3. Kombinatorische Grundschaltungen

3.1 Gatter

3.1.1 Die vollständige Realisierungsbasis

Welche elementaren Schaltfunktionen sind notwendig, um mit einer Anzahl entsprechender Schaltelemente (Gatter) jede beliebige Schaltfunktion darstellen zu können? – Jeder Satz elementarer Schaltfunktionen, der dies ermöglicht, heißt eine vollständige Realisierungsbasis.

Sicher ist, dass man mit den drei Funktionen UND, ODER, NICHT auskommt. Ob eine beliebige andere Sammlung von Funktionen eine vollständige Realisierungsbasis bildet oder nicht, kann man überprüfen, indem man versucht, diese drei Elementarfunktionen damit aufzubauen. UND, ODER und NICHT sind auch schaltungstechnisch einfach zu verwirklichen. Demgegenüber wäre es unpraktisch, Antivalenz- oder Äquivalenzverknüpfungen in die Realisierungsbasis aufzunehmen, denn diese Verknüpfungen erfordern Wechselschaltungen, oder sie müssen ihrerseits auf UND, ODER und NICHT zurückgeführt werden (zu lange Verzögerungszeiten, zu kompliziert). Zwei weitere Elementarfunktionen sind aber jeweils für sich allein als vollständige Realisierungsbasis ausreichend: NAND und NOR. Auch diese Verknüpfungen sind schaltungstechnisch einfach zu verwirklichen. So ergibt sich die NAND-Verknüpfung durch Reihenschaltung und die NOR-Verknüpfung durch Parallelschaltung von Transistorstufen.

Die Realisierungsbasis in der Entwurfspraxis
Sie ist dann von Bedeutung, wenn die elementaren Gatter als Bauelemente oder Funktionseinheiten einzusetzen sind (Digitalschaltkreise niedrigen Integrationsgrades oder diskrete Schaltungslösungen). Beim Einsatz programmierbarer oder anwendungsspezifischer Schaltkreise spielt die Realisierungsbasis keine Rolle, da die Entwurfsunterstützungssysteme ein umfangreiches Sortiment an Gatterfunktionen bereitstellen, so dass es kaum noch erforderlich sein dürfte, eine Entwurfsabsicht von Hand auf ein bestimmtes eingeschränktes Gattersortiment umzusetzen. Trotzdem schadet es nicht, das Problem zu kennen und gelegentlich mitzudenken (vor allem dann, wenn der Entwurf unerklärlicherweise nicht in den ausgewählten Schaltkreis passen will oder wenn sich zu lange Verzögerungszeiten ergeben).

3.1.2 Funktionselemente (Grundgatter)

Grundgatter (Basic Gates) sind die einfachsten Funktionselemente. Sie sind in Gatterschaltkreisen und als elementare Zellenstrukturen in programmierbaren und anwendungsspezifischen Schaltkreisen verfügbar. Kennzeichnende funktionelle Merkmale sind die logische Verknüpfung und die Anzahl der Eingänge. Zu den technischen Merkmalen gehören u. a. die Verzögerungszeiten, die Speisespannung, die Stromaufnahme und die Lastkennwerte der Ein- und Ausgänge.

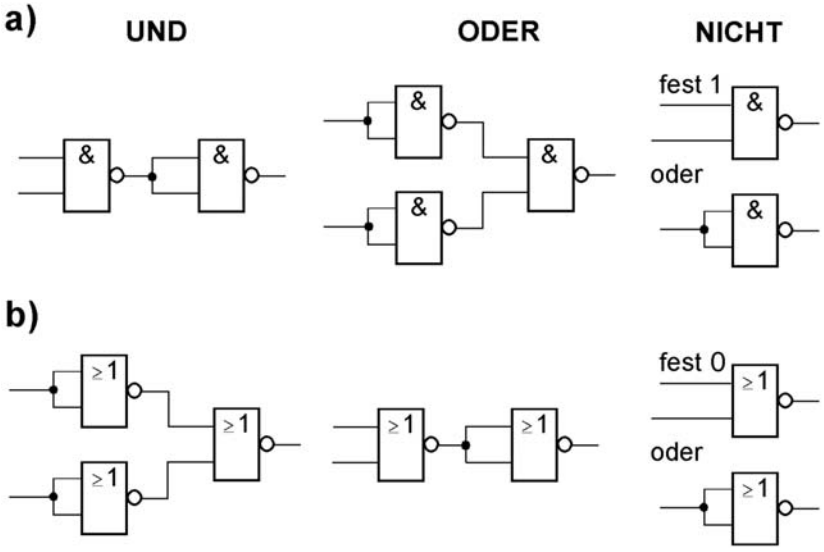


Abb. 3.1 Elementare aussagenlogische Verknüpfungen mit NAND und NOR. a) vollständige Realisierungsbasis UND, ODER, NICHT mit NAND, b) vollständige Realisierungsbasis UND, ODER, NICHT mit NOR.

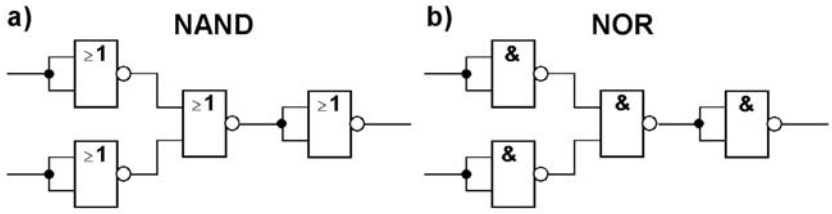


Abb. 3.2 Wechselseitige Wandlungen. a) NAND aus NOR; b) NOR aus NAND.

Welche Gattertypen werden als Funktionselemente bereitgestellt?

Das hängt von der Schaltkreistechnologie ab. Es richtet sich vor allem danach, wie die einfachsten Schaltungen aussehen, mit denen sich brauchbare Gatterfunktionen darstellen lassen. Gatterschaltungen in modernen integrierten Schaltkreisen sind Transistorschaltungen (gleichgültig, ob mit bipolaren oder mit Feldeffekttransistoren). Die einfachsten Transistorschaltstufen wirken invertierend (ein High-Pegel am Eingang bewirkt einen Low-Pegel am Ausgang und umgekehrt); die einfachsten Gatterschaltungen sind NAND-oder NOR-Verknüpfungen. Alles weitere baut darauf auf. Nicht invertierende Gatter enthalten typischerweise zusätzliche Schaltstufen. Sie haben deshalb längere Verzögerungszeiten.

Die Anzahl der Eingänge

Jeder Eingang bedeutet eine kapazitive Belastung. Je mehr Eingänge, desto länger die Verzögerungszeit des Gatters. Es ist deshalb nicht zweckmäßig, die Anzahl der Eingänge über einen gewissen Wert hinaus zu erhöhen. Braucht man mehr Eingänge, so sind die betreffenden Gatterfunktionen mit mehreren Gattern darzustellen (Kaskadierung).

Anzahl der Eingänge*	Technologie
2 oder 3	ASIC (Gate Array)
4 bis 6	Programmierbare Funktionszuordner (CPLDs, FPGAs); ODER-Verknüpfungen von Produkttermen (CPLDs)
2, 3, 4, 8, 13	TTL und CMOS; herkömmliche Baureihen (SSI)
2 bis 5	ECL (SSI)
2 oder 3	Gatterschaltkreise in modernen Logikbaureihen
um 20...über 50	Programmierbare Produktterme (PALs, GALs, CPLDs)

*: Typische Werte.

Tabelle 3.1 Die Anzahl der Eingänge von Gatterstrukturen in verschiedenen Technologien.

Die Invertierung

Es hängt von der Schaltungstechnologie ab, ob das Invertieren eines Signals zusätzlichen Aufwand kostet oder nicht. Danach richtet sich auch die Darstellung im Schaltbild.

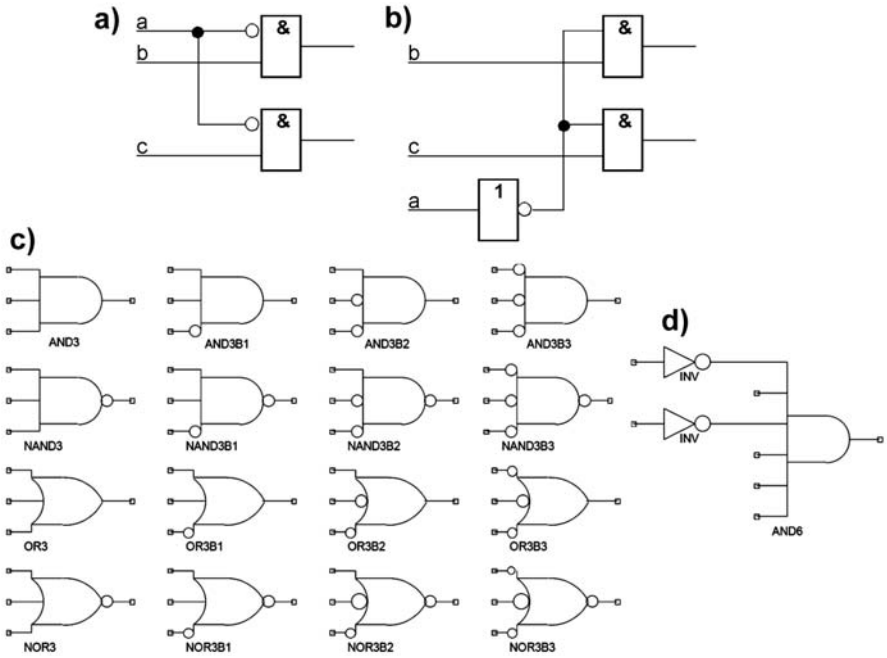


Abb. 3.3 Darstellung der Invertierung anhand von Beispielen. a) wenn es nur um Prinzipdarstellungen geht oder wenn die Invertierung keinen zusätzlichen Aufwand kostet (z. B. in CPLDs); b) wenn eigens Bauelemente (z. B. Negatoren) erforderlich sind. c) Gatter mit drei Eingängen. Die Funktionselementebibliothek dieser Entwicklungsumgebung enthält das vollständige Sortiment. d) ansonsten kostet es nichts, den Gattereingängen Inverter vorzuschalten.

3.1.3 Wechselseitige Wandlungen

Positive und negative Logik

Ein UND in positiver Logik entspricht einem ODER in negativer Logik und umgekehrt. Sinngemäß entsprechen NAND und NOR sowie XOR und XNOR einander.

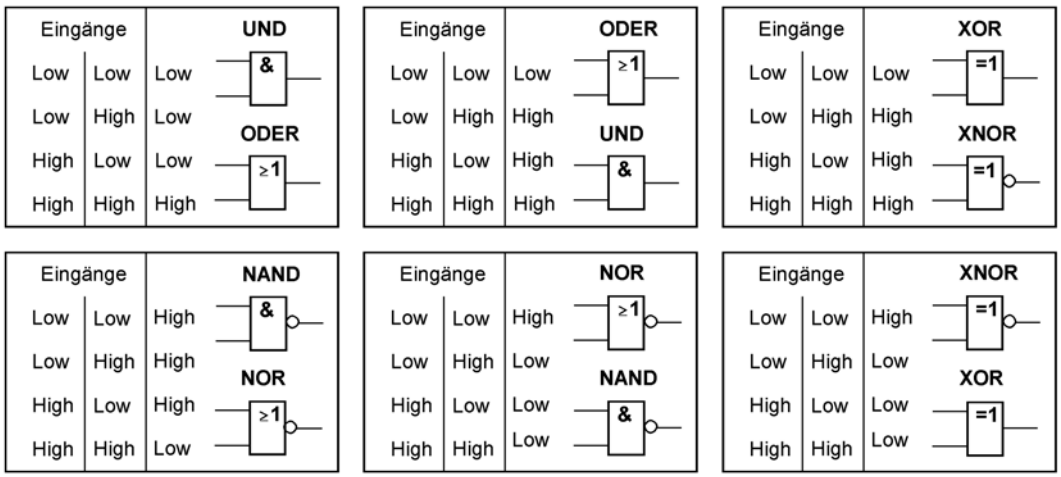


Abb. 3.4 Positive und negative Logik. Oben jeweils die Gatterfunktion in positiver, darunter in negativer Logik.

Gatter mit invertierten Eingängen

Werden die Eingänge eines Gatters invertiert oder werden dem Gatter invertierte Signale zugeführt, wirken sich die DeMorganschen Regeln folgendermaßen aus (Abb. 3.6):

a) Ein UND-Gatter wird zum NOR: $\bar{a} \cdot \bar{b} = \overline{a \vee b}$. (3.1)

b) Ein NAND-Gatter wird zum ODER: $\overline{\bar{a} \cdot \bar{b}} = a \vee b$. (3.2)

c) Ein NOR-Gatter wird zum UND: $\overline{\bar{a} \vee \bar{b}} = a \cdot b$. (3.3)

d) Ein ODER-Gatter wird zum NAND: $\bar{a} \vee \bar{b} = \overline{a \cdot b}$. (3.4)

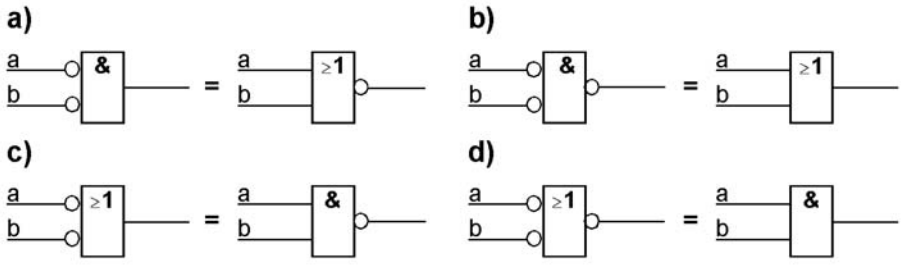


Abb. 3.5 Gatter mit invertierten Eingängen.

Elementare Verknüpfungen von Signalen, die aktiv Low wirken

Manche Signale sind als aktiv Low spezifiziert. Das ist z. B. typisch für Schaltkreisauswahlsignale (Chip Enable, Chip Select), für Speicher-Schreibimpulse, für Rücksetzsignale usw. Nicht selten ist es erforderlich, elementare Verknüpfungen derartiger Signale zu realisieren. Hierzu lassen sich die DeMorganschen Regeln vorteilhaft ausnutzen:

a) Ein UND-Gatter wirkt als ODER für Signale, die aktiv Low sind.

b) Ein NAND-Gatter wirkt als ODER mit aktiv-High-Ausgang für aktiv-Low-Eingangssignale.

c) Ein ODER-Gatter wirkt als UND für Signale, die aktiv Low sind.

d) Ein NOR-Gatter wirkt als UND mit aktiv-High-Ausgang für aktiv-Low-Eingangssignale.

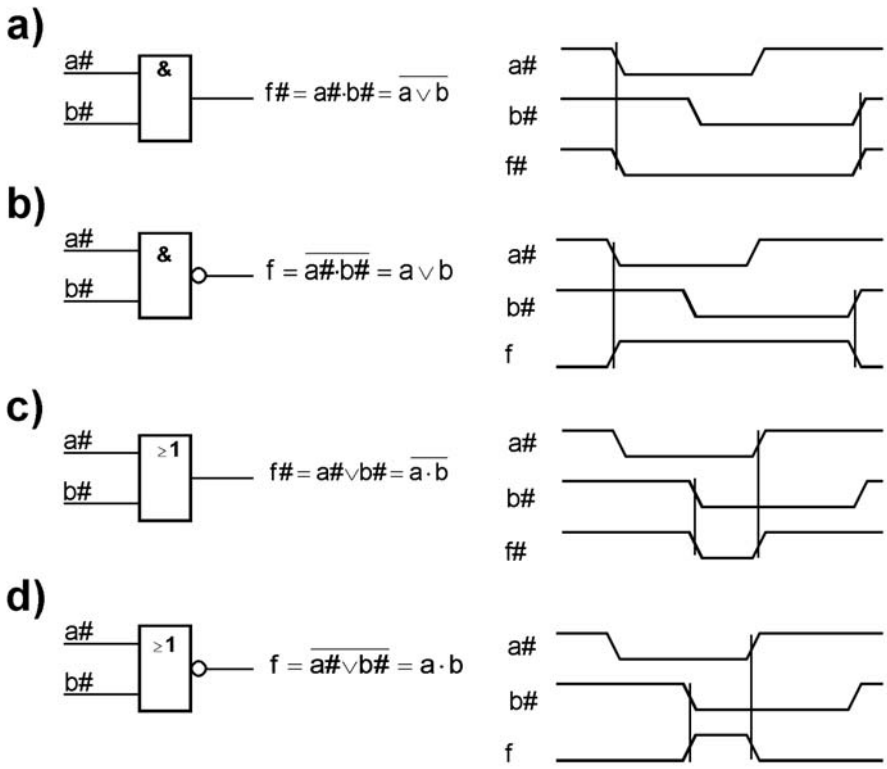


Abb. 3.6 Elementare Verknüpfungen von Signalen, die aktiv Low wirken. a) und b) disjunktive, c) und d) konjunktive Verknüpfungen.

3.1.4 Antivalenz und Äquivalenz (XOR und XNOR)

Wechselseitige Wandlungen

Ein XOR in positiver Logik entspricht einem XNOR in negativer Logik und umgekehrt.

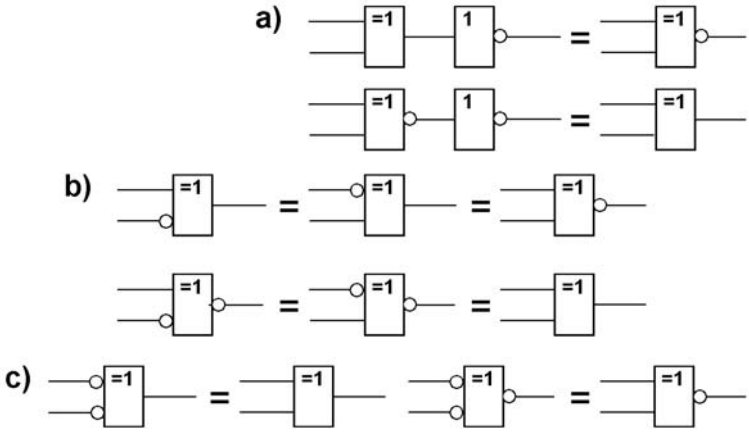


Abb. 3.7 Wechselseitige Wandlungen von Antivalenz- und Äquivalenzverknüpfungen (XOR und XNOR mit zwei Eingängen).

a) Ein XOR mit nachgeschaltetem Negator ergibt ein XNOR und umgekehrt.

b) Wird einer der Eingänge invertiert angesteuert, so wird aus einem XOR ein XNOR und umgekehrt.

c) Werden beide Eingänge invertiert angesteuert, bleibt die jeweilige Funktion erhalten.

XOR und XNOR als steuerbare Inverter

XOR- und XNOR-Gatter können als steuerbare Inverter eingesetzt werden[1)].

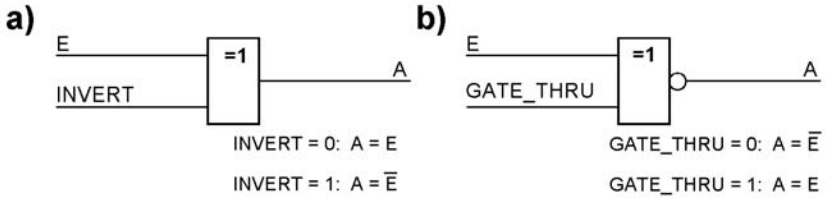


Abb. 3.8 Steuerbare Inverter.a) XOR; b) XNOR.

XOR und XNOR mit Grundgattern

Antivalenz- und Äquivalenzverknüpfungen können mit Grundgattern aufgebaut werden.

XOR und XNOR mit mehreren Eingängen

Die mehrfache Antivalenzverknüpfung (XOR mit mehr als zwei Eingängen) entspricht der mehrstelligen Addition modulo 2. Sie kann als Kaskadierung von elementaren Antivalenzverknüpfungen (mit jeweils zwei Variablen) dargestellt werden. Infolge der Kaskadierung ergibt sich eine beträchtliche Schaltungstiefe. Man kann sie in gewissem Maße verringern, wenn man Verknüpfungen mit beispielsweise drei oder vier Eingängen nicht

1: Anwendungsbeispiele: Signalübertragung mit minimaler Anzahl an Schaltflanken (Transition Minimized Signaling TMS), Verschlüsselungstechnik.

durch Kaskadierung verwirklicht, sondern (gemäß Wahrheitstabelle) direkt in zweistufige Gatternetzwerke umsetzt.

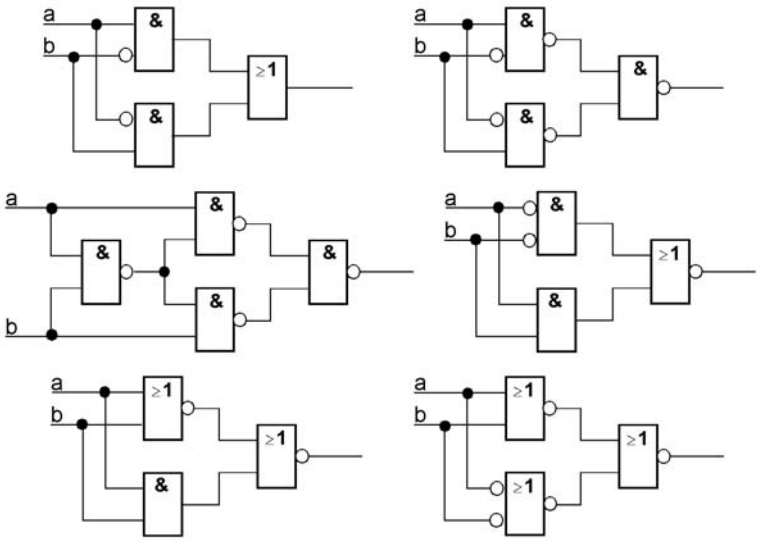


Abb. 3.9 XOR-Schaltungen mit zwei Eingängen.

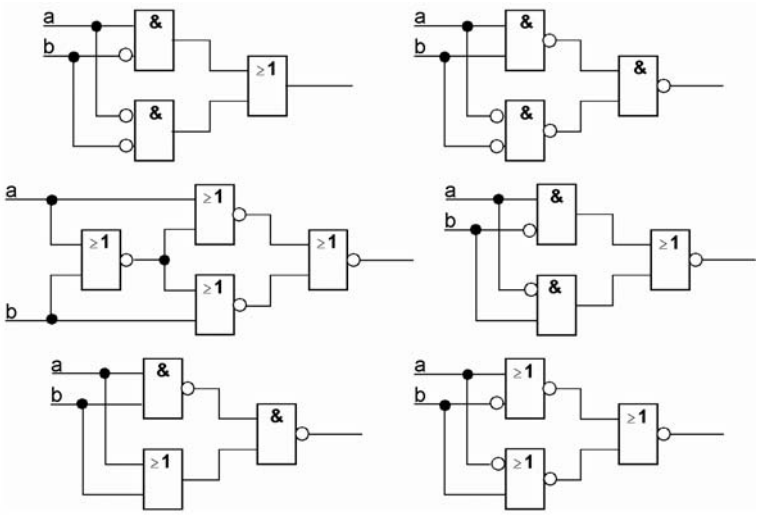


Abb. 3.10 XNOR-Schaltungen mit zwei Eingängen.

Die mehrfache Äquivalenzverknüpfung (XNOR mit mehr als zwei Eingängen) gibt es in zwei Varianten:

a) Als Kaskadierung von XNOR-Gattern mit zwei Eingängen. Dies entspricht der Hintereinanderausführung von elementaren Äquivalenzverknüpfungen. Diese Ausführung ist beispielsweise notwendig, um Schaltfunktionen in einer Äquivalenz-Normalform darstellen zu können.

b) Als Invertierung der XOR-Verknüpfung: $a \otimes b \otimes c... = \overline{a \oplus b \oplus c...}$ Beispiele: die XNOR-Gatter in den Funktionselementebiblioheken von Entwicklungsumgebungen.

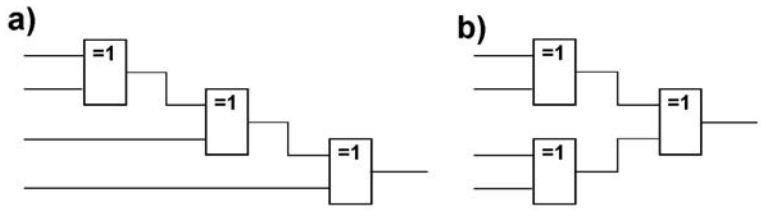


Abb. 3.11 XOR mit mehr als zwei Eingängen: Kaskadierung elementarer Antivalenzverknüpfungen. a) Ketten-, b) Baumstruktur.

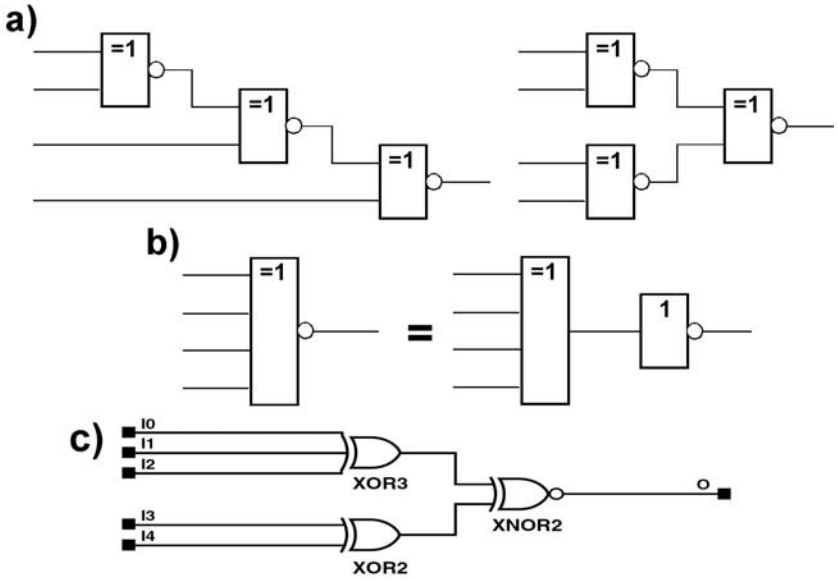


Abb. 3.12 XNOR mit mehr als zwei Eingängen. a) Kaskadierung elementarer Äquivalenzverknüpfungen; b), c) XNOR als Invertierung der XOR-Verknüpfung. c) stammt aus der Funktionselementebeschreibung einer Entwicklungsumgebung).

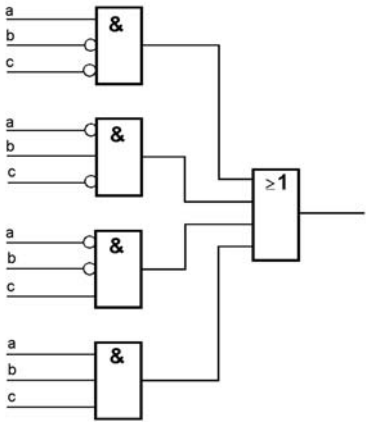


Abb. 3.13 XOR-Verknüpfung mit drei Eingängen als zweistufiges Schaltnetz aus Grundgattern.

3.1.5 Kaskadierung

Kaskadierung bedeutet hier, aus Gattern oder Schaltungen mit vergleichsweise wenigen Eingängen funktionell gleichartige Gatter oder Schaltungen mit mehreren Eingängen aufzubauen.

Kaskadierung von Grundgattern

Oftmals braucht man Verknüpfungen, wie UND, ODER, NAND, NOR usw. mit mehr als zwei Eingängen. Grundsätzlich läßt sich die Anzahl der Eingänge durch fortgesetztes Kaskadieren beliebig erhöhen.

Kaskadierung von Gattern des gleichen Typs

Die folgende Abbildung zeigt, wie aus Gattern mit zwei Eingängen gleichartige Verknüpfungen mit vier Eingängen aufgebaut werden können. Nicht invertierende Gatter (UND, ODER) kann man unmittelbar hintereinanderschalten. Handelt es sich um invertierende Gatter (NAND, NOR), sind Negatoren einzufügen. Hierdurch ergibt sich ein vergleichsweise hoher Aufwand.

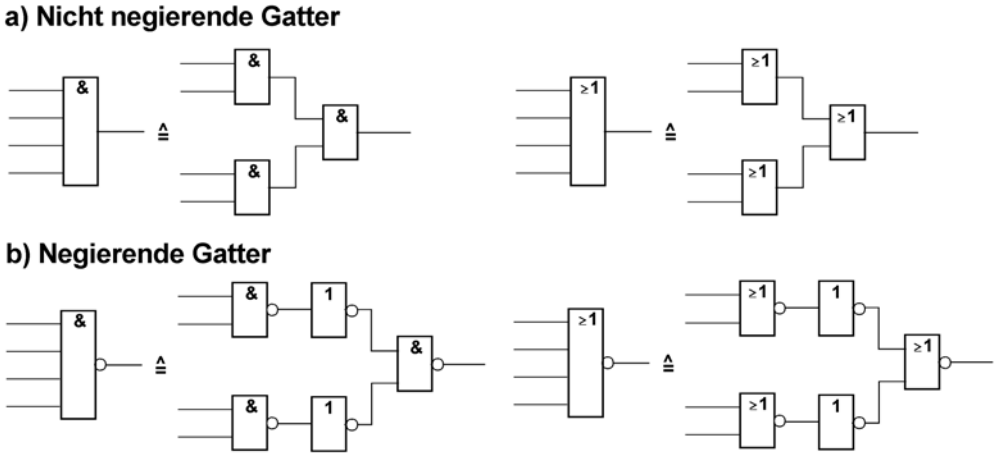


Abb. 3.14 Kaskadierung von Grundgattern. a) nicht invertierend (UND, ODER); b) invertierend (NAND, NOR).

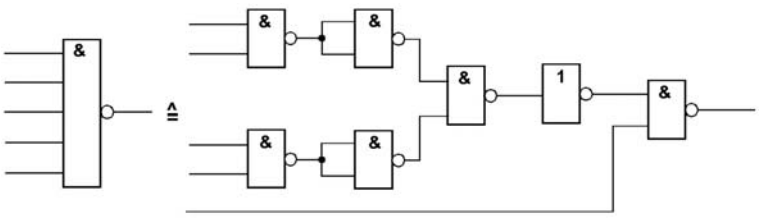


Abb. 3.15 Kaskadierungsbeispiel: ein NAND-Gatter mit fünf Eingängen aus NAND-Gattern mit zwei Eingängen. Es sind sieben Gatterfunktionen erforderlich.

Kaskadierung mit Gattern unterschiedlicher Typen
NAND-Gatter können durch Vorschalten von UND-Gattern erweitert werden, NOR-Gatter durch Vorschalten von ODER-Gattern.

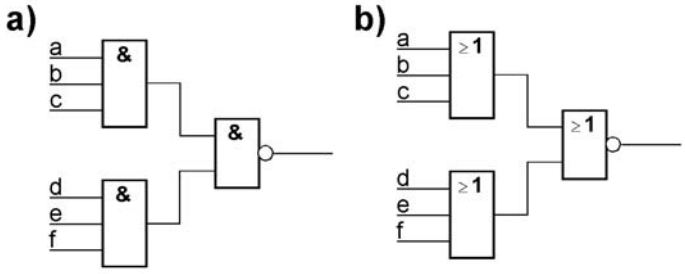


Abb. 3.16 Erweiterung von NAND und NOR. a) NAND mit UND; b) NOR mit ODER.

Erweiterung für wahre (nicht invertierte) und invertierte Signale
Der Anwendungsfall: die zu verknüpfenden Signale liegen alle in einer Polarität vor (wahr oder invertiert), es sind aber teils wahre, teils invertierte Signale zu verknüpfen. Das Ziel: zusätzliche Negatoren vermeiden. Die Lösung: die Verknüpfungen der zu invertierenden Signale werden mit vorgeschalteten Gattern eines jeweils anderen Typs erledigt. UND- und NAND-Gatter werden in diesem Sinne mit NOR-Gattern erweitert, ODER- und NOR-Gatter mit NAND-Gattern (Ausnutzung der DeMorganschen Regeln).

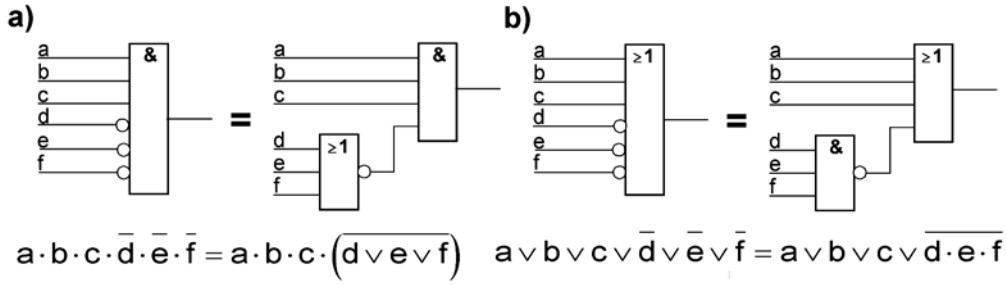


Abb. 3.17 Negatoren einsparen. Hierzu werden konjunktive und disjunktive Verknüpfungen mit invertierten Signalen auf vorgeschaltete Gatter ausgelagert. a) Erweiterung einer UND- oder NAND-Verknüpfung. NOR wirkt als UND für invertierte Variable. b) Erweiterung einer ODER- oder NOR-Verknüpfng. NAND wirkt als ODER für invertiete Variable.

Kaskadierung nach DeMorgan
Mit NAND- und NOR-Gattern kann man Verknüpfungen (UND, ODER, NAND, NOR) mit vielen Eingängen aufbauen, ohne zusätzliche Negatoren zu benötigen. Da die typischen modernen Logikbaureihen sowohl NAND- als auch NOR-Funktionen enthalten, ist dies die naheliegende Grundschaltung der Kaskadierung.

UND-Verknüpfung
Ein NOR entspricht einer UND-Verknüpfung invertierter Signale. Eine UND-Verknüpfung mit vielen Eingängen kann somit durch NAND-Gatter dargestellt werden, deren Ausgänge an ein NOR-Gatter angeschlossen sind.

ODER-Verknüpfung
Ein NAND entspricht einer ODER-Verknüpfung invertierter Signale. Eine ODER-Verknüpfung mit vielen Eingängen kann somit durch NOR-Gatter dargestellt werden, deren Ausgänge an ein NAND-Gatter angeschlossen sind.

NAND- und NOR-Gatter mit vielen Eingängen ergeben sich sinngemäß, indem die ausgangsseitige Invertierung weggelassen wird.

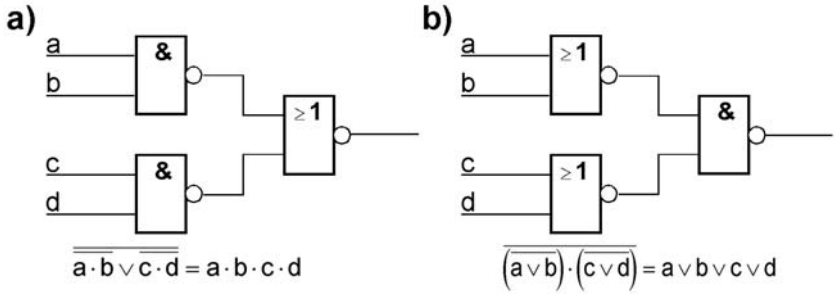


Abb. 3.18 Kaskadierung nach DeMorgan (1). a) UND-Verknüpfung; b) ODER-Verknüpfung mit jeweils vier Eingängen.

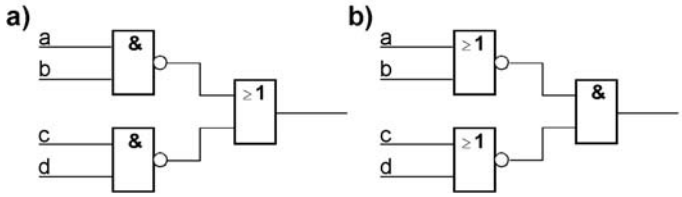


Abb. 3.19 Kaskadierung nach DeMorgan (2). a) NAND-Verknüpfung; b) NOR-Verknüpfung mit jeweils vier Eingängen.

Um Gatterfunktionen mit noch mehr Eingängen zu bilden, können die gezeigten Strukturen ihrerseits kaskadiert werden. Haben alle Gatter zwei Eingänge, so ergeben sich Baumstrukturen (DeMorgan Trees) mit vier, 16, 64 usw.

Grundstrukturen der Kaskadierung
Sind kombinatorische Verknüpfungen durch Kaskadieren zu verwirklichen, so hat man die Wahl zwischen zwei grundsätzlichen Strukturen: Kette und Baum. Haben die Grundgatter nur zwei Eingänge, so ergibt sich der Binärbaum als einfachste Baumstruktur.

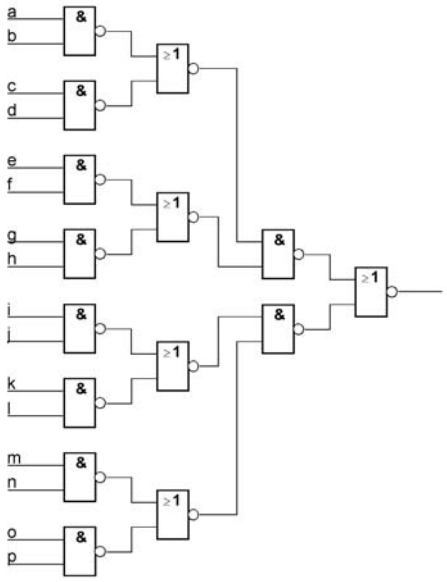


Abb. 3.20 Kaskadierung nach DeMorgan (3). Zweistufige Kaskadierung am Beispiel der UND-Verknüpfung. Die ausgangsseitige Stufe gemäß Abb. 3.19 hat vier Eingänge. Werden vier gleichartige Stufen vorgeschaltet, ergibt sich ein Gatter mit 16 Eingängen.

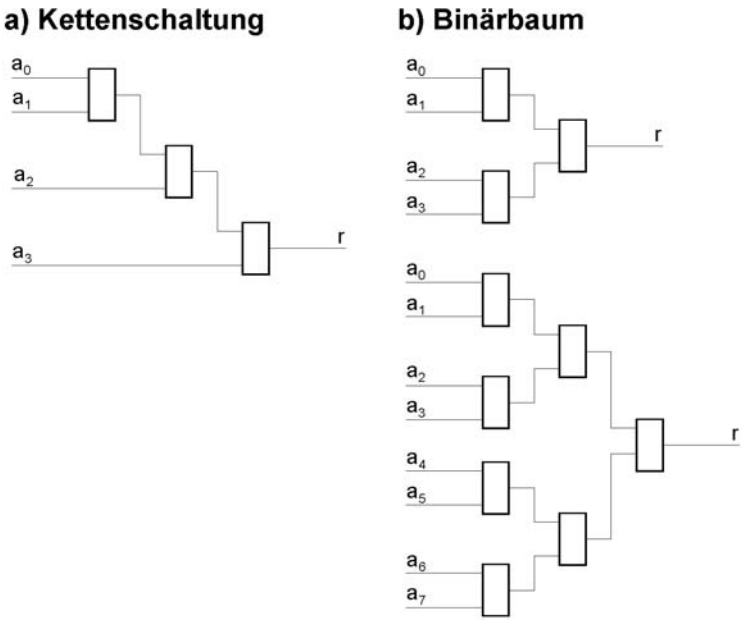


Abb. 3.21 Mehrfachkaskadierung. a) Kettenschaltung (Daisy Chain; Ripple Thru); b) Binärbaum (Binary Tree) anhand von zwei Beispielen (oben für vier, darunter für acht Variable).

Ersichtlicherweise erfordert es stets n – 1 Gatter mit zwei Eingängen, um ein Gatter mit n Eingängen aufzubauen. Worin sich beide Strukturen wesentlich unterscheiden, ist die Schaltungstiefe s (und damit die Schaltverzögerungszeit):

a) Kettenschaltung. Die Schaltungstiefe s wächst linear mit der Anzahl der Eingänge, also gemäß O(n). Bei n Eingängen ist s = n – 1.

b) Binärbaum. Die Schaltungstiefe s wächst gemäß dem Zweierlogarithmus der Anzahl der Eingänge, also gemäß O(ld n). Bei n Eingängen ist s = ceil (ld n).

Damit ist – was die Geschwindigkeit angeht – der Binärbaum der Kettenschaltung grundsätzlich überlegen, und zwar umso mehr, je größer die Anzahl der Eingänge ist. Beispiel: Bei 32 Eingängen ist die Schaltungstiefe s der Kettenschaltung = 31 und die des Binärbaums = 5.

Der Schaltungsaufwand

Das Aufwandsproblem soll am Beispiel des Binärbaums erläutert werden. Es zeigt sich dann, wenn Folgen von Verknüpfungen gemeinsamer Eingangssignale zu bilden sind. Im Beispiel von Abb. 3.22 gilt:

- Das erste Ausgangssignal hängt von den ersten beiden Eingangssignalen ab: $r_1 = f_1 (a_0, a_1)$.
- Das zweite Ausgangssignal hängt von den ersten drei Eingangssignalen ab: $r_2 = f_2 (a_0, a_1, a_2)$.
- Das dritte Ausgangssignal hängt von den ersten vier Eingangssignalen ab: $r_3 = f_3(a_0, a_1, a_2, a_3)$ usw.

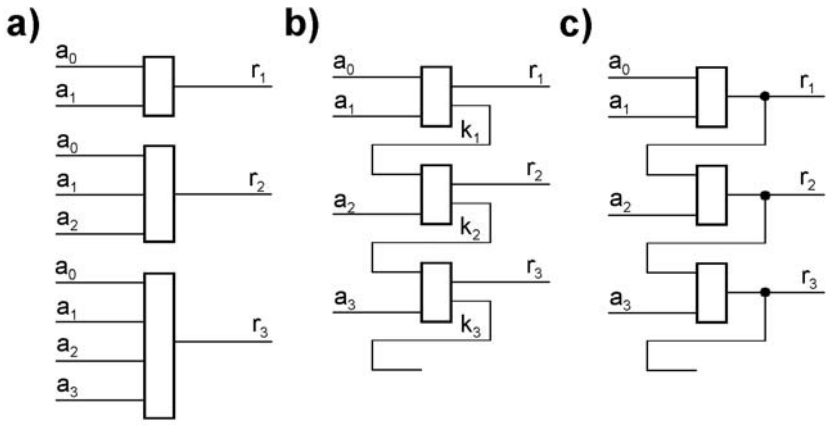


Abb. 3.22 Typische Verknüpfungen gemeinsamer Eingangssignale. a) unabhängige Bildung; b) und c) Kaskadierungen in Kettenschaltung.

Oftmals sind die Schaltfunktionen f_1, f_2 usw. auf Verknüpfungen der jeweils neu hinzukommenden Eingangssignale mit Signalen zurückzuführen, die in vorhergehenden Schaltungsteilen gebildet werden können. Typische Beispiele sind Prioritätsnetzwerke, Zählnetzwerke sowie die Übertragsweitergabe beim Addieren. Im Beispiel von Abb. 3.22b werden in jeder Stelle i (i = 1, 2, 3 usw.) ein Resultatsignal r_i und ein Kaskadierungssignal k_i gebildet:

- $r_1 = f_1 (a_0, a_1)$; $k_1 = g_1 (a_0, a_1)$,
- $r_2 = f_2 (k_1, a_2)$; $k_2 = g_2 (k_1, a_1)$,
- $r_3 = f_3 (k_2, a_3)$; $k_3 = g_3 (k_2, a_1)$ usw.

Im einfachsten Fall können die vorhergehenden Ausgangssignale direkt in die nachfolgende Schaltfunktion einfließen (Abb. 3.22c):

- $r_1 = f_1 (a_0, a_1)$,
- $r_2 = f_1 (r_1, a_2)$,
- $r_3 = f_3 (r_2, a_3)$ usw.

Derartige funktionelle Abhängigkeiten legen eine Kettenstruktur nahe. Der Vorteil: geringer Aufwand, da die jeweils vorhergehenden Gatter für die nachfolgenden Verknüpfungen mitgenutzt werden. Der Nachteil: die Schaltungstiefe wächst linear mit der Verarbeitungsbreite (Abhängigkeit gemäß O(n)).

Alternativ dazu könnte man alle Ausgangