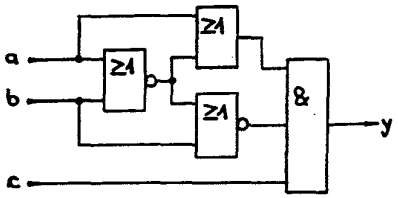
KNF und DNF einer Funktion:

Geben Sie für die Funktion $a \vee (b \not\equiv c)$ die konjunktive und die disjunktive Normalform an.

Logischer Ausdruck und Wahrheitstabelle einer Schaltung:



Geben Sie den logischen Ausdruck für die gezeichnete Funktion an. Stellen Sie die Wahrheitstabelle auf und geben Sie eine Realisierung an, die nur *NAND-Gatter* verwendet.

Äquivalenzüberprüfung:

Stellen Sie an Hand der Wahrheitstabelle fest, ob die beiden Funktionen

$f_1 = (\bar{a}\&\bar{b}\&\bar{c}) \vee (a\&\bar{b}\&c) \vee (a\&b\&\bar{c}) \vee (\bar{a}\&b\&c)$

$f_2 = (\bar{a}\&\bar{b}) \vee (\bar{a}\&\bar{c}) \vee (\bar{b}\&\bar{c})$

äquivalent sind.

KNF und minimale DF einer logischen Funktion:

Eine logische Funktion y von vier Variablen ist durch die Wahrheitstabelle gegeben:

a	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
d	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y	1	0	1	0	0	1	1	1	1	0	1	0	1	1	1	0

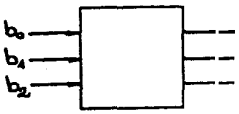
Geben Sie die konjunktive Normalform und (mit Hilfe des KV-Diagrammes) eine minimale disjunktive Form dieser Funktion an.

Codeumsetzer:

In einem digitalen System werden die Dezimalziffern von 0 bis 5 durch einen Code A und einen Code B dargestellt:

	a_2	a_1	a_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1

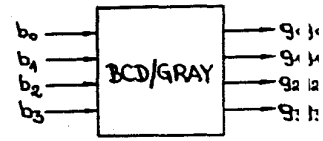
b_2	b_1	b_0
1	0	1
1	1	0
1	1	1
0	0	0
0	0	1
0	1	0



Entwerfen Sie ein logisches Netzwerk, das den Code B in den Code A umwandelt. Die notwendigen logischen Funktionen sind mit Hilfe des KV-Diagramms in ihrer minimalen disjunktiven Form darzustellen. Es kann dazu vorausgesetzt werden, daß nur die im Code vorkommenden Bitmuster an den Eingängen des Netzwerkes auftreten können. Geben Sie zu die von Ihnen gefundenen minimalen Funktionen eine Gatterrealisierung an.

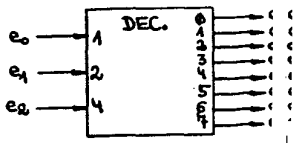
Codewandler (BCD-Code in Gray-Code):

Entwerfen Sie einen Codewandler, der einen vierstelligen BCD-Code in einen Gray-Code umwandelt. Geben Sie dazu eine mögliche Realisierung an.



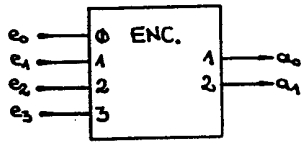
Decoder:

Geben Sie eine mögliche Realisierung für einen Decoder an, der einen dreistelligen BCD-Code in einen 1 aus n-Code umwandelt.



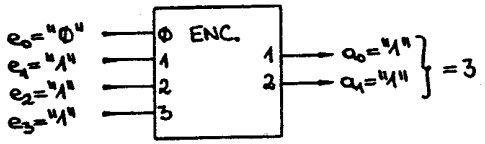
Encoder:

Entwerfen Sie einen Encoder mit vier Eingängen e_0, e_1, e_2, e_3 und zwei Ausgängen a_0 und a_1, wobei folgende Forderungen erfüllt werden sollen:



- Sind alle vier Eingänge "0", so sollen beide Ausgänge ebenfalls "0" sein.
- Ist nur ein Eingang "1" (1 aus n−Code am Eingang), so soll am Ausgang die zugehörige Dualzahl ausgegeben werden.
- Wenn mehr als ein Eingang "1" ist, so bestimmt das höchstwertigste Bit die auszugebende Dualzahl am Ausgang.

Beispiel für die Funktionsweise:

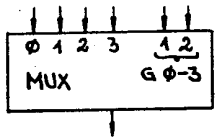


Am Ausgang soll nicht unterschieden werden können, ob alle Eingänge "0" sind, oder ob nur e_0="1" ist.

Multiplexer (Realisierung einer Wahrheitstabelle):

Eine logische Funktion f von vier Variablen a, b, c und d ist durch folgende Wahrheitstabelle gegeben:

a	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
d	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
f	1	1	1	0	0	1	1	1	0	1	0	0	0	1	0	1



Die Funktion ist mit einem *4zu1-Multiplexer* (Bild) zu realisieren. Als zusätzliche Bauteile sind nur *NOR-Gatter* erlaubt. Die Anzahl der verwendeten *NOR-Gatter* soll so klein wie möglich sein.

Rundungsfunktion:

Ein logisches Netzwerk hat vier Eingänge a, b, c, d und einen Ausgang r. An die Eingän
wird die Dualzahl angelegt, die eine Dezimalziffer Z ($0 \le Z \le 9$) repräsentiert (a ist das LSI
Andere Bitmuster können nicht auftreten. der Ausgang r soll nur "1" sein, wenn
Dezimalziffer ≥ 5 ist.
Tragen Sie die Funktion r in ein KV-Diagramm ein und geben Sie eine minimale disjunkti
und eine minimale konjunktive Form als algebraische Formel an. Don't care-Stellen sind sow
möglich bei der Minimierung auszuwerten.

Asynchroner Binärzähler mit JK-Flip-Flops:

Geben Sie die Schaltung eines asynchronen Binärzählers an, der aus *JK-Flip-Flops* aufgeb
ist und der die Dualzahlen 0−15 in der Reihenfolge 0, 1, 2, ..., 15 zyklisch durchläuft.

Synchroner Dualzähler mit Haltemodus:

Der Ausgang A eines synchronen Dualzählers mit *JK-Flip-Flops* und Steuereingang E
solange E="0" ist, zyklisch von 0 bis 5 zählen. Für E="1" soll der Zähler den jeweili
Zustand beibehalten. Wenn E von "1" auf "0" geht, soll weitergezählt werden.

Beispiel für die gewünschte Wirkungsweise:

E 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 . . .
A 0 1 2 3 4 5 0 1 2 2 2 3 4 5 0 1 2 3 . . .

Geben Sie die erforderliche Schaltung zur Realisierung dieses Zählers an.

FET

Synchrones Schaltwerk:

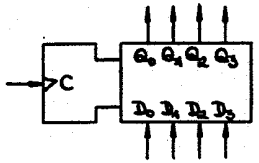
Es ist ein synchrones Schaltwerk mit einem Eingang *E* und einem Ausgang *Z* zu entwickeln. Der Eingang *E* ändert sich nur synchron mit dem Takt. Der Ausgang *Z* ist im Ruhezustand "0". Er wird immer dann für eine Taktperiode "1", wenn das Eingangssignal mehr als zwei unmittelbar aufeinanderfolgende "1"en aufweist.

Beispiel für die gewünschte Wirkungsweise:

E 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 . . .
Z 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 . . .

Eine Anfangsinitialisierung braucht nicht vorgesehen zu werden.

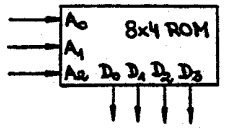
Das Schaltwerk ist aus einem *4fach D-Flip-Flop* und einem *8×4Bit-ROM* ohne zusätzliche Gatter aufzubauen. Es ist die Schaltung und der ROM-Inhalt anzugeben.



Außerdem ist die maximale Taktfrequenz zu ermitteln, mit der die Schaltung betrieben werden kann, wobei folgende Daten zu Grunde gelegt werden:

D-Flip-Flop: max. Taktfrequenz *20 MHz*
Verzögerungszeit *D* bis zum Ausgang *40 ns*
Vorbereitungszeit *15 ns*

ROM: Zugriffszeit Adreßeingang bis Datenausgang: *60 ns*



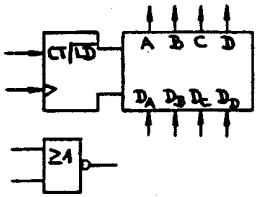
Synchrones Schaltwerk mit Zähler:

Entwickeln Sie ein synchrones Schaltwerk mit einem Eingang *RESET* und einem Ausgang *F*. Für *RESET*="0" soll der Ausgang das Bitmuster "110111" periodisch wiederholt werden. Für *RESET*="1" soll der Ausgang sofort "0" werden. Wird *RESET* wieder "0", so soll die Periode von Anfang an beginnen. *RESET* ändert sich nur synchron mit dem Takt.

Beispiel für die gewünschte Wirkungsweise:

RESET . . . 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 . . .
F . . . 1 0 1 1 1 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 1 1 . . .

Das Schaltwerk ist mit einem *ladbaren Zähler* und *NOR-Gattern* mit zwei Eingängen aufzubauen. Für die Funktion *F* und die logische Funktion, mit den der *CT/$\overline{LD}$*-Eingang beschaltet wird, sind die Wahrheitstabellen anzugeben. Die Gesamtschaltung ist aufzuzeichnen.



FET

KNF und DNF einer Funktion:

Wahrheitstabelle:

a	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1
b≠c	0	0	1	1	1	1	0	0
y=av(b≠c)	0	1	1	1	1	1	0	1

Konjunktive Normalform (KNF):

$y_{KNF} = y_{190} = (a \vee b \vee c) \& (a \vee \bar{b} \vee \bar{c})$

Disjunktive Normalform (DNF):

$y_{DNF} = y_{190} = (a \& \bar{b} \& \bar{c}) \vee (\bar{a} \& b \& \bar{c}) \vee (a \& b \& \bar{c}) \vee (\bar{a} \& \bar{b} \& c) \vee (a \& \bar{b} \& c) \vee (a \& b \& c)$

Aus dem KV-Diagramm würde sich für die minimale disjunktive Form $y = av(b\&\bar{c}) \vee (\bar{b}\&c)$, minimale konjunktive Form $y = y_{KNF} = (a \vee b \vee c) \& (a \vee \bar{b} \vee \bar{c})$ ergeben.

Logischer Ausdruck und Wahrheitstabelle einer Schaltung:

Durch die Schaltung wird die logische Funktion

$$y=[\overline{(a \vee b)} \vee a] \,\&\, [\overline{(a \vee b)} \vee b] \,\&\, c$$

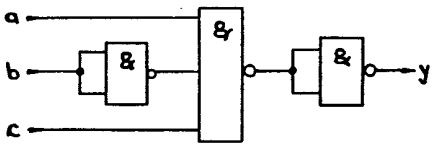
realisiert.

Wahrheitstabelle:

a	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1
$\overline{a \vee b}$	1	0	0	0	1	0	0	0
$\overline{(a \vee b)} \vee a$	1	1	0	1	1	1	0	1
$\overline{(a \vee b)} \vee b$	1	0	1	1	1	0	1	1
$\overline{\overline{(a \vee b)} \vee b}$	0	1	0	0	0	1	0	0
y	0	0	0	0	0	1	0	0

Aus der Wahrheitstabelle ist ersichtlich, daß durch die Schaltung die Vollkonjunktion $y=K_5=a\,\&\,\bar{b}\,\&\,c$ realisiert wird.

Realisierung ausschließlich mit NAND-Gatter:



$$y=\overline{\overline{\overline{a\,\&\,\bar{b}\,\&\,c}}}$$

Äquivalenzüberprüfung:

Funktion f_1: $f_1=y_1 \vee y_2 \vee y_3 \vee y_4$ mit $y_1=\bar{a}\,\&\,\bar{b}\,\&\,\bar{c}$, $y_2=a\,\&\,\bar{b}\,\&\,c$

$y_3=a\,\&\,b\,\&\,\bar{c}$, $y_4=\bar{a}\,\&\,b\,\&\,c$

a	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
d	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y_1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
y_2	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
y_3	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
y_4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
f_1	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0

Funktion f_2: $f_2=z_1 \vee z_2 \vee z_3$ mit $z_1=\bar{a}\,\&\,\bar{b}$, $z_2=\bar{a}\,\&\,\bar{c}$, $z_3=$

a	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
d	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
z_1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
z_2	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
z_3	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
f_2	1	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0

Def.: Zwei Ausdrücke heißen äquivalent, wenn sie für alle möglichen Kombinationen der Variablenwerte denselben Wert ergeben.

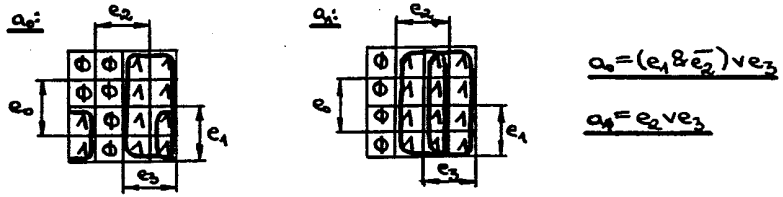
Die oben genannte Bedingung ist bei f_1 und f_2 offensichtlich nicht erfüllt, d.h. f_1 und f_2 sind nicht äquivalent.

Encoder:

Wahrheitstabelle:

e_0	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1
e_1	Φ	Φ	1	1	Φ	Φ	1	1	Φ	Φ	1	1	Φ	Φ	1	1
e_2	Φ	Φ	Φ	Φ	1	1	1	1	Φ	Φ	Φ	Φ	1	1	1	1
e_3	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	1	1	1	1	1	1	1	1
a_0	Φ	Φ	1	1	Φ	Φ	Φ	Φ	1	1	1	1	1	1	1	1
a_1	Φ	Φ	Φ	Φ	1	1	1	1	1	1	1	1	1	1	1	1

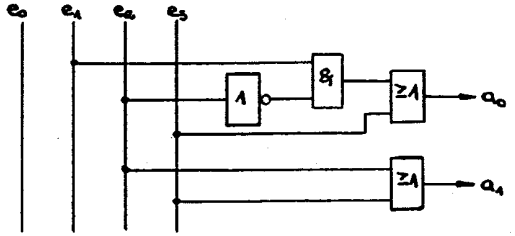
KV-Diagramm:



$a_0 = (e_1 \& \bar{e}_2) \vee e_3$

$a_1 = e_2 \vee e_3$

Realisierung:

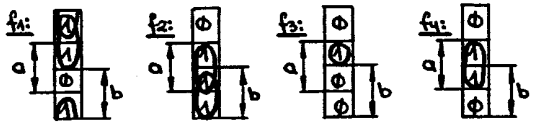


Multiplexer (Realisierung einer Wahrheitstabelle):

Wahrheitstabelle:

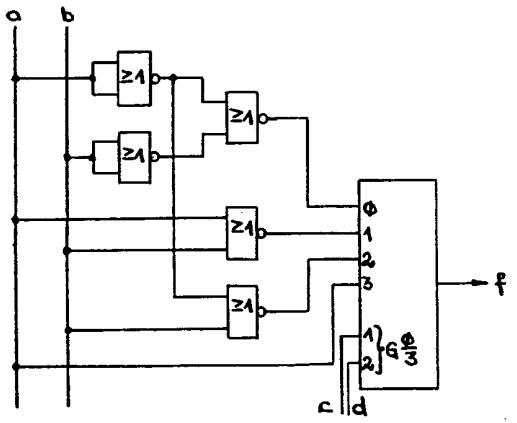
a	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1	Φ	1
b	Φ	Φ	1	1	Φ	Φ	1	1	Φ	Φ	1	1	Φ	Φ	1	1
c	Φ	Φ	Φ	Φ	1	1	1	1	Φ	Φ	Φ	Φ	1	1	1	1
d	Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	1	1	1	1	1	1	1	1
f	1	1	1	Φ	Φ	1	1	1	Φ	1	Φ	Φ	Φ	1	Φ	1
		f_1				f_2				f_3				f_4		

KV-Diagramme:



$f_1 = \bar{a} \vee \bar{b} = \overline{\bar{a} \vee \bar{b}}$ $f_2 = a \vee b = \overline{\overline{a \vee b}}$ $f_3 = a \& \bar{b} = \overline{\bar{a} \vee b}$ $f_4 = a$

Realisierung nur mit NOR-Gatter:

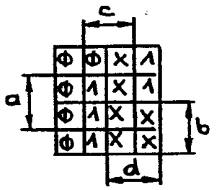


Rundungsfunktion:

Wahrheitstabelle:

Dezimalziffer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
c	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
d	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
r	0	0	0	0	0	1	1	1	1	1	X	X	X	X	X	X

KV-Diagramm:



Minimale disjunktive Form:

r=(a&c)v(b&c)vd

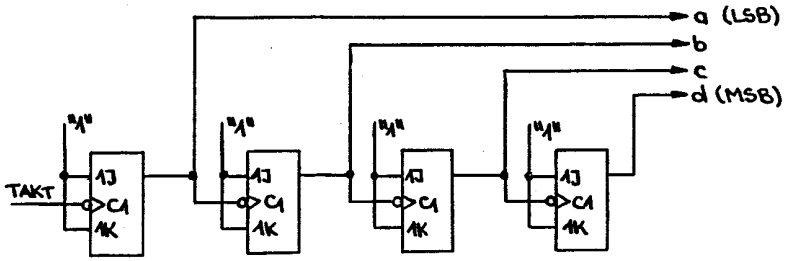
Minimale konjunktive Form:

r=(avbvd)&(cvd)

FET

Asynchroner Binärzähler mit JK-Flip-Flops:

Um die Zählerausgangszustände 0-15 realisieren zu kön
werden 4 Bit, d.h. 4 JK-Flip-Flops, benötigt.

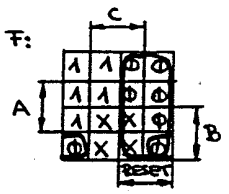


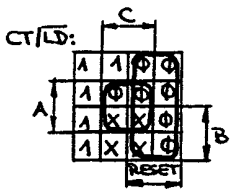
Synchrones Schaltwerk mit Zähler:

Wahrheitstabelle:

Zählerstand	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
A	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
B	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
C	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
F	1	1	0	1	1	1	X	X	0	0	0	0	0	0	X	X
CT/$\overline{LD}$	1	1	1	1	1	0	X	X	0	0	0	0	0	0	X	X

KV-Diagramme:





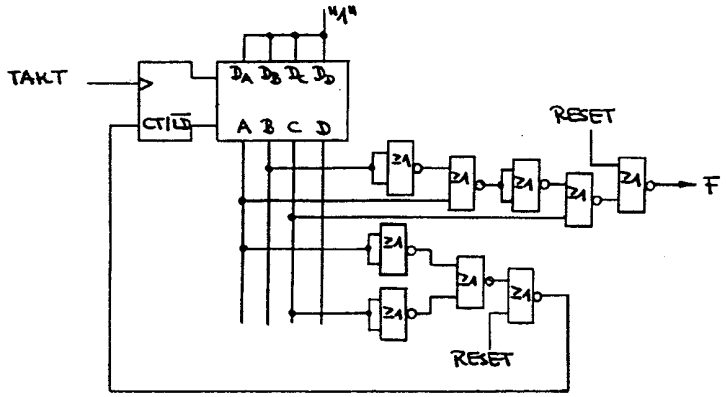
$F = (A \vee \overline{B} \vee C) \,\&\, \overline{RESET}$

$\overline{F} = \overline{(A \vee \overline{B} \vee C)} \vee RESET$

$CT/\overline{LD} = (\overline{A} \vee \overline{C}) \,\&\, \overline{RESET}$

$\overline{CT/\overline{LD}} = \overline{(\overline{A} \vee \overline{C})} \vee RESET$

Realisierung nur mit NOR-Gattern:



Digitale Systeme 28.6.90 Ⓐ Seite 1/2 **4**

Bsp 1: (15%)

Ein logisches Netzwerk hat vier Eingänge A, B, C, D und einen Ausgang R. An die Eingänge wird eine Dualzahl angelegt, die eine Dezimalziffer Z ($0 \leq Z \leq 9$) repräsentiert (A ist die Einerstelle). Andere Bitmuster können nicht auftreten. Der Ausgang R soll nur "1" sein, wenn die Dezimalziffer größergleich fünf ist.

Tragen Sie die Funktion R in das KV-Diagramm ein und geben Sie eine minimale disjunktive und eine minimale konjunktive Form als algebraische Formel an. "Don't Care"-Stellen sind soweit möglich bei der Minimierung auszunützen.

Bsp 2: (30%)

Entwerfen Sie ein synchrones Schaltwerk mit einem Eingang E und einem Ausgang A, das die Eingangsfolge 101 erkennt und mit dem letzten Einser beginnend zwei Einser ausgibt. Erkannte Folgen können unmittelbar aufeinanderfolgen, dürfen sich aber nicht überlappen. Eine Anfangsinitialisierung der Schaltung braucht nicht vorgesehen zu werden.

Beispiel:

E	0	0	0	1	0	1	0	1	0	1	1	0	1	0	0	1	0	1	0	1	0	0		
A	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	0	0	1	1	0	0	0		

Das Schaltwerk ist aus einem Baustein mit vier D-Flip-Flops und einem 16x4 ROM aufzubauen (andere Bausteine dürfen nicht verwendet werden)

3. Es werden von 4 Geräten die Daten „Gerät eingeschaltet Bzw. „Gerät ausgeschaltet" 255 x eingelesen

Abspeicherung: je L Abfragelöcke pro Byte (→ 128 Byte):

20.11.95 (A)

G0	G1	G2	G3	G0	G1	G2	G3

ie ... „0": Gerät eingeschaltet
„1": Gerät ausgeschaltet

Die Abspeicherung in diesem Format und das Einlesen der Zustände erfolgt im Hauptprogramm.

Gesucht: Algorithmus „Test":

Ermittelt die Anzahl der Abfragelöcke, die „1 oder mehrere Geräte eingeschaltet" anzeigen.
Anzahl bei Ende im B-Register

a) Struktogramm

b) Programmierung in MC-8 - Assembler

4. - Blockschaltbild der MC8 - CPU zeichnen
- Erkläre die Funktion des Stack-Pointers

5. Lesen eines Programms

Programm verändert sich selbst

→ verschieden Zahlendarstellung: hex, binär, dez

- Es muß ein paar mal zwischen hex, binär Bzw. dez umgewandelt werden
- Bitmuster muß als Befehl interpretiert werden

Gesucht: Tabelle ist auszufüllen:

Befehl	PC vorher	PC nachher	A nachher	B nachher	Kommentar