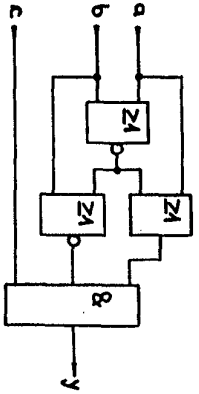


KNF und DNF einer Funktion:

Geben Sie für die Funktion $a \vee (b \neq c)$ 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

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

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

äquivalent sind.

KNF und minimale DF einer logischen Funktion:

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

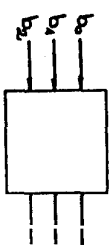
a	0	1	0	1	0	1	0	1	0	1	0	1	0	1
b	0	0	1	1	0	0	1	1	0	0	1	1	0	0
c	0	0	0	0	1	1	1	1	0	0	0	1	1	1
d	0	0	0	0	0	0	0	0	1	1	1	1	1	1
y	1	0	1	0	1	1	1	0	1	0	1	1	1	0

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		b_2	b_1	b_0
0	0	0	0	→	1	0	1
1	0	0	1		1	1	0
2	0	1	0		1	1	1
3	0	1	1		0	0	0
4	1	0	0		0	0	1
5	1	0	1		0	1	0

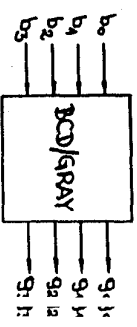


Entwerfen Sie ein logisches Netzwerk, das den Code B in den Code A umwandelt. In den notwendigen logischen Funktionen sind mit Hilfe des KV-Diagramms in ihrer minimalen disjunktiven Form darzustellen. Es kann dazu vorausgesetzt werden, daß nur die im Code vorkommenden Bitmuster an den Eingängen des Netzwerkes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

Codewandler (BCD-Code in Gray-Code):

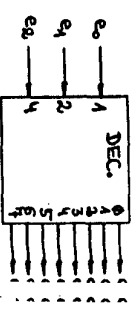
Entwerfen Sie einen Codewandler, der einen vierstelligen BCD-Code in einen Gray-Code umwandelt.

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.

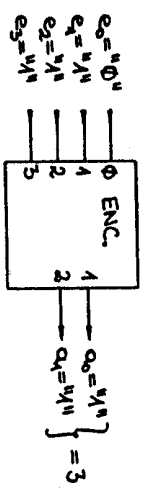
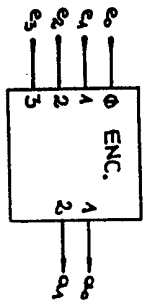


~~FET~~

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.



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

Rundungsfunktion:

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

Asynchroner Binärzähler mit JK-Flip-Flops:

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

Synchroner Dualzähler mit Haltemodus:

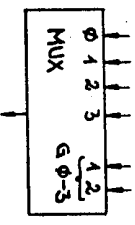
Der Ausgang A eines synchronen Dualzählers mit JK-Flip-Flops und Steuereingang E solange $E = "0"$ ist, zyklisch von 0 bis 5 zählen. Für $E = "1"$ soll der Zähler den jeweiligen Zustand beibehalten. Wenn E von "1" auf "0" geht, soll weitergezählt werden.

Beispiel für die gewünschte Wirkungsweise:
 E 0 0 0 0 0 0 0 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.

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	0	0	0	1	1	1	1	1	1
d	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
f	1	1	1	1	0	0	1	1	1	0	1	0	0	1	1	0



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.



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 unmittlbar aufeinanderfolgende "1"en aufweist.

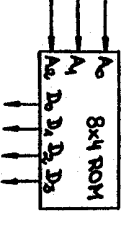
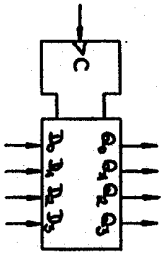
Beispiel für die gewünschte Wirkungsweise:
E 0 0 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
Z 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

Eine Anfangsinitialisierung braucht nicht vorgesehen zu werden.

Das Schaltwerk ist aus einem 4fach D-Flip-Flop und einem 8x4Bit-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 40ns
- Vorbereitungszeit 15ns
- ROM: Zugriffszeit Adreßeingang bis Datenausgang: 60ns

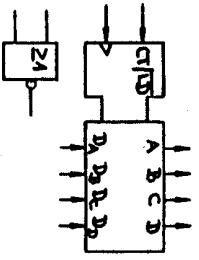


Synchrones Schaltwerk mit Zähler:

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 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
F 1 0 1 1 1 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 1 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/D-Eingang beschaltet wird, sind die Wahrheitstabellen anzugeben. Die Gesamtschaltung ist aufzuzeichnen.



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	1	1	1	1	1
y = a ∨ (b#c)	0	1	1	1	1	1	1	1

Konjunktive Normalform (KNF):

$y_{KNF} = y_{100} = (a \vee b \vee c) \& (a \vee b \vee \bar{c})$

Disjunktive Normalform (DNF):

$y_{DNF} = y_{100} = (a \& b \& \bar{c}) \vee (\bar{a} \& b \& \bar{c}) \vee (a \& b \& c)$

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

Aus dem KV-Diagramm würde sich für die

minimale disjunktive Form $y = a \vee (b \& \bar{c}) \vee (b \& \bar{c})$

minimale konjunktive Form $y = y_{100} = (a \vee b \vee c) \& (a \vee b \vee \bar{c})$

ergeben.

Logischer Ausdruck und Wahrheitstabelle einer Schaltung:

Durch die Schaltung wird die logische Funktion

$$y = \overline{[(\overline{a}b)va]g}[(\overline{a}b)vb]gc$$

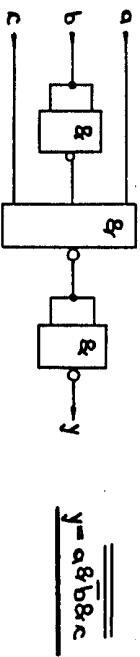
realisiert.

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
$\overline{a}b$	1	0	0	0	1	0	0	0
$(\overline{a}b)va$	1	1	0	1	1	1	0	1
$(\overline{a}b)vb$	1	0	1	1	1	0	1	1
$(\overline{a}b)vb$	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 $y = K_8 = a\overline{b}bc$ realisiert wird.

Realisierung ausschließlich mit NAND-Gatter:



$$y = a\overline{b}bc$$

Äquivalenzüberprüfung:

Funktion f_1 : $f_1 = y_1 v y_2 v y_3 v y_4$ mit $y_1 = \overline{a}b\overline{c}$, $y_2 = a\overline{b}bc$

$$y_3 = a\overline{b}bc, \quad y_4 = \overline{a}b\overline{c}$$

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

Funktion f_2 : $f_2 = x_1 v x_2 v x_3$ mit $x_1 = \overline{a}b\overline{c}$, $x_2 = \overline{a}b\overline{c}$, $x_3 =$

a	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
c	0	0	0	1	1	1	0	0	0	1	1	1
d	0	0	0	0	0	0	1	1	1	1	1	1
x_1	1	0	0	0	1	0	0	1	0	0	1	0
x_2	1	0	1	0	0	0	0	1	0	1	0	0
x_3	1	1	0	0	0	0	1	1	0	0	0	0
f_2	1	1	1	0	1	0	0	1	1	1	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.

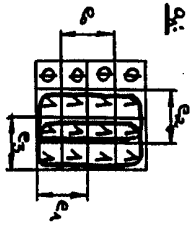
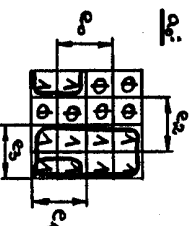


Encoder:

Wahrheitstabelle:

e_0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
e_1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
e_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1
e_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1
a_0	0	0	1	1	0	0	0	0	1	1	1	1	1	1
a_1	0	0	0	0	1	1	1	1	0	0	0	0	1	1

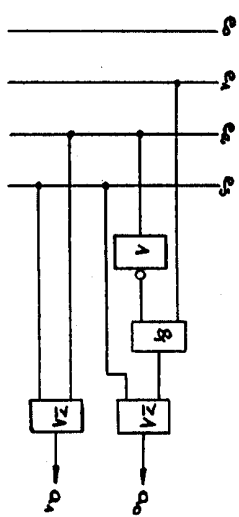
KV-Diagramm:



$$a_0 = (e_1 \& e_2) \vee e_3$$

$$a_1 = e_2 \vee e_3$$

Realisierung:

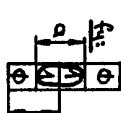
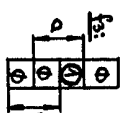
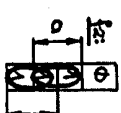
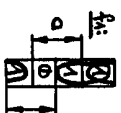


Multiplexer (Realisierung einer Wahrheitstabelle):

Wahrheitstabelle:

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

KV-Diagramm:



