Systemorientierte Informatik 1

2. Grundlagen Digitaler Schaltungen
   - 2.10 Minimierung Boole‘scher Funktionen
   - 2.11 CMOS Komplexgatter

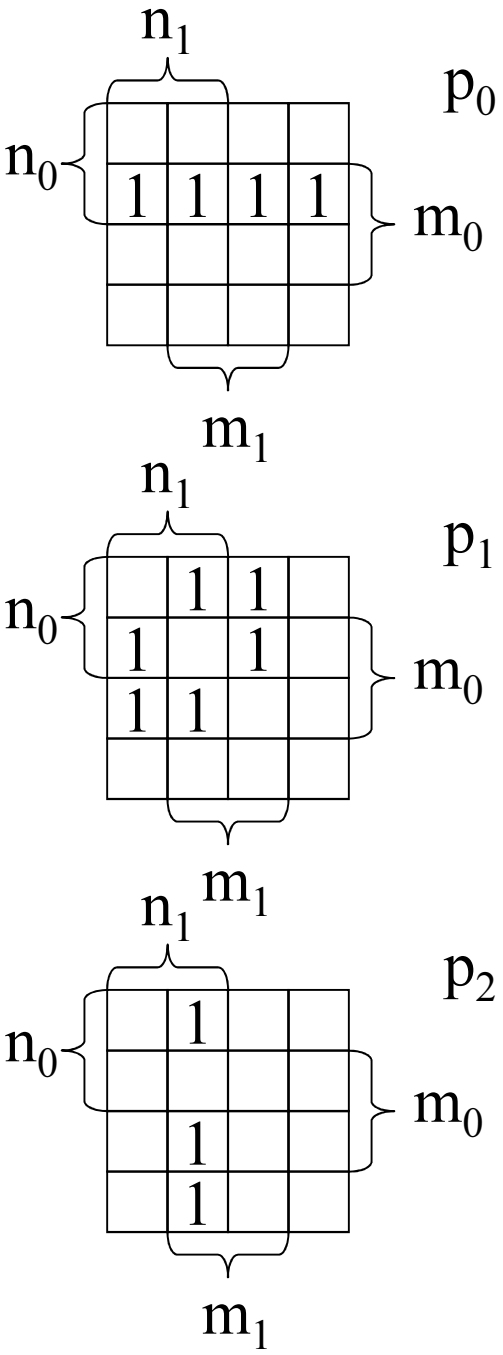
Die Einsen im KV-Diagramm werden zu Blöcken maximaler Größe zusammengefasst. Dabei müssen die Blöcke immer im Raster der Zweierpotenzen beginnen und enden. Eine Zusammenfassung von zwei Blöcken zu einem Block doppelter Größe entspricht einer Anwendung der Vereinfachungsregel: Wenn ein Block x_0x_1 und ein zweiter Block $x_0\neg x_1$ beide nur aus Einsen bestehen, liegen diese beiden Blöcke im KV-Diagramm nebeneinander und können zu einem doppelt so großen Block x_0 zusammengefasst werden. Die gegenüberliegenden Ränder eines KV-Diagramms sind zu identifizieren. Man kann sich das Diagramm als Torus vorstellen.

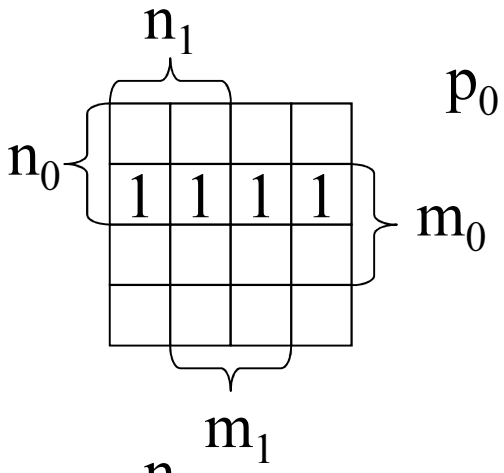
Wenn mehr als 4 Eingabevariablen vorliegen, muss ein 2-dimensionales KV-Diagramm so dargestellt werden, dass einzelne Variablen Bereiche überdecken, die nicht zusammenhängend in der Ebene sind. Dabei sind aber die zueinander zeigenden Ränder dieser Bereiche als identisch anzusehen.

Einträge in KV-Diagrammen können 0en und 1en sein. In diesem Fall kann man durch Zusammenfassen aller 1en zu maximalen Blöcken eine DMF ablesen. (Leider ist sie nicht eindeutig). In solchen Fällen schreibt man meist nur die Einsen in das Diagramm und lässt die 0en weg. Zusammenfassen der Nullen und benutzen der Komplemente der Variablen führt zur KMF.
Wenn einzelne Elemente in der Wertetabelle don‘t cares sind, können diese in den Blöcken der Einsen bei der DMF (oder Nullen bei der KMF) mit auftauchen. Sie schaden nichts. Aber es müssen durchaus nicht alle don‘t cares (dargestellt durch den Buchstaben X oder d) mit in Blöcke aufgenommen werden.

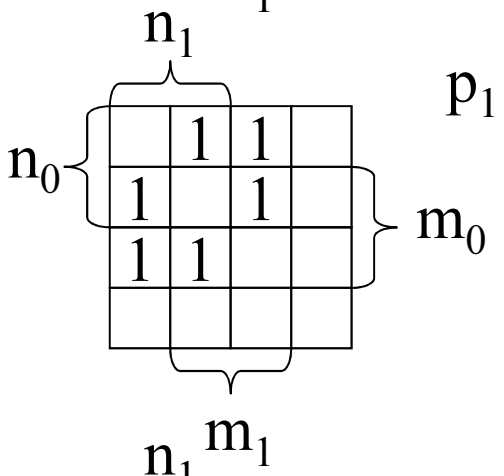
2-Bit Multiplizierer:

n_1	n_0	m_1	m_0	p_3	p_2	p_1	p_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1



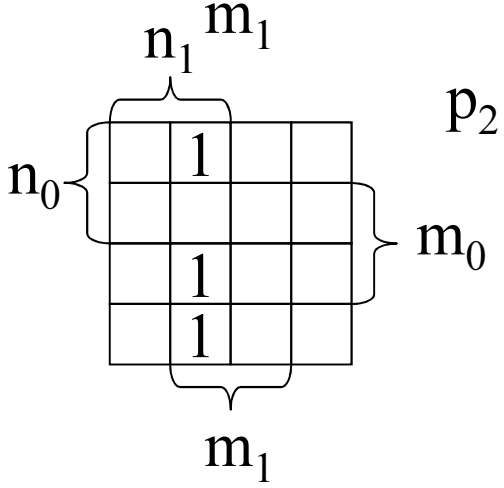


$$p_0 = n_0 \cdot m_0$$



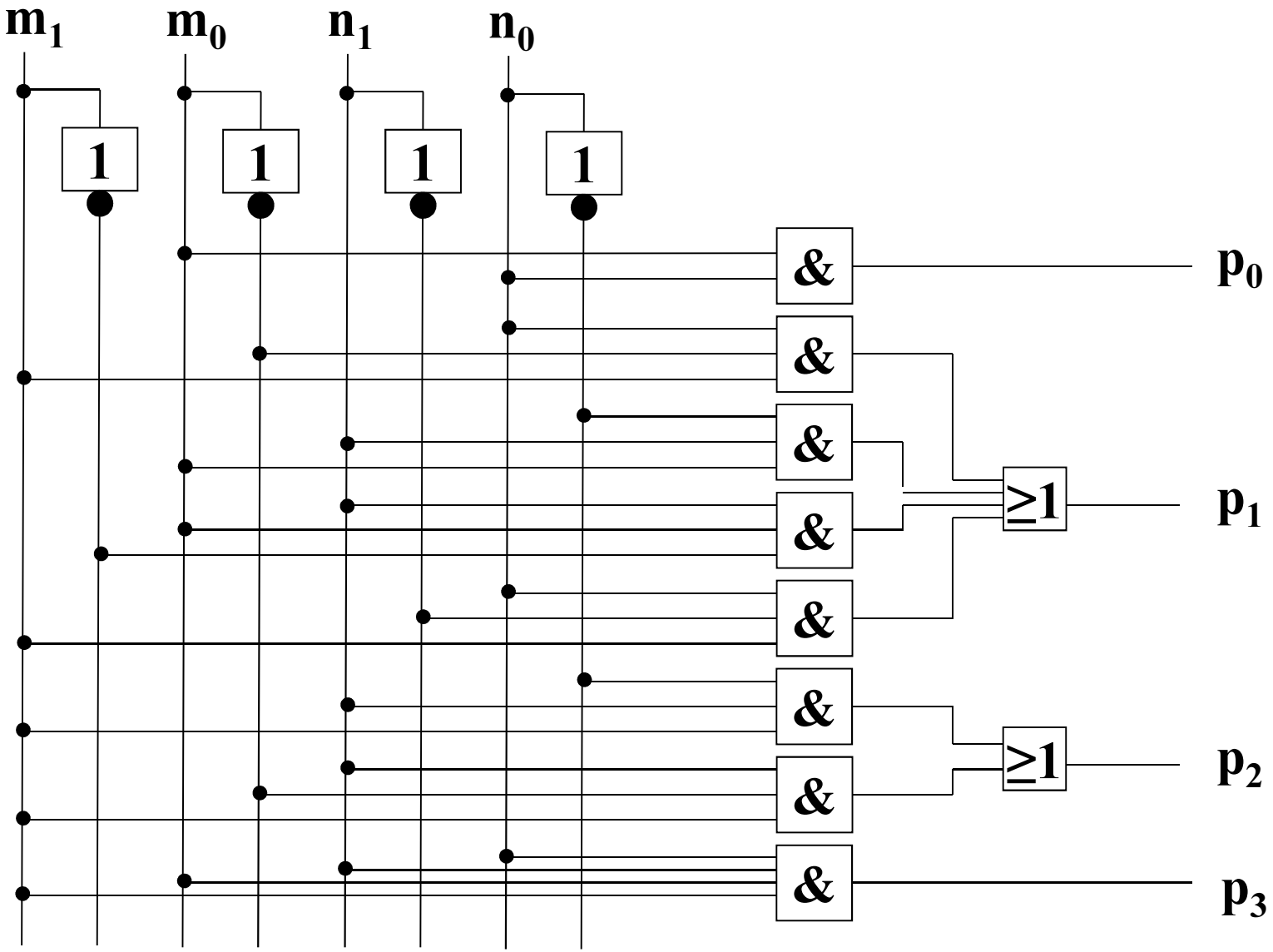
$$p_1 = n_0 \cdot \overline{m_0} \cdot m_1 + \overline{n_0} \cdot n_1 \cdot m_0 + n_1 \cdot m_0 \cdot \overline{m_1} + n_0 \cdot \overline{n_1} \cdot m_1$$

$$p_2 = \overline{n_0} \cdot n_1 \cdot m_1 + n_1 \cdot \overline{m_0} \cdot m_1$$



$$p_3 = n_0 \cdot n_1 \cdot m_0 \cdot m_1$$

2-Bit-Multiplizierer



1. Aufstellen der Wertetabelle
2. Eintragen der Werte in KV-Diagramm
3. Zusammenfassen von benachbarten Einsen zu Blöcken maximaler Größe
4. Ablesen der DMF

4-Bit Codewandler: Dezimal -> Aiken

x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	1	1	0	1
1	0	0	0	1	1	1	0
1	0	0	1	1	1	1	1
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Definition:

Eine in konjuktiver Normalform angegebene Boole‘sche Funktion ist in konjunktiver Minimalform (KMF), wenn

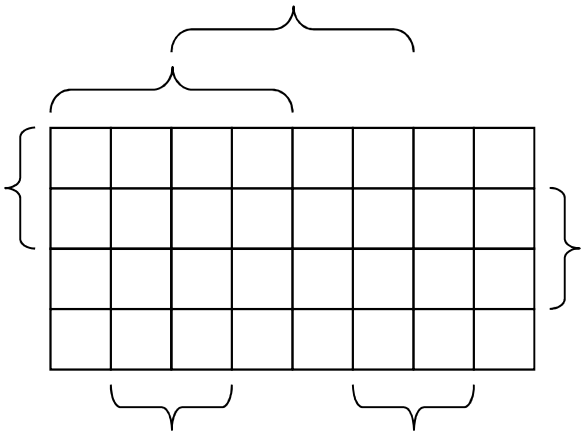
- jede äquivalente Darstellung derselben Funktion in KNF mindestens genausoviele Summenterme besitzt,

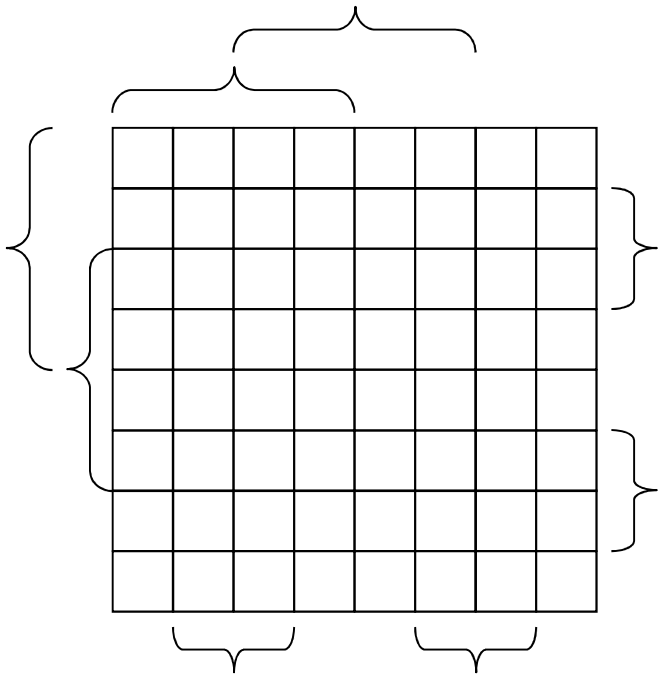
und wenn

- für jede äquivalente Darstellung in KNF mit genausovielen Summentermen die Anzahl der Eingänge in diese Summenterme mindestens genauso groß ist wie die Anzahl bei dieser Darstellung.

1. Aufstellen der Wertetabelle
2. Eintragen der Werte in KV-Diagramm
3. Zusammenfassen von benachbarten Nullen zu Blöcken maximaler Größe
4. Ablesen der KMF, indem die Summenterme gebildet werden, die diese Blöcke von Nullen nicht abdecken. Zu diesem Zweck verodert man die invertierten Eingabevariablen, die diese Blöcke von Nullen überdecken.

KV-Diagramme mit mehr als 4 Variablen





Das Verfahren von Quine und McCluskey

Mit KV-Diagrammen kommen wir nicht weiter, wenn die Anzahl der Eingabevariablen größer als 6 wird. In diesem Fall empfiehlt sich das Verfahren von Quine und McCluskey. Es beginnt mit der KDMF und besteht aus zwei Schritten:

Erstens: Das Verfahren von McCluskey erzeugt durch systematische Anwendung der Vereinfachungsregeln alle Primterme einer Funktion.

Zweitens: Das Verfahren von Quine wählt aus dieser Menge aller Primterme eine minimale Teilmenge aus, deren Oder-Verknüpfung die gesamte Funktion repräsentiert.

Definition:

Ein Primterm von f ist eine Konjunktion von Variablen, die in f erfüllt ist, für die aber keine echte Teilkonjunktion in f erfüllt ist.

Beispiel: Warnleuchte:

Z	H	P	W
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

ZH ist ein Primterm der Funktion W.

Satz:

Die disjunktive Minimalform einer Funktion ist eine Disjunktion von Primtermen.

Das Verfahren von Quine und McCluskey

1. McCluskey:

Systematische Anwendung der Regel

$$xy + x\bar{y} = x \quad \Rightarrow$$

Konstruktion aller Primterme

2. Quine

Treffen einer minimalen Auswahl von Primtermen,
deren Disjunktion die Funktion realisiert.

Einleitendes Beispiel

I II III IV V VI

$$f = abcd + a\bar{b}cd + \bar{a}bcd + abc\bar{d} + \bar{a}\bar{b}cd + \bar{a}bc\bar{d}$$

I,II I,III I,IV II,V III,V III,VI IV,VI

$$acd + bcd + abc + \bar{b}cd + \bar{a}cd + \bar{a}bc + bc\bar{d}$$

A B C D E F G

A,E B,D B,G C,F

$$cd + cd + bc + bc = cd + bc$$

Das Verfahren von McCluskey

Begonnen wird mit der Funktion in KDNF

1. Für jedes Paar von Produkttermen wird geprüft, of die Regel

$$xy + x\bar{y} = x$$

   anwendbar ist. Wenn ja, wird in der nächsten Zeile der Produktterm x aufgenommen. Alle Terme, die nicht zu einem solchen Produktterm beigetragen haben, werden unverändert in die nächste Zeile übernommen.

2. Wenn keine neuen Produktterme in der neuen Zeile entstehen, ist man fertig. Sonst wird bei 1. weitergemacht. Am Ende stehen in der letzten Zeile alle Primterme.

Zweites Beispiel

I II III IV

$$f = ab\bar{c} + abc + \bar{a}bc + \bar{a}\bar{b}c$$

I,II II,III III,IV

$$ab + bc + \bar{a}c$$

bc ist ein redundanter Term, wie man am KV-Diagramm leicht sehen kann. Daher benötigen wir das Verfahren von Quine.

	ab	bc	$\bar{a}c$
$ab\bar{c}$	1		
abc	1	1	
$\bar{a}bc$		1	1
$\bar{a}\bar{b}c$			1

Das Verfahren von Quine

Eine Primterm-Minterm-Tabelle wird aufgestellt: Die Minterme in der Zeile und die Primterme in der Spalte.

1.Alle Spalten, in denen eine 1 aus einer dominanten Zeile (Zeile mit nur einer 1) steht, werden markiert. Alle Zeilen, in denen 1en aus markierten Spalten stehen, werden gestrichen.
2.Wenn keine ungestrichene Zeile mehr vorhanden ist, wird das Verfahren beendet. Die markierten Spalten bilden die Minterme der DMF.
3. Wenn keine dominante Zeile mehr vorhanden ist, aber noch ungestrichene Zeilen existieren, wird eine Spalte mit den meisten ungestrichenen 1en markiert und bei 1. fortgefahren.

Letztes Beispiel für Quine-McCluskey

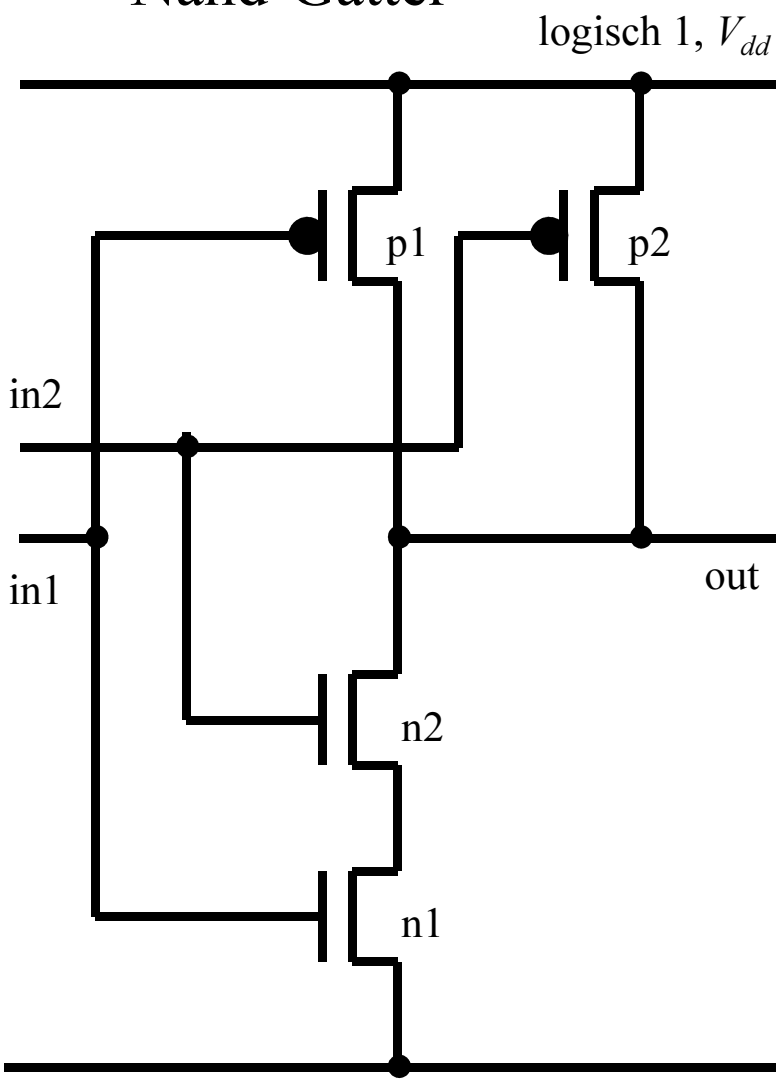
I II III IV V

$$f = a\bar{b}\bar{c}d + ab\bar{c}d + abcd + \bar{a}bcd + \bar{a}bc\bar{d}$$

2.11 CMOS Komplexgatter

Mit den einfachen Gattern wie Inverter, Nand und Nor haben wir Schaltungen in statischer CMOS-Technik gebaut. Diese ist dadurch gekennzeichnet, dass die Signalpegel über beliebig lange Zeiten erhalten bleiben, wenn keine Signalwechsel an den Eingängen auftreten. Realisiert wird das dadurch, dass alle Gates der Tansistoren über eine leitende Strecke mit jeweils einer der beiden Versorgungsspannungspotenziale verbunden sind. Schauen wir uns unter diesem Aspekt noch einmal das Nand-Gatter in statischer CMOS-Technik an:
In einem Netzwerk aus p-Transistoren wird eine leitende Strecke von der V_{dd}-Leitung zum Ausgang geschaltet, wenn die Funktion den Wert 1 haben soll. In einem zweiten (komplementären) Netzwerk aus n-Transistoren wird eine leitende Strecke von der *GND*-Leitung zum Ausgang geschaltet, wenn die Funktion den Wert 0 haben soll.

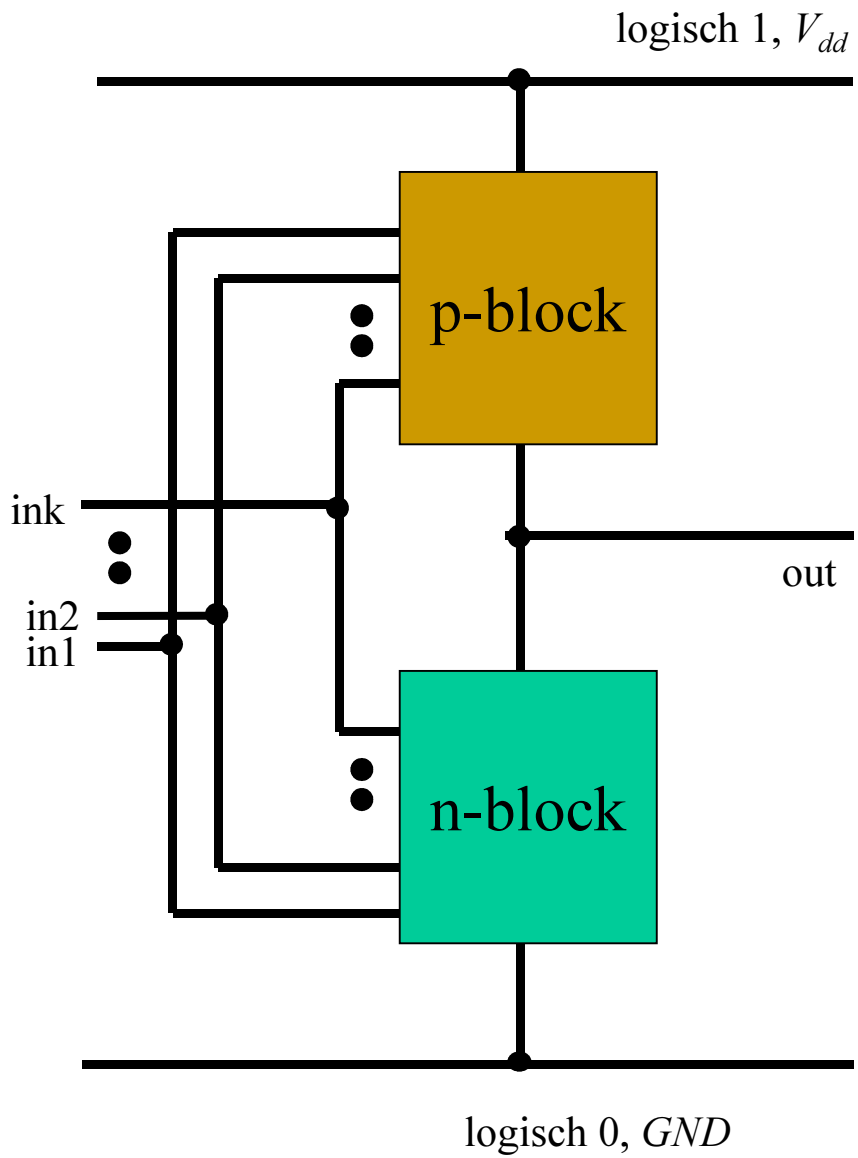
Nand-Gatter



in1	in2	p1	p2	n1	n2	out
0	0	L	L	S	S	1
0	1	L	S	S	L	1
1	0	S	L	L	S	1
1	1	S	S	L	L	0

L : leitend
S : sperrend

Mit derselben Technik kann man nun auch komplexere Schaltungen aufbauen.



Beispiel: Nehmen wir an, wir wollen die Funktion $f = (\overline{a} + b) \cdot (c + \overline{d})$ aufbauen. Mit CMOS sind solche Funktionen besonders einfach aufzubauen.

Betrachten wir zunächt den p-Block: Ein „Oder“ realisieren wir durch Parallelschaltung zweier Transistoren, ein „Und“ durch Reihenschaltung. Nun müssen wir aber beachten, dass ein p-Transistor genau bei Eingabe einer 0 durchschaltet und bei Eingabe einer 1 sperrt. Er invertiert also die Eingänge. Daher müssen wir ihn gleich mit invertierten Eingängen beschalten.

