codierschaltung zum Umsetzen von Dezimalziffern in einen tetradischen Code. Wir werden diese Schaltungen gleich in einer Aufgabe besprechen.

Es gibt Steuerungen, die mit verschiedenen Codes arbeiten. Manche Codes eignen sich gut für arithmetische Schaltungen (Rechenschaltungen), bei anderen ist die Fehlererkennbarkeit leichter. Schließlich können auch Eingabe- bzw. Ausgabegeräte für verschiedene Codes ausgelegt sein. Damit Anlagenteile und Geräte unterschiedlicher Codierung zusammenarbeiten können, sind **Codeumsetzer** erforderlich. Sie übernehmen die Funktion eines „Dolmetschers“. Das Bild B78.1 zeigt Ihnen ein Schaltnetz zur Umsetzung des Exzeß-3-Codes in den BCD-Code (8-4-2-1 Code). Auch hierauf wird in einer Aufgabe noch eingegangen.

Zur Rückumsetzung einer codierten Information wird ein Codeentschlüssler bzw. **Decoder** benötigt. Hier gruppieren wir auch die Schaltnetze zur Ansteuerung von Anzeigeeinheiten ein. Gebräuchlich sind die Sieben-Segment-, die Punkt-Matrix- und die alphanumerische Anzeige (siehe Bild B74.1). Zur Anzeige von Dezimalziffern wird in der Regel die Sieben-Segment-Anzeige benutzt.

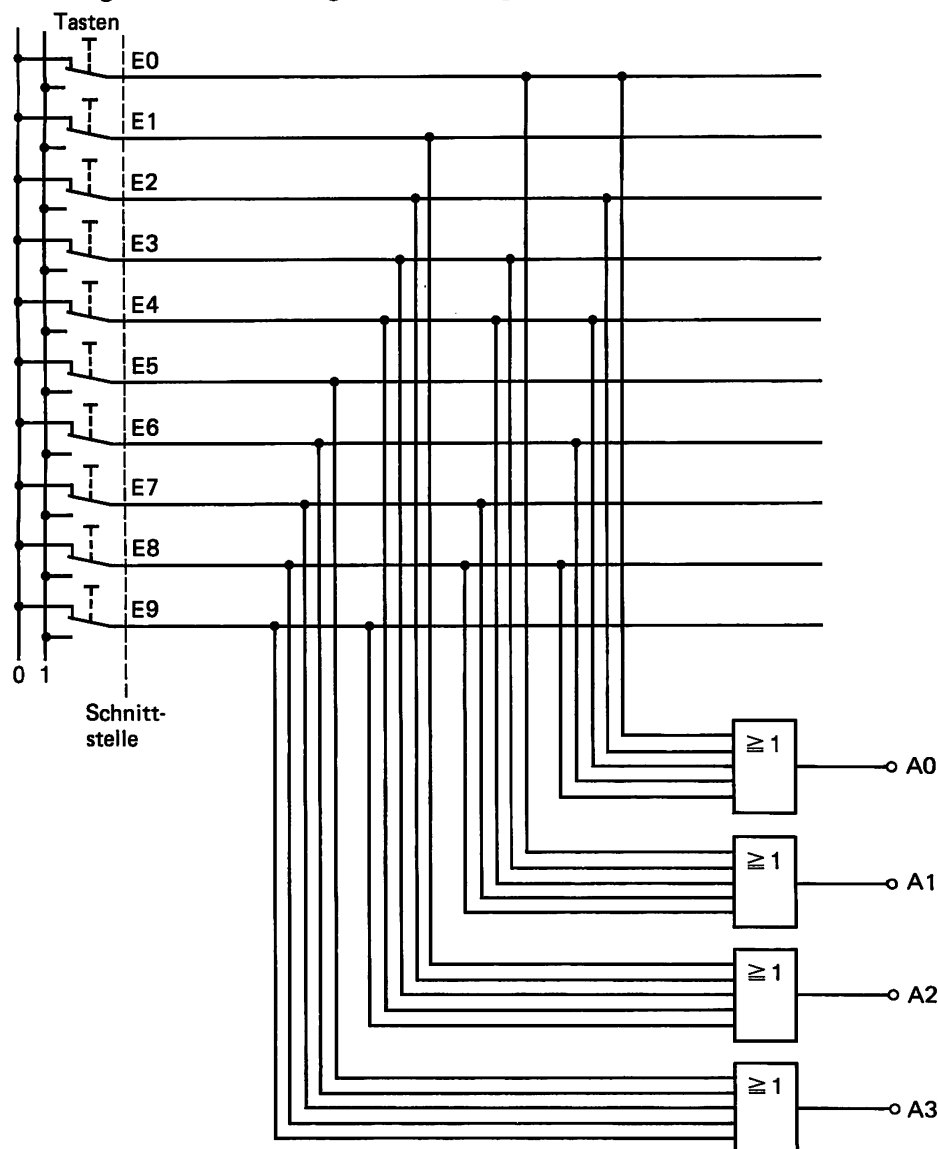


Bild B75.1
Codierschaltung zur Umsetzung von Dezimalziffern in einen tetradischen Code.

In der technischen Ausführung unterscheidet man zwischen LED- und LCD-Anzeige. **LED (Light-Emitting-Diode)** ist die Abkürzung für Leucht- oder Lumineszenzdiode. Es ist eine Halbleiterdiode, die bei Stromdurchgang Licht aussendet. Durch Dotierung mit besonderen Elementen kann man sichtbares Licht in den Farben Rot, Orange, Gelb, Grün oder Blau erzeugen. Bei der LED-Anzeige bestehen die Segmente aus Leuchtdioden.

Die **LCD-Anzeige (Liquid-Crystal-Display)** ist eine Flüssigkristallanzeige. Sie ist passiv, leuchtet also nicht von selbst. Zur Anzeige ist Tages- oder Kunstlicht erforderlich.

Codierer zur Umsetzung der Dezimalziffern in den Exzeß-3-Code

Das Bild B 75.1 zeigt das Schaltnetz des Codierers. Jeder Eingabetaste ist eine Dezimalziffer zugeteilt. In unserem Beispiel dürfen die Tasten immer nur nacheinander betätigt werden. Eine Logik verschlüsselt die eingegebene Dezimalziffer. An den Ausgängen A0 bis A3 erscheint als Codewort die **codierte Dezimalziffer**.

Beispiel B76.1

Programmierung eines Codierers zur Umsetzung der Dezimalziffern in den Exzeß-3-Code.

Für das Schaltnetz in Bild B 75.1 sind folgende Aufgaben durchzuführen:

- Stellen Sie bitte nach dem Muster von Tabelle B76.1 die Codetabelle auf.

Tabelle B76.1: Muster der Codetabelle zu Beispiel B76.1

Dezimal- ziffer	Taste	Codewort am Ausgang			
		A3	A2	A1	A0
0	E0				
1	E1				
.	..				
.	..				
.	..				
9	E9				

- Stellen Sie die Schaltfunktionen für die Ausgangsvariablen A0 bis A3 auf.
- Entwickeln und testen Sie mit der Schaltfunktion von b) das Programm. Überprüfen Sie, ob das Programm die ausgefüllte Codetabelle B76.1 erfüllt.

Lösung:

- a) Eine Codetabelle enthält immer die Zuordnung der Ausgangsvariablen zu den Eingangsvariablen. In unserem Beispiel ist das codierte Ausgangssignal einer Eingangstaste zuzuweisen. Ein betätigter Taster hat Signalzustand 1.

Tabelle B77.1: Funktionstabelle zu Beispiel B76.1

Dezimal- ziffer	Taste	Codewort am Ausgang			
		A3	A2	A1	A0
0	E0	0	0	1	1
1	E1	0	1	0	0
2	E2	0	1	0	1
3	E3	0	1	1	0
4	E4	0	1	1	1
5	E5	1	0	0	0
6	E6	1	0	0	1
7	E7	1	0	1	0
8	E8	1	0	1	1
9	E9	1	1	0	0

- b) Nach den Regeln der Digitaltechnik ist für jede Ausgangsvariable eine Schaltfunktion aufzustellen. Darin müssen die Eingangsvariablen enthalten sein, die zum Signalzustand 1 der Ausgangsvariablen führen.

$$A0 = E0 \vee E2 \vee E4 \vee E6 \vee E8$$

$$A1 = E0 \vee E3 \vee E4 \vee E7 \vee E8$$

$$A2 = E1 \vee E2 \vee E3 \vee E4 \vee E9$$

$$A3 = E5 \vee E6 \vee E7 \vee E8 \vee E9$$

- c) Die Schaltfunktionen lassen sich unmittelbar programmieren. Wir erhalten folgendes Programm:

```
!E0/E2/E4/E6/E8=A0
```

```
!E0/E3/E4/E7/E8=A1
```

```
!E1/E2/E3/E4/E9=A2
```

```
!E5/E6/E7/E8/E9=A3
```

```
!PE
```

Umcodierer

Wie wir bereits beschrieben haben, müssen oftmals Geräte zusammenarbeiten, die für verschiedene Codes ausgelegt sind. In diesem Fall sind die unterschiedlichen Zeichenkombinationen über einen **Codeumsetzer** an die Geräte anzupassen. Das Bild B78.1 zeigt ein Schaltnetz, das den Exzeß-3-Code in den BCD-Code übersetzt.

Der Code des Eingangssignals heißt **Quellcode**, der des Ausgangssignals wird als **Zielcode** bezeichnet.

Beispiel B78.1:**Programmierung eines Codeumsetzers**

Stellen Sie für den Codeumsetzer in Bild B78.1

- die Schaltfunktionen für die Ausgangsvariablen A1 bis A4 und
- nach den Schaltfunktionen das Programm auf.

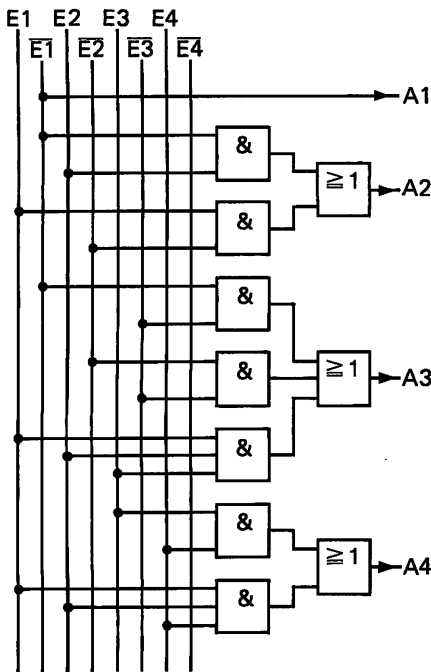
B**78****Lösung:**

Bild B78.1
Codeumsetzer vom Exzeß-3-Code
in den BCD-Code (8-4-2-1 Code).

- Aus dem Schaltnetz in Bild B78.1 können Sie unmittelbar folgende Schaltfunktionen ablesen:

$$A1 = \overline{E1}$$

$$A2 = \overline{E1}E2 \vee E1\overline{E2}$$

$$A3 = \overline{E1}\overline{E3} \vee \overline{E2}\overline{E3} \vee E1E2E3$$

$$A4 = E3E4 \vee E1E2E4$$

- Die Schaltfunktionen von a) ergeben folgendes Programm:

$$!NE1=A1$$

$$!NE1\&E2/E1\&NE2=A2$$

$$!NE1\&NE3/NE2\&NE3/E1\&E2\&E3=A3$$

$$!E3\&E4/E1\&E2\&E4=A4$$

$$!PE$$

Aufgabe B78.1

Überprüfen Sie das in Beispiel B78.1 aufgestellte Programm. Nehmen Sie experimentell die Codetabelle B78.1 auf. Vergleichen Sie die Ergebnisse (Zielcode) mit den Codeworten in Tabelle B72.1, Spalte 2.

Tabelle B78.1: Codetabelle des Umsetzers von Aufgabe B78.1

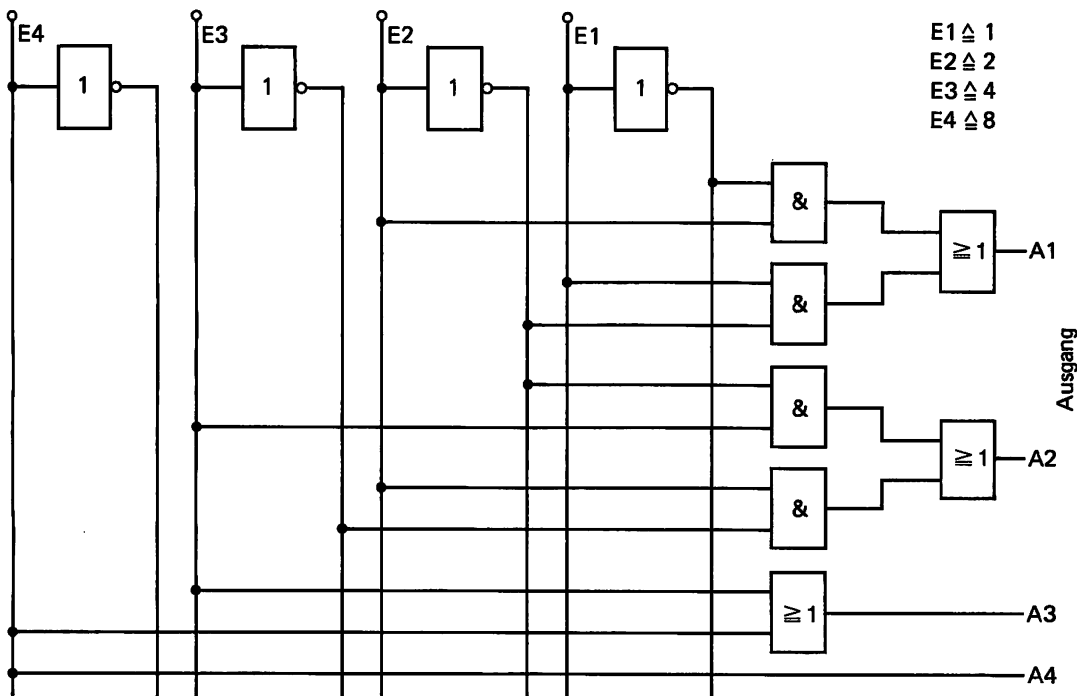
Dezimal- ziffer	Quellcode				Zielcode			
	E4	E3	E2	E1	A4	A3	A2	A1
0	0	0	1	1				
1	0	1	0	0				
2	0	1	0	1				
3	0	1	1	0				
4	0	1	1	1				
5	1	0	0	0				
6	1	0	0	1				
7	1	0	1	0				
8	1	0	1	1				
9	1	1	0	0				

Aufgabe B 79.1

Das Bild B 79.1 zeigt das Schaltnetz eines Codeumsetzers. Bekannt ist der Quellcode. Die Dezimalziffern sehen Sie eingangsseitig im BCD-Code (8-4-2-1). Der Zielcode (Code des Ausgangssignals) ist unbekannt.

- Stellen Sie mit Hilfe des Schaltnetzes in Bild B 79.1 die Schaltfunktionen für die Ausgangsvariablen auf.
- Mit Hilfe der Schaltfunktionen aus a) ist das Schaltnetz zu programmieren.

Eingänge: BCD-Code



- Nehmen Sie experimentell die Codetabelle B 79.1 auf. An Hand der Tabelle B 72.1 ist der Zielcode zu ermitteln.

Bild B 79.1
Codeumsetzer zur Aufgabe B 79.1
Quellcode: BCD-Code,
Zielcode unbekannt.

Tabelle B 79.1: Codetabelle zur Aufgabe B 79.1

Dezimal- ziffer	Quellcode				Zielcode			
	E4	E3	E2	E1	A4	A3	A2	A1
0	0	0	0	0				
1	0	0	0	1				
2	0	0	1	0				
3	0	0	1	1				
4	0	1	0	0				
5	0	1	0	1				
6	0	1	1	0				
7	0	1	1	1				
8	1	0	0	0				
9	1	0	0	1				

Die Lösung dieser Aufgaben finden Sie auf den Seiten F 52 und F 53.

Pseudotetradenkontrolle

Im folgenden Beispiel arbeitet ein Gerät (oder eine Anlage) mit Dezimalziffern im BCD-Code. Bei der Informationsverarbeitung dürfen keine Pseudotetraden auftreten. Pseudotetraden sind Vier-Bit-Muster, denen keine Dezimalziffern zugeordnet sind (siehe Tabelle B72.1).

Die codierten Dezimalziffern sind deshalb auf das Vorhandensein von Pseudotetraden zu überprüfen. Bei deren Auftreten soll eine Fehlermeldung ausgegeben werden.

Das Schaltnetz in Bild B 80.1 überprüft die Eingangstetraden (E1 bis E4) auf Pseudotetraden. Gültige Tetraden werden vom Eingang (E1 bis E4) auf den Ausgang (A1 bis A4) durchgeschaltet. Es gilt die Zuordnung:

$$A1 = E1 \quad A2 = E2 \quad A3 = E3 \quad A4 = E4$$

Beim Auftreten einer Pseudotetrade am Eingang sollen die Ausgänge A1 bis A4 den Zustand 0 annehmen.

Beispiel B 80.1

Programmieren und testen Sie das Schaltnetz von Bild B 80.1.

- Stellen Sie für die Ausgangsvariablen A1 bis A4 die Schaltfunktionen auf.
- Nach den Schaltfunktionen ist der KOP zu entwickeln.
- Setzen Sie den KOP in ein Programm um.

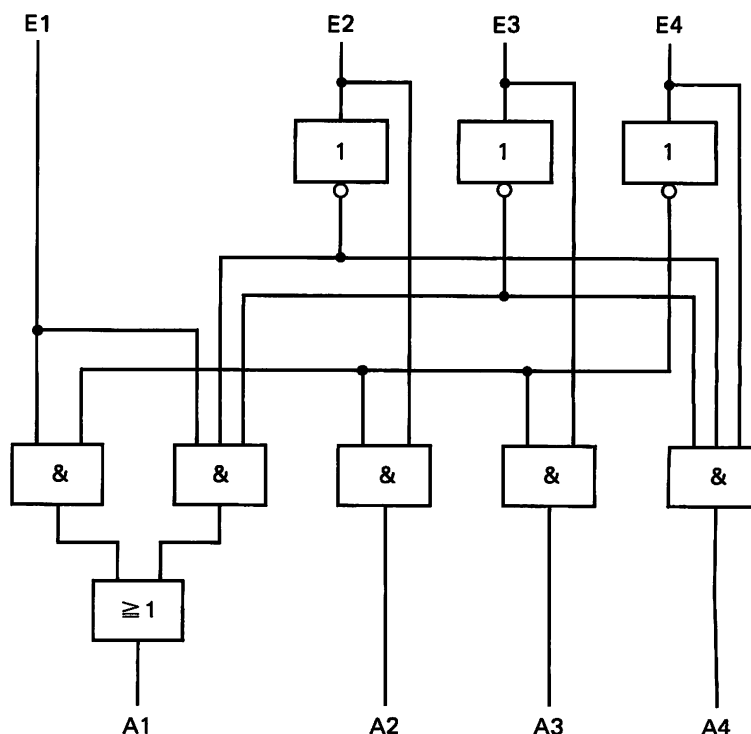


Bild B 80.1
Pseudotetradenkontrolle.
E1 \triangleq 1 E2 \triangleq 2 E3 \triangleq 4 E4 \triangleq 8

- d) Mit Hilfe des Programms ist die Funktion der vorgegebenen Pseudotetradenkontrolle von Bild B 80.1 experimentell zu überprüfen. Nehmen Sie die folgende Funktionstabelle B 81.1 auf.

Tabelle B 81.1: Funktionstabelle zum Beispiel B 80.1

Dezimal- ziffer	Eingang				Ausgang				
	E4	E3	E2	E1	A4	A3	A2	A1	
0	0	0	0	0	0	0	0	0	Tetraden
1	0	0	0	1	0	0	0	1	
2	0	0	1	0	0	0	1	0	
3	0	0	1	1	0	0	1	1	
4	0	1	0	0	0	1	0	0	
5	0	1	0	1	0	1	0	1	
6	0	1	1	0	0	1	1	0	
7	0	1	1	1	0	1	1	1	
8	1	0	0	0	1	0	0	0	
9	1	0	0	1	1	0	0	1	
	1	0	1	0	0	0	0	0	Pseudo- tetraden
	1	0	1	1	0	0	0	0	
	1	1	0	0	0	0	0	0	
	1	1	0	1	0	0	0	0	
	1	1	1	0	0	0	0	0	
	1	1	1	1	0	0	0	0	

Lösung:

- a) Nach dem Schaltnetz von Bild B 80.1 können Sie folgende Schaltfunktionen aufstellen:

$$A1 = E1 \overline{E4} \vee E1 \overline{E2} \overline{E3}$$

$$A2 = E2 \overline{E4}$$

$$A3 = E3 \overline{E4}$$

$$A4 = \overline{E2} \overline{E3} E4$$

- b) Den KOP zeigt das Bild B 81.1.

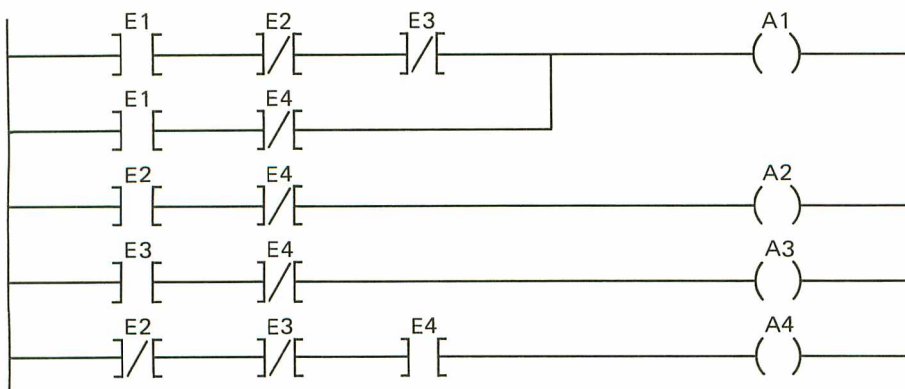


Bild B 81.1
KOP der Pseudotetradenkontrolle,
Beispiel B 80.1.

- c) Die Schaltfunktionen kann man unmittelbar programmieren.

Programm:

$!E1 \& NE4 / E1 \& NE2 \& NE3 = A1$

$!E2 \& NE4 = A2$

$!E3 \& NE4 = A3$

$!NE2 \& NE3 \& E4 = A4$

$!PE$

- d) Der Programmtest ergibt die in Tabelle B 80.1 ROT eingetragenen Ergebnisse. Das Programm erfüllt damit die Vorgaben.

B

82

Aufgabe B 82.1

Pseudotetradenkontrolle

Das Programm von Beispiel B 80.1 ist wie folgt zu erweitern:

Bei gültigen Tetraden soll zusätzlich der Ausgang A0 den Zustand 0, bei Pseudotetraden den Zustand 1 annehmen. Mit dem Wert von A0 sind die Anzeigen A1 bis A4 eindeutig den Tetraden bzw. den Pseudotetraden zugeordnet.

- Stellen Sie für die Variable A0 die Schaltfunktion auf.
- Die Schaltfunktion aus a) ist mit Hilfe der Schaltalgebra optimal zu vereinfachen.
- Programmieren Sie das Ergebnis von b).
- Mit dem Programm von Beispiel B 80.1 und dem Teilprogramm aus c) ist das Gesamtprogramm der Pseudotetradenkontrolle aufzustellen und zu testen. Nehmen Sie die Funktionstabelle nach folgendem Muster auf:

Tabelle B 82.1: Muster der Funktionstabelle zur Aufgabe B 82.1.

Dezimal- ziffer	Eingabe				Ausgabe				A0
	E4	E3	E2	E1	A4	A3	A2	A1	
0	0	0	0	0					
.								
.								
9	1	0	0	1					
<hr/>									
	1	0	1	0					
								
								
	1	1	1	1					

Die Lösung dieser Aufgabe finden Sie auf Seite F53.

Programm zur Fehlererkennung im Minimalcode mittels Parity-Check

Es ist Ihnen bereits bekannt, daß beim Minimalcode dem Codewort zur Fehlererkennung noch ein Prüfbit angefügt wird. Im folgenden Beispiel ist der Wert des Prüfbits so gewählt, daß die Quersumme der 1-Zustände im erweiterten Codewort gerade ist (wird als Code mit ganzzahligem Gewicht bezeichnet). Für den BCD-Code gilt die folgende Codetabelle B 83.1.

Tabelle B 83.1: Codewort mit Prüfbit des BCD-Codes bei ganzzahligem Gewicht

Dezimal- ziffer	Codewort				Prüf- bit
	E4	E3	E2	E1	E0
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0

Das Schema der Prüfmethode ist aus Bild B 83.1 zu ersehen. Nach dieser Methode wird untersucht, ob das erweiterte Codewort fehlerbehaftet ist.

Wirkungsweise:

Aus Tabelle B 83.1 werden zunächst die Codeworte ausgesucht, bei denen das Prüfbit den Wert 1 haben muß. Eine Logik liefert für diese Codeworte am Ausgang den Zustand 1 ($M0=1$). Der Zustand der Ausgangsvariablen $M0$ und des Prüfbits $E0$ werden miteinander verglichen. Bei gleichen Zuständen von $E0$ und $M0$ soll der Vergleich den Wert $A0=1$ liefern.

Die Codeworte, bei denen das Prüfbit den Wert 0 haben muß, ergeben (falls sie nicht fehlerbehaftet sind) am Ausgang der Logik den Wert $M0=0$. Die Werte $M0=0$ und $E0=0$ sollen am Ausgang der Prüfschaltung ebenfalls den Wert $A0=1$ erzeugen (Meldung: kein Fehler vorhanden). Es können nicht alle Fehler erkannt werden. 2 oder 4 falsche Bits im Codewort bzw. 1 oder 3 falsche Bits und ein falsches Prüfbit ergeben die Meldung: kein Fehler. Aus dem Vergleich folgt:

Bei $A0=1$ liegt kein Übertragungsfehler vor, bei $A0=0$ ist das Codewort fehlerbehaftet.

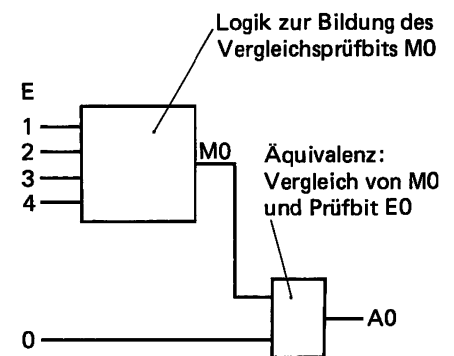


Bild B 83.1
Schema zur Prüfung eines Codeworts (E1 bis E4) mit Prüfbit (E0).

Beispiel B 84.1

Entwicklung eines Programms zur Fehlererkennung

Die zu übertragenden Dezimalzahlen sind nach dem BCD-Code verschlüsselt und zur Fehlererkennung mit einem Prüfbit versehen. Die Quersumme der 1-Zustände im Codewort mit Prüfbit ist gerade (Tabelle B 83.1).

- Für die Logik zur Bildung des Vergleichsprüfbits M0 (siehe Bild B 83.1) ist die Schaltfunktion aufzustellen.
- Geben Sie die Gesamtfunktion der Prüfschaltung als FUP wieder.
- Stellen Sie das Programm nach dem FUP auf.
- Das Programm ist durch Aufnahme der Funktionstabelle B 85.1 zu testen. In der Tabelle sind fehlerbehaftete und fehlerfreie Kombinationen aufgeführt.

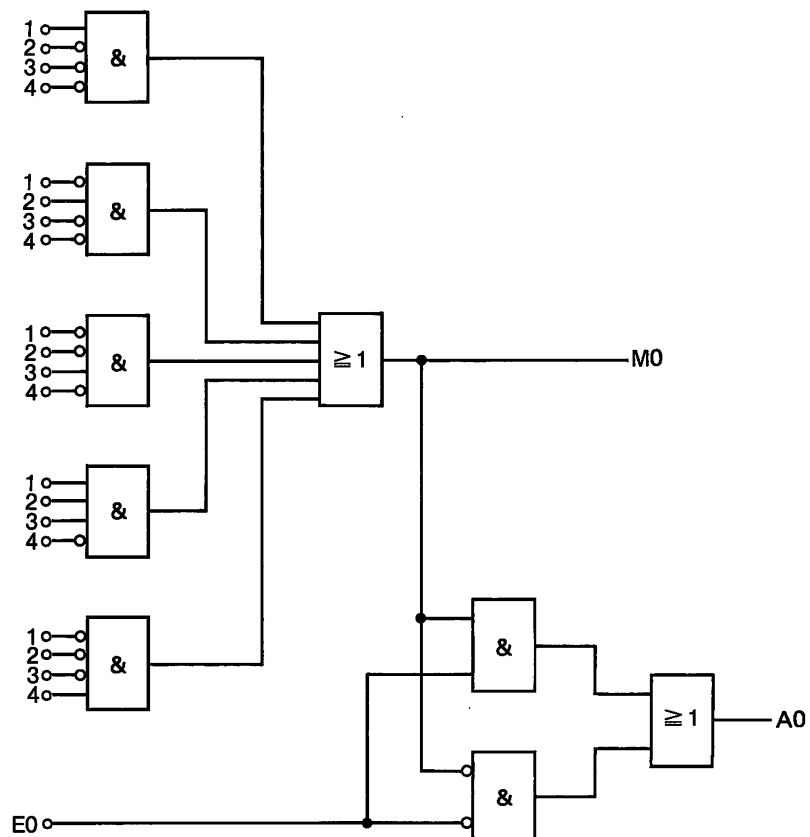


Bild B 84.1
FUP zur Überprüfung eines Codeworts mit Prüfbit.

Lösung:

- Die Schaltfunktion enthält die Signalkombinationen, die nach der Aufgabenstellung den Zustand $M0=1$ ergeben.

$$M0 = E1 \bar{E2} \bar{E3} \bar{E4} \vee \bar{E1} E2 \bar{E3} \bar{E4} \vee \bar{E1} \bar{E2} E3 \bar{E4} \vee E1 E2 E3 \bar{E4} \vee \bar{E1} \bar{E2} \bar{E3} E4$$

Tabelle B 85.1: Funktionstabelle zum Beispiel B 84.1

E4	E3	E2	E1	E0	A0	Fehler (ja/nein)
0	0	0	0	0	1	nein
0	0	0	1	0	0	ja
0	0	1	0	1	1	nein
0	0	1	1	0	1	nein
0	1	0	0	0	0	ja
0	1	0	1	1	0	ja
0	1	1	0	0	1	nein
0	1	1	1	0	0	ja
1	0	0	0	1	1	nein
1	0	0	1	0	1	nein
1	0	0	1	1	0	ja

- b) Den FUP zeigt das Bild B 84.1. Zur Bildung der Meldung (A0) sind die Variablen M0 und E0 nach einer Äquivalenz-Funktion zu verknüpfen. Laut Aufgabenstellung gilt: $A0 = E0M0 \vee \overline{E0}\overline{M0} = 1$
- c) Das Programm läßt sich unmittelbar nach dem FUP erstellen. Sicherlich haben Sie beim Programmieren derartiger Aufgaben keine Probleme mehr.

Programm:

```

!E1&NE2&NE3&NE4      /NE1&NE2&NE3&E4
/NE1&E2&NE3&NE4      =M0
/NE1&NE2&E3&NE4      !M0&E0/NM0&NE0=A0
/E1&E2&E3&NE4        !PE

```

- d) Der Test muß die in Tabelle B 85.1 ROT eingetragenen Ergebnisse liefern.

Decoder

Die im Minimalcode verschlüsselten Dezimalziffern umfassen vier Bits. Man könnte die Codeworte durch Ansteuern von vier LEDs zur Anzeige bringen. Für den Menschen wäre bereits diese Anzeige nur schwierig zu übersetzen, Lesefehler wären die Folgen.

Wir benötigen deshalb **Decoder**, die die verschlüsselten Informationen in eine leicht lesbare Darstellung übersetzen.

Beispiel B 85.1

Programmierung eines Decoders „BCD-Code in 1 aus 10“

Die Dezimalziffern liegen im BCD-Code verschlüsselt vor. Beim Decoder „1 aus 10“ ist jeder Dezimalziffer eine Anzeige (LED) zugeordnet. Von den 10 Anzeigeelementen darf immer nur ein Element aktiv sein. Das Schema des Decoders zeigt das Bild B 86.1. Die Decodierung soll nach der Codetabelle B 86.1 erfolgen.

Tabelle B 86.1: Codetabelle zum Beispiel B 85.1

Dezimal- ziffer	Eingang				Ausgang									
	E3	E2	E1	E0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0					
2	0	0	1	0	0	0	1	0	0				
3	0	0	1	1	0	0	0	1	0	0			
4	0	1	0	0	0	0	1	0	0			
5	0	1	0	1	0	0	1	0	0			
6	0	1	1	0	0	0	1	0	0			
7	0	1	1	1	0	0	1	0	0			
8	1	0	0	0	0	0	1	0					
9	1	0	0	1	0	0	1	0					

- a) Stellen Sie bitte die Schaltfunktionen für die Ausgangsvariablen A0 bis A9 auf.
- b) Die Funktion des Decoders ist durch ein Programm nachzubilden, zum Programmtest ist die Codetabelle experimentell zu überprüfen.

Lösung:

- a) Schaltfunktionen:

$$A0 = \overline{E0} \overline{E1} \overline{E2} \overline{E3}$$

$$A1 = \overline{E0} \overline{E1} E2 \overline{E3}$$

$$A2 = \overline{E0} E1 \overline{E2} \overline{E3}$$

$$A3 = \overline{E0} E1 E2 \overline{E3}$$

$$A4 = \overline{E0} \overline{E1} E2 \overline{E3}$$

$$A5 = \overline{E0} \overline{E1} E2 E3$$

$$A6 = \overline{E0} E1 E2 \overline{E3}$$

$$A7 = \overline{E0} E1 E2 E3$$

$$A8 = \overline{E0} \overline{E1} \overline{E2} E3$$

$$A9 = \overline{E0} \overline{E1} E2 E3$$

- b) Programm:

$$NE0 \& NE1 \& NE2 \& NE3 = A0$$

$$!E0 \& NE1 \& NE2 \& NE3 = A1$$

$$!NE0 \& E1 \& NE2 \& NE3 = A2$$

$$!E0 \& E1 \& NE2 \& NE3 = A3$$

$$!NE0 \& NE1 \& E2 \& NE3 = A4$$

$$!E0 \& NE1 \& E2 \& NE3 = A5$$

$$!NE0 \& E1 \& E2 \& NE3 = A6$$

$$!E0 \& E1 \& E2 \& NE3 = A7$$

$$!NE0 \& NE1 \& NE2 \& E3 = A8$$

$$!E0 \& NE1 \& NE2 \& E3 = A9$$

$$!PE$$

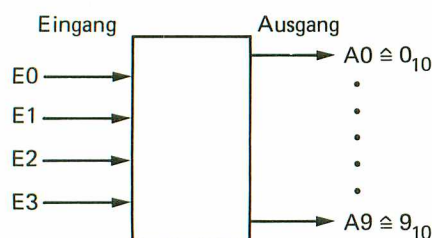


Bild B 86.1
Schema des Decoders „BCD in
1 aus 10“.

Nach fehlerfreier Eingabe stimmen die Testergebnisse mit der Codetabelle B 86.1 überein.

Aufgabe B 86.1

Programmierung eines Decoders „3 zu 8“

Die Schaltung des Decoders zeigt das Bild B 87.1. Mit den drei Eingangsvariablen E0 bis E2 lassen sich acht verschiedene Signalkombinationen bilden. Nach deren Decodierung wird einer der acht Ausgänge A0 bis A7 angesteuert.

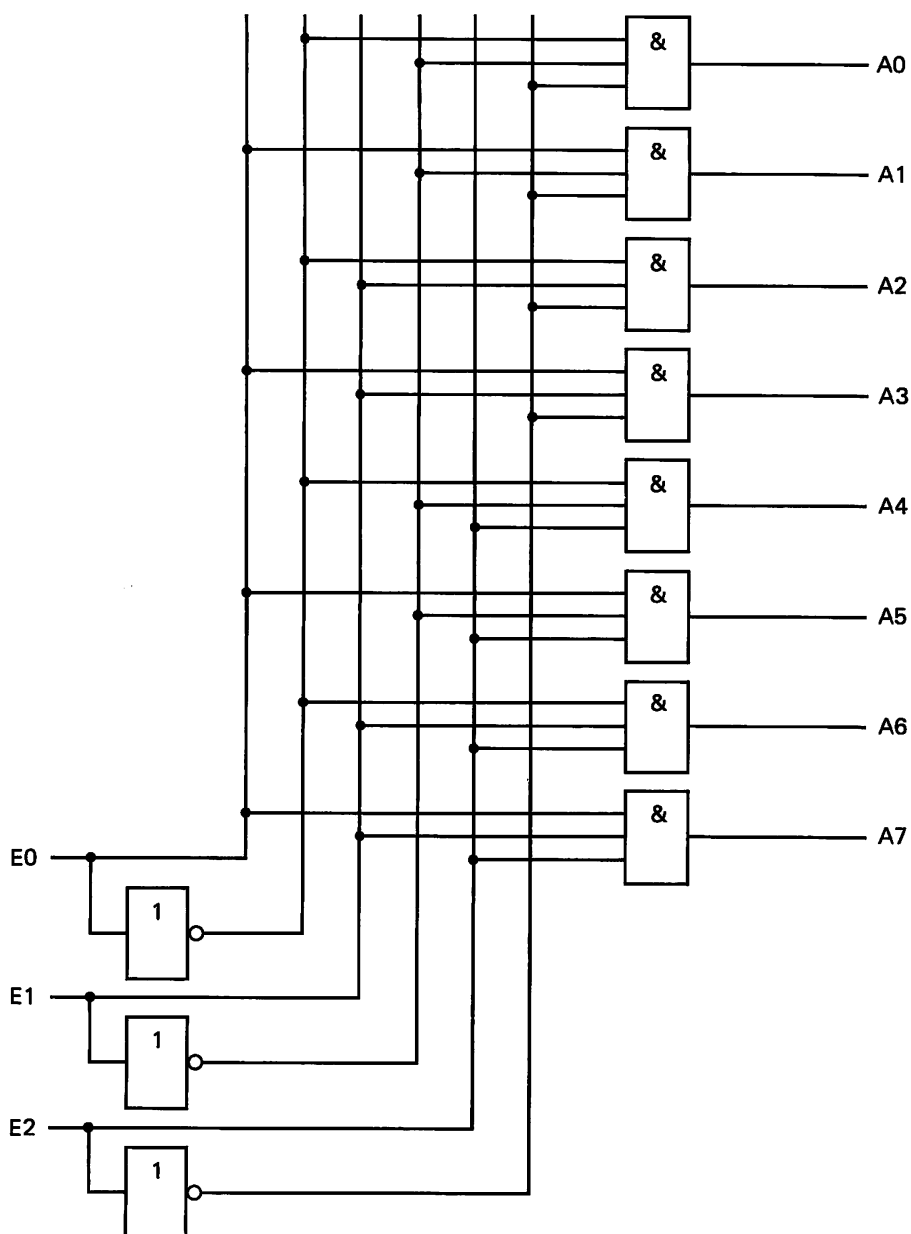


Bild B 87.1
Decoder 3 zu 8, Aufgabe B 86.1.

Tabelle B 87.1: Codetabelle zur Aufgabe B 86.1

Eingang			Ausgang							
E2	E1	E0	A0	A1	A2	A3	A4	A5	A6	A7
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

- a) Übersetzen Sie die Schaltung von Bild B87.1 in einen KOP.
- b) Nach dem KOP ist das Programm aufzustellen.
- c) Ergänzen Sie experimentell die Codetabelle B87.1.

Die Lösung dieser Aufgabe finden Sie auf Seite F54.

B**88**

Zusammenfassung

Ein Code ist eine Vorschrift für die eindeutige Zuordnung eines Zeichenvorrats zu den Zeichen eines anderen Vorrats. Binärcodes für Dezimalzahlen verschlüsseln jeweils 4 Binärzeichen (4 Bit = 1 Tetrade) zu den Dezimalziffern 0 bis 9. Da sich mit vier Bit 16 Ziffern verschlüsseln lassen, enthalten diese Codes sechs nicht verwertbare Bitkombinationen, die Pseudotetraden genannt werden. Minimalcodes enthalten nur soviel Binärzeichen je Codewort wie zur Darstellung der Information unbedingt erforderlich sind. Optimalcodes verwenden mehr Binärzeichen. Ein zusätzliches Binärzeichen (Prüfbit) kann z. B. zur Erkennung von Codierungsfehlern angefügt werden.

Zählfunktionen

Wenn wir die Definition für Zählschaltungen (darunter versteht man die Hardwarelösung einer Zähleinrichtung) auf programmierte Zähler übertragen, so gilt nach DIN 44 300:

Ein Zähler ist ein Schaltwerk, in dem eine Zahl gespeichert ist. Zu dieser Zahl kann in Abhängigkeit einer Schaltvariablen (z. B. eines Taktsignals) eine konstante Zahl, die sogenannte Zähleinheit, addiert werden. In unserem Fall beträgt die Zähleinheit +1 oder -1.

Zähler finden Sie u. a. in Programmen der Meßtechnik und Steuerungstechnik eingesetzt. Der **Zählerstand** gibt Auskunft über den Wert einer Prozeßgröße, den Stand eines Arbeitsablaufs oder den Zustand einer Steuerung. Zählschaltungen sind mit Speichergliedern aufgebaut, Zählfunktionen benötigen Speicherfunktionen.

Die Zählschaltungen und -funktionen kann man nach verschiedenen Gesichtspunkten einteilen. Eine Grobeinteilung und deren charakteristische Merkmale enthält die folgende Zusammenstellung.

Die Einteilung der Zählschaltungen und Zählfunktionen kann erfolgen nach:

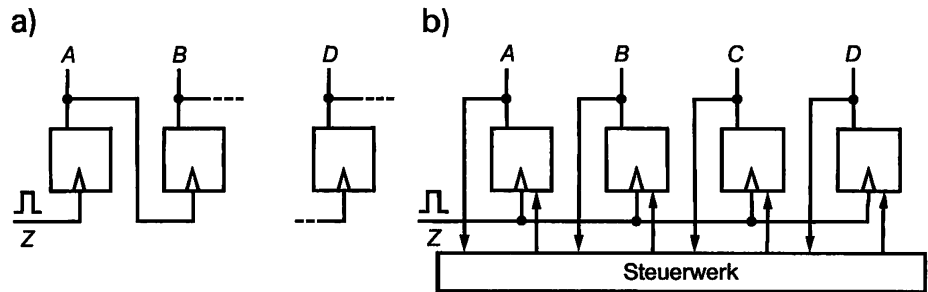
1) Art des verwendeten Zahlensystems:

Wir unterscheiden zwischen dem Dezimalzähler und dem Dualzähler. Ein Zähler mit vier Speicherelementen kann von 0 bis 15 zählen. Er arbeitet in diesem Fall als Dualzähler. Die binären Zustände der Ausgangsvariablen der Speicherglieder bzw. -funktionen bilden den **codierten Dezimalwert**. Dezimalzähler haben eine Dekadenstruktur. Eine Dekade zählt nur von 0 bis 9. Beim 10-ten Zählimpuls schaltet sie wieder in die Nullstellung und erzeugt ein Übertragungssignal zum Ansteuern der nächsthöheren Dekade. Auch eine Zähldekade benötigt vier Speicherglieder. Zählschaltungen können für verschiedene Codes (siehe Tabelle B72.1) entworfen werden.

2) Art der Ansteuerung durch die Zählsignale:

Wir unterscheiden zwischen **asynchroner** und **synchroner** Ansteuerung. Das Bild B 90.1 zeigt die beiden Prinzipien. Beim Asynchrone Zähler erhält nur der Zähl Eingang der ersten Speicherfunktion das Zähl signal zugeführt. Die weiteren Speicherfunktionen (oder -glieder) werden von der Ausgangsvariablen vorangehender Speicherfunktionen angesteuert. Das asynchrone Prinzip hat für programmierte Zählfunktionen Bedeutung. Bekanntlich bearbeitet die SPS die Anweisungen nacheinander.

Bild B 90.1
Zähleransteuerung,
a) Asynchronzähler,
b) Synchronzähler,
Z Zählimpulse.



Für die Schaltungstechnik sind synchron arbeitende Zählschaltungen wichtiger. Alle Speicherglieder erhalten den Zählimpuls Z gleichzeitig zugeführt. Das Steuerwerk, das den momentanen Zählerstand auswertet, sorgt dafür, daß mit dem eintreffenden Zählimpuls die Speicherglieder in die vom Zählcode festgelegte Lage kippen. Die maximale Zählfrequenz ist bei synchron arbeitenden Zählern größer als bei asynchron arbeitenden Zählern.

3) Art der Zählrichtung:

Es gibt Vor- und Rückwärtszähler. Ein Vorwärtszähler hat in der Grundstellung den Zählerstand Null, der Rückwärtszähler hat dagegen seinen höchsten Zählerstand. Mit einem prozeßabhängigen Steuersignal kann die Zählrichtung gewählt oder umgeschaltet werden.

4) Zählerarten:

Beschrieben wurden bereits die Dezimal- und Dualzähler. Außer diesen zwei Arten gibt es noch weitere **Zählerstrukturen**, wovon einige für die Steuerungstechnik sehr wichtig sind. Ihr Zählprinzip ist nachstehend kurz erläutert.

Modulo-Zähler: Die Steuerungstechnik benötigt für verschiedene Aufgaben Zähler mit einer festen Schrittzahl. Deren Zählkapazität (=Zahl der Zählsschritte) ist meistens geringer als diejenige, die durch die Zahl der verwendeten Speicher erreichbar wäre.

Modulo-n-Zähler: n bezeichnet die Zahl der Zählsschritte. Ein Beispiel soll Ihnen die Arbeitsweise näher erläutern. Der Modulo-6-Zähler hat 6 Zählsschritte. Er benötigt drei Speicherfunktionen. Sind A1, A2 und A3 deren Ausgangsvariablen, so wäre

A1, A2, A3 = 0, 0, 0 der niedrigste Zählerstand und
 = 1, 0, 1 der höchste Zählerstand.

Es werden somit 6 verschiedene Ausgangszustände von den 8 möglichen ausgenutzt. Die Wertigkeiten der Ausgangsvariablen sind nach dem Dualcode abgestuft.

Modulo-2^m-Zähler: Der Zähler arbeitet ebenfalls im Dualcode. Bei m = 3 beträgt beispielsweise die Schrittzahl 2³ = 8.

Ringzähler: Sie bestehen aus einer Kettenschaltung von Speichergliedern (-funktionen) und arbeiten vielfach mit dem (^m/_n)-Code

(sprich: „n aus m Code“). m bezeichnet die Zahl der in Kette geschalteten Speicherglieder (= Zählkapazität). Mit n wird die Zahl der im Codewort enthaltenen 1-Zustände bezeichnet.

Beim Ringzähler mit dem (10_1) -Code besteht das Codewort aus 10 Bits (10 Speichergliedern). Von diesen führt immer nur ein Bit in den Zustand 1. Mit jedem Zählimpuls wird der 1-Zustand um ein FF weitergeschaltet.

Zusammenfassung

Zähler bzw. Zählfunktionen gehören zur Gruppe der Schaltwerke. Sie sind durch ein sequentielles Verhalten charakterisiert. Es stellen sich nacheinander in einer bestimmten zwangsläufigen Folge bestimmte Schaltzustände ein. Die Konfiguration der Ausgangsvariablen nach einer Ansteuerung ist nicht nur allein von der Konfiguration der Eingangsvariablen, sondern auch noch vom inneren Schaltzustand des Schaltwerks vor der Ansteuerung abhängig. Darin unterscheiden sich die Schaltwerke von den Schaltnetzen.

Zählschaltungen

Zur Vorbereitung auf die Programmierung von Zählfunktionen wollen wir die Arbeitsweise des Zählers an einigen typischen Zählschaltungen untersuchen.

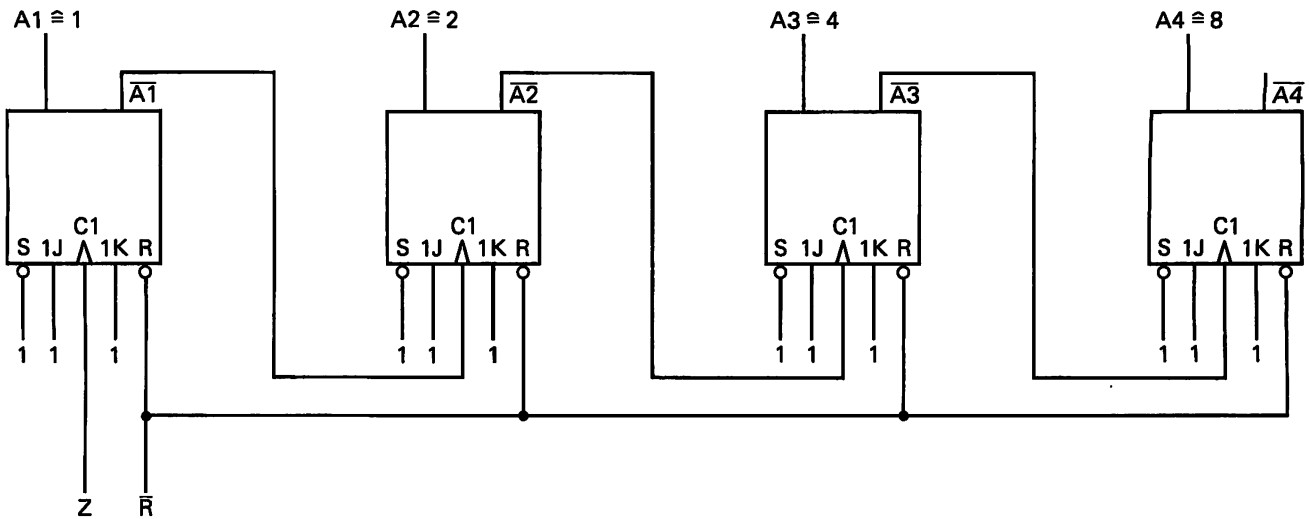
Asynchroner Vorwärtzähler für den Dualcode

Die Schaltung in Bild B 92.1a ist mit vier einflankengesteuerten JK-Speichergliedern aufgebaut. Bei Beschaltung der Eingänge J und K mit 1-Signal ändert sich mit jedem Zähltakt am Takteingang C1 der Zustand der Ausgangsvariablen A. Die zusätzlichen statischen Eingänge S und R können zum taktunabhängigen Setzen bzw. Rücksetzen des Speicherglieds benutzt werden. Das Umkehrzeichen am Eingang drückt aus, daß beide Eingänge 0-aktiv sind, d. h. die Ansteuerung muß mit einem 0-Signal erfolgen. Der S-Eingang wird nicht benötigt, er erhält deshalb ein 1-Signal. Mit einem 0-Signal am R-Eingang der Speicherglieder bringt man die Zählschaltung in die Grundstellung, die Ausgänge der Speicherglieder A1 bis A4 nehmen den Zustand 0 an. Von dieser Einstellung gehen wir aus.

Das Kippen der Speicherglieder wird durch einen Zustandswechsel von 0 in 1 am Takteingang C1 ausgelöst. Mit jedem Kippvorgang ändert sich der Ausgangszustand A von 0 nach 1 oder von 1 nach 0. Wenn wir den invertierenden Ausgang A eines Speicherglieds mit dem Takteingang C1 des nachfolgenden Speicherglieds verbinden, wird mit jedem **zweiten Zustandswechsel** $0 \rightarrow 1$ am Eingang des ersten Speicherglieds das folgende Speicherglied zum Kippen gebracht.

Besser als viele Worte gibt das Signal-Zeit-Diagramm in Bild B 92.1b den Sachverhalt wieder.

a)



b)

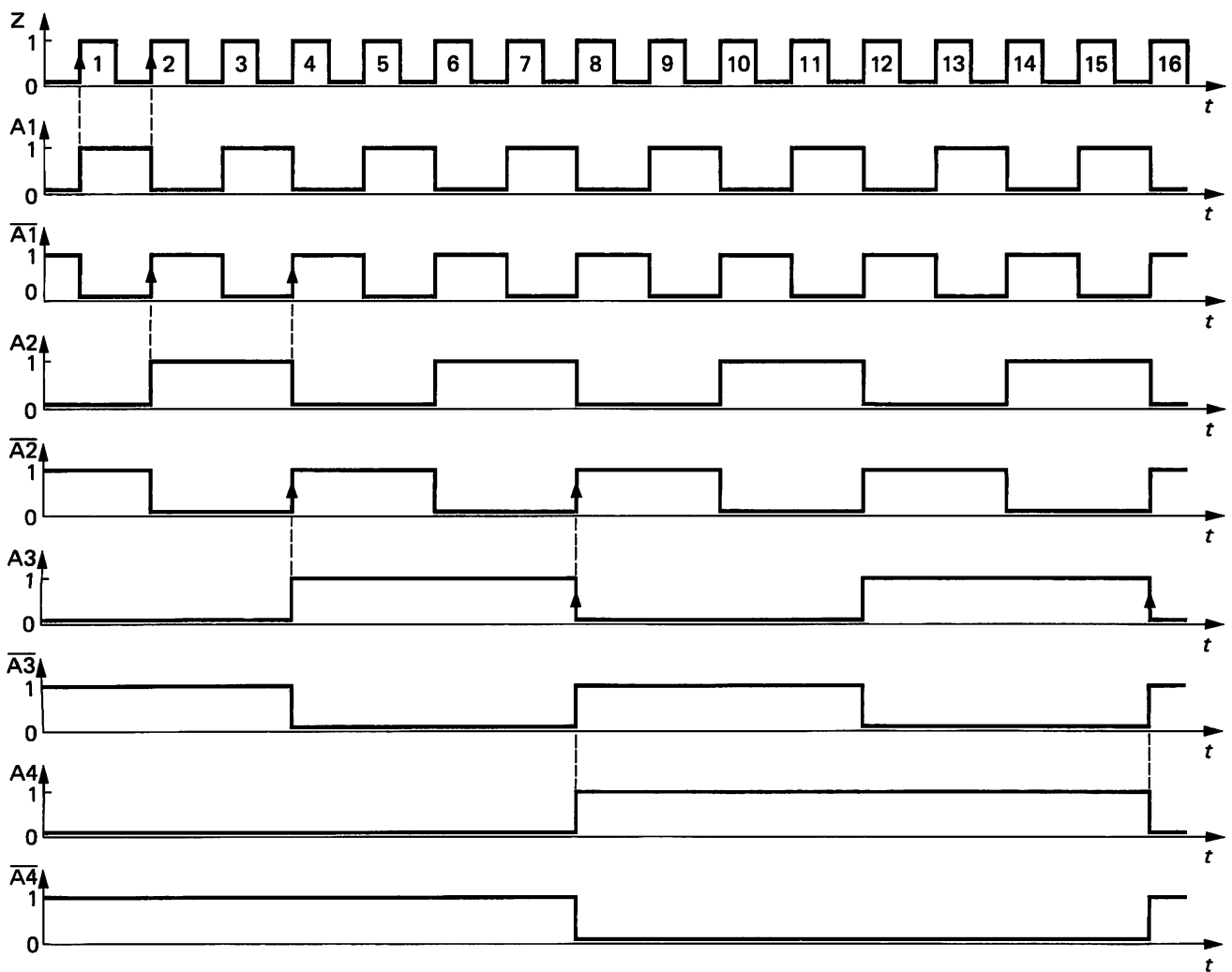


Bild B 92.1
Asynchroner Dualzähler.
a) Schaltung,
b) Signal-Zeit-Diagramm.

Beim Asynchron-Vorwärtszähler nach Bild B 92.1a steuert ein in den Rücksetzzustand kippendes Speicherglied das nächstfolgende Speicherglied aktiv an.

Jedes Speicherglied bewirkt zwischen Zählakteingang und Speichergliedausgang eine Frequenzteilung im Verhältnis 2:1 (zwei Eingangs-takte erzeugen einen Ausgangstakt). Jedem Speichergliedausgang ist eine dezimale Wertigkeit zugeordnet. Die Summe der Wertigkeiten der Ausgänge, die den Zustand 1 führen, ergeben den Zählerstand.

Durch schaltungstechnische Maßnahmen kann man erreichen, daß die Schaltung nur vom Dezimalwert 0 bis zum Dezimalwert 9 zählt (A1, A2, A3, A4 = 0,0,0,0 bis 1,0,0,1) und beim zehnten Impuls wieder in die Ausgangsstellung (Zustand 0,0,0,0) kippt. Die Schaltung arbeitet dann als **dezimale Zähldekade**.

Synchrone Vorwärtzähldekade für den BCD-Code

Das Bild B 93.1 zeigt die Zähl-schaltung einer synchron arbeitenden Zähldekade. Der eigentliche Zähler besteht aus vier zweiflanken-gesteuerten JK-MS-FFs. An den Ausgängen A1 bis A4 steht das Code-wort, das dem Zählerstand entspricht. Die Ausgänge haben die ange-ggebenen, dezimalen Wertigkeiten.

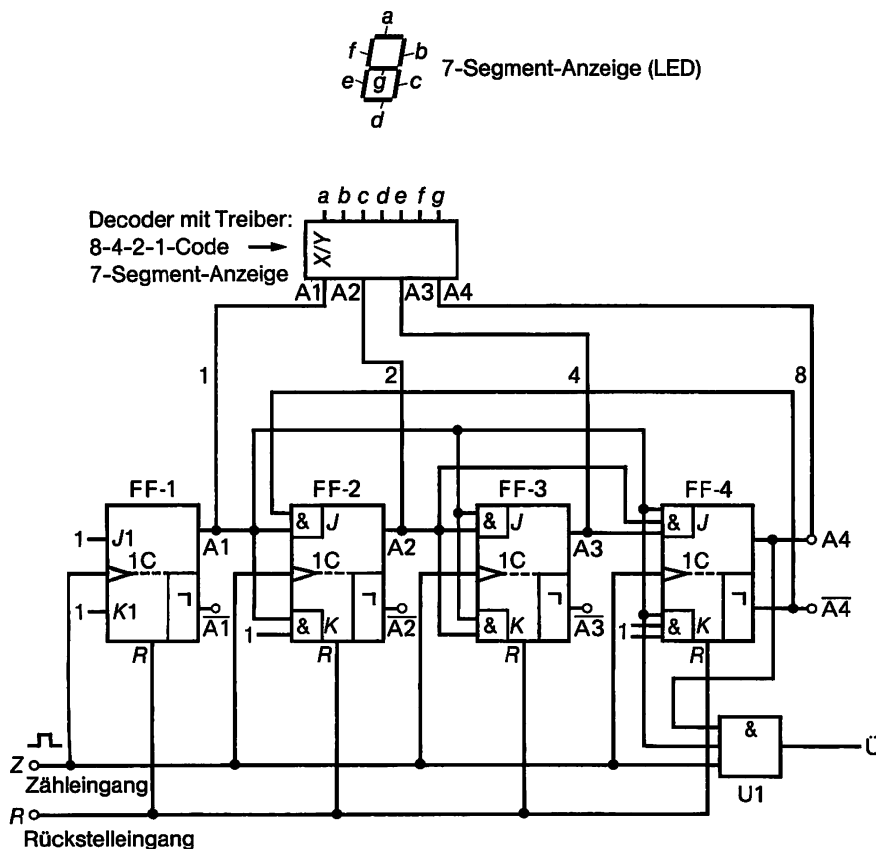


Bild B 93.1
Synchrone Vorwärtszähldekade
mit JK-MS-FF für den BCD-Code
mit Decoder und Sieben-Segment-
Anzeige.

Eine Sieben-Segment-Anzeige soll den codierten Dezimalwert des Zählers anzeigen. Der vorgeschaltete Decoder setzt die BCD-Signale des Zählers in Signale zur Ansteuerung der LED-Segmente um. Ein Treiber erzeugt den für die LEDs erforderlichen Strom.

Mit Zustand 1 an R erfolgt das Rücksetzen der Dekade auf den Anfangszustand Null ($A_1, A_2, A_3, A_4 = 0, 0, 0, 0$).

B**94**

Beispiel B 94.1

Für die synchrone Zähldekade nach Bild B 93.1 wollen wir eine Zustandstabelle aufstellen.

Die Zustandstabelle enthält die sich nach jedem Zähltakt ergebenden Signalzustände an den Ausgängen und an den vorbereitenden Eingängen. Sie ist die Funktionstabelle einer Zehlschaltung.

Wir lösen diese Aufgabe schrittweise gemeinsam.

Lösung:

1. Schritt: An den vorbereitenden Eingängen der Speicherglieder 2 bis 4 liegen Steuersignale an, die mit UND verknüpft sind. Das Ergebnis der UND-Verknüpfungen in Verbindung mit dem Zähltakt steuert den Kippvorgang des betreffenden Speicherglieds. Für die steuernden J- und K-Signale lassen sich die folgenden Schaltfunktionen nach Bild B 93.1 aufstellen:

$$\begin{array}{lll}
 \text{FF 1:} & J = 1 & K = 1 \\
 \text{FF 2:} & J = A_1 \wedge \overline{A_4} & K = A_1 \\
 \text{FF 3:} & J = A_1 \wedge A_2 & K = A_1 \wedge A_2 \\
 \text{FF 4:} & J = A_1 \wedge A_2 \wedge A_3 & K = A_1 \\
 \text{Übertrag:} & \dot{U} = A_1 \wedge A_4 \wedge Z
 \end{array}$$

In der elektronischen Schaltungstechnik sind die UND-Glieder der vorbereitenden Eingänge im Baustein integriert. Mit diesen UND-Gliedern ist das Steuerwerk von Bild B 90.1b realisiert.

2. Schritt: Wir stellen die Zustandstabelle B 95.1 auf. Mit den Zählimpulsen muß ein vom verwendeten Zählcode abhängiger Signalzustand an den Ausgängen der Speicherglieder entstehen. Dies setzt voraus, daß an den vorbereitenden Eingängen der Speicherglieder bestimmte Signalkombinationen wirksam sind. Wir können die Zustände der Steuersignale mit den Schaltfunktionen berechnen.

In jeder Zeile von Tabelle B 95.1 kennzeichnet die obere Signalfolge die momentanen Zustände der Ausgänge. Die sich daraus ergebenden Signalzustände an den vorbereitenden Eingängen J und K sind etwas tiefer gesetzt.

Tabelle B95.1: Zustandstabelle der Vorwärtszähldekade im BCD-Code

Zählimpuls Z	FF1 A1 1	FF2 J K A2 2	FF3 J K A3 4	FF4 J K A4 8	Dezimalziffer
0	0	0 0	0 0	0 0	0
1	1	1 1	0 0	0 1	1
2	0	0 0	0 0	0 0	2
3	1	1 1	1 1	0 1	3
4	0	0 0	0 0	0 0	4
5	1	1 1	0 0	0 1	5
6	0	0 0	0 0	0 0	6
7	1	1 1	1 1	1 1	7
8	0	0 0	0 0	0 0	8
9	1	0 1	0 0	0 1	9
10	0	0 0	0 0	0 0	0

Sie müssen die in Tabelle B95.1 eingetragenen Signalfolgen selbst nachvollziehen. Vorher sollten Sie sich noch einmal an die Funktion des JK-MS-FF erinnern (siehe Tafel 6, Seite 99 in Lehrbrief 2). Als Hilfestellung geben wir Ihnen folgende Hinweise:

- Führen die beiden vorbereitenden Eingänge den Zustand $J=K=1$, Dann ändert sich mit jedem Zähltakt der Signalzustand an den Ausgängen in der Folge 0-1-0-1
- Bei $J=0$ und $K=1$ wird das Speicherglied mit dem Zähltakt rückgesetzt ($A=0$). Befindet es sich bereits in dieser Lage, dann entsteht am Ausgang keine Zustandsänderung.
- Bei $J=1$ und $K=0$ wird das Speicherglied mit dem Zähltakt gesetzt ($A=1$). Befindet es sich schon in dieser Lage, dann entsteht am Ausgang keine Zustandsänderung.

Zur Veranschaulichung wollen wir einige Zeilen in Tabelle B95.1 gemeinsam betrachten. Bei $Z=0$ ist noch kein Zählimpuls eingetroffen. Alle Speicherglieder befinden sich im rückgesetzten Zustand und haben am Signalausgang den Zustand 0.

Beim 1. Zählimpuls ($Z=1$) kippt nur A_1 vom Zustand 0 in den Zustand 1. Nur dort ist die Kippbedingung $J=K=1$ erfüllt.

Die nunmehr entstandene Signalkombination $A_1, A_2, A_3, A_4 = 1, 0, 0, 0$ erzeugt an den vorbereitenden Eingängen von Speicherglied 2 die Signale $J=K=1$, denn es sind $J = A_1 \wedge \bar{A}_4 = 1$ und $K = A_1 = 1$. Mit dem zweiten Zählimpuls schaltet das Speicherglied 2 in den Zustand 1 und Speicherglied 1 schaltet wieder in den Zustand 0.

Nach dem Schema dieser kurzen Anleitung können Sie die Signalzustände für die weiteren Zählakte in Tabelle B 95.1 selbst überprüfen.