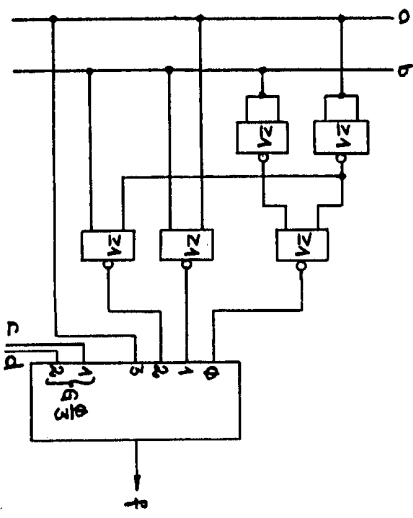
$$f_1 = \bar{a} \vee b = \overline{a \bar{b}}$$

$$f_2 = a \vee \bar{b} = \overline{a \bar{b}}$$

$$f_3 = a \& b = \overline{a \bar{b}}$$

$$f_4 = a$$

Realisierung nur mit NOR-Gatter:



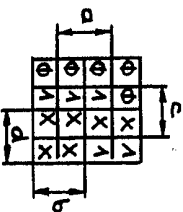
FET

Rundungsfunktion:

Wahrheitstabelle:

Dezimalwert	a	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	1	1	1	1	1	1	1	1	1
c	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

KV-Diagramm:



Minimale disjunktive Form:

$$c = (a \& b) \vee (b \& c) \vee d$$

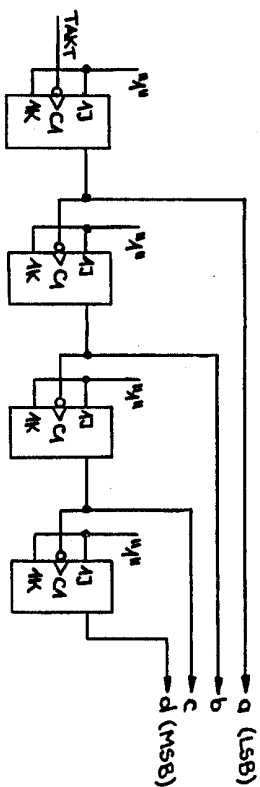
Minimale konjunktive Form:

$$c = (a \vee b \vee d) \& (c \vee d)$$

FET

Asynchroner Binärzähler mit JK-Flip-Flops:

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

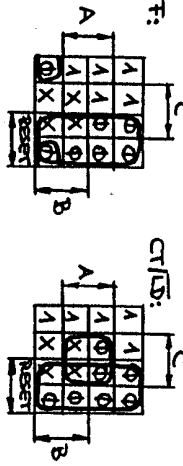


Synchrones Schaltwerk mit Zähler:

Wahrheitstabelle:

Zählerstand	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
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
F	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1
CT/D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

KN-Diagramme:



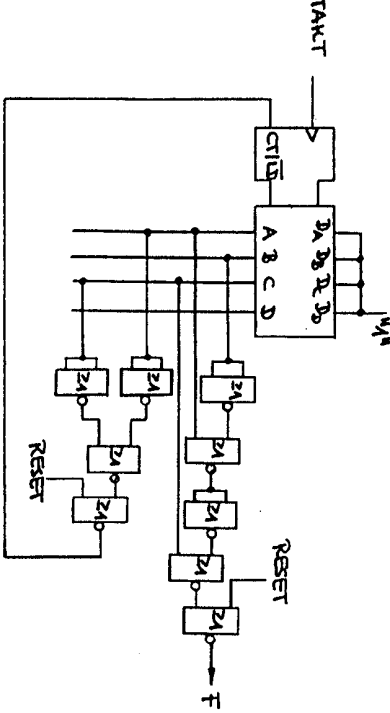
$F = (A \vee \bar{B} \vee C) \wedge \overline{\text{RESET}}$

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

$\text{CT/D} = (A \vee C) \wedge \overline{\text{RESET}}$

$\text{CT/D} = (A \vee C) \vee \text{RESET}$

Realisierung nur mit NOR-Gattern:



FET
K

Bsp 1: (15%)

Ein logisches Netzwerk hat vier Eingänge A, B, C, D und einen Ausgang R . In 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.

Gegeben: So die Funktion R in das KV-Diagramm eintragen. So eine minimale Disjunktive und eine minimale konjunktive Form als algebraischer Ausdruck angeben. Die "1" Stellen sind soweit möglich in die Minimierung einzusetzen.

Bsp 2: (10%)

Entwerfen Sie ein synchrones Schaltwerk mit einem Eingang E und einem Ausgang A , das die Eingangsfolge 1010101001010100 liest und mit dem letzten Einsen beginnend zwei Einsen zurück die bekannte Folge liest. Die Einsen dürfen sich aber nicht überlappen. Eine Anfangsinitialisierung der Schaltung braucht nicht vorgesehen zu werden.

Beispiel:

E	0	0	0	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	0	1	0	0
A	0	0	0	0	0	1	1	0	0	1	1	0	0	0

Das Schaltwerk ist aus einem Baustein mit vier D-Flip-Flops und einem 16×4 ROM aufzubauen (andere Bausteine dürfen nicht verwendet werden)