Wir schalten also zwei p-Transistoren mit a und ¬b als Gate parallel und ebenso zwei p-Transistoren mit ¬ c und d als Gate. Diese beiden Parallelschaltungen schalten wir in Reihe.

Im n-Block erzeugen wir ein Null. Wir müssen also mit dem n-Block die invertierte Funktion generieren. Dazu müssen wir die Funktionsgleichung mit den Gesetzen von de Morgan so umformen, dass eine Invertierung über der gesamten Funktion steht.

$$f = (\overline{a} + b) \cdot (c + \overline{d}) = \overline{\overline{(\overline{a} + b) \cdot (c + \overline{d})}} = \overline{\overline{(\overline{a} + b)} + \overline{(c + \overline{d})}}$$
$$= \overline{(a \cdot \overline{b}) + (\overline{c} \cdot d)}$$

Somit werden zwei n-Transistoren mit a und ¬b als Gate in Reihe geschaltet und ebenso zwei n-Transistoren mit ¬ c und d. Beide Zweige werden zueinander parallel geschaltet.

Alle Ausgänge werden an einem Punkt zusammengelegt. Die Schaltung ist auf der folgenden Folie zu sehen.

Bitte überzeugen Sie sich davon, dass

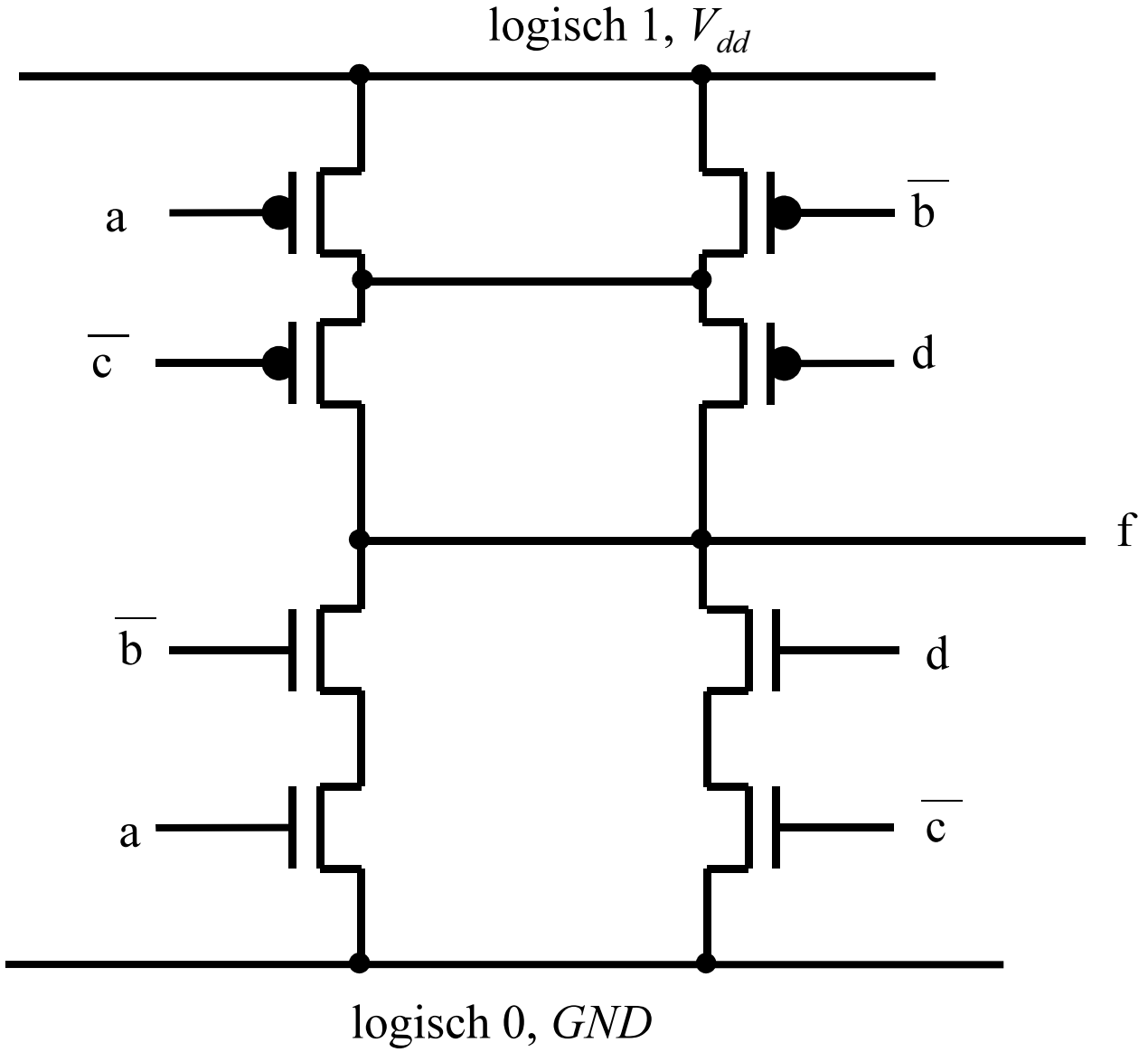
• Die Schaltung eine Realisierung der oben angegebenen Funktion darstellt.

• Bei keiner Belegung der Eingänge ein Stromfluss von V_{dd} nach *GND* entstehen kann.

• Bei jeder Belegung der Eingänge der Ausgang durch einen leitenden Pfad entweder mit V_{dd} oder mit *GND* verbunden ist.

• Am Ausgang immer „gute“ Signale anliegen.

Komplex-Gatter für f = (¬a∨b)∧(c ∨¬d)

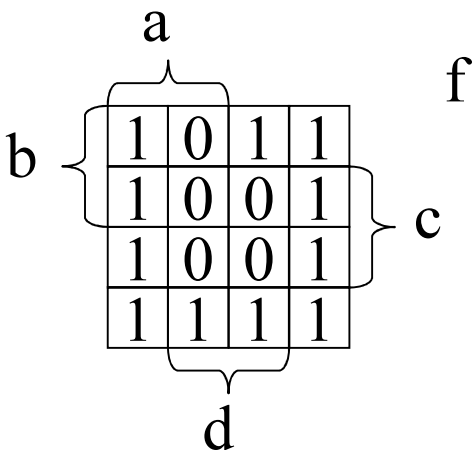


Methode zur Konstruktion von Komplexgattern:

- Erstellen der KMF und DMF
- Konstruktion des p-Blocks entsprechend der einfacheren der beiden Formen, dabei: alle Eingänge invertieren, ODER-Verknüpfung durch Parallelschaltung und UND-Verknüpfung durch Reihenschaltung
- Umwandeln in NAND bzw. NOR-Form
- Konstruktion des n-Blockes entsprechend der einfacheren der beiden Formen, dabei wieder: ODER-Verknüpfung durch Parallelschaltung und UND-Verknüpfung durch Reihenschaltung

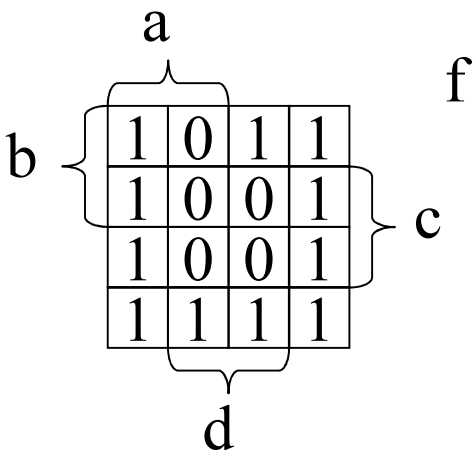
Beispiel:

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



Beispiel:

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



$$
\begin{aligned}
f &= (\overline{c} + \overline{d}) \cdot (\overline{a} + \overline{b} + \overline{d}) \\
&= (\overline{c \cdot (\overline{a} + \overline{b})}) + \overline{d} \\
&= \overline{\overline{(\overline{c \cdot (\overline{a} + \overline{b})}) + \overline{d}}} \\
&= \overline{\overline{(\overline{c \cdot (\overline{a} + \overline{b})})} \cdot d} \\
&= \overline{(c + \overline{\overline{(\overline{a} + \overline{b})}}) \cdot d} \\
&= \overline{(c + (a \cdot b)) \cdot d}
\end{aligned}
$$

Komplex-Gatter für f = (¬c∧(¬a∨¬b))∨¬d

$$
\begin{aligned}
f &= (\overline{c} + \overline{d}) \cdot (\overline{a} + \overline{b} + \overline{d}) \\
&= (\overline{c} \cdot (\overline{a} + \overline{b})) + \overline{d} \\
&= \overline{\overline{(\overline{c} \cdot (\overline{a} + \overline{b})) + \overline{d}}} \\
&= \overline{\overline{(\overline{c} \cdot (\overline{a} + \overline{b}))} \cdot d} \\
&= \overline{(c + (\overline{a} + \overline{b})) \cdot d} \\
&= \overline{(c + (a \cdot b)) \cdot d}
\end{aligned}
$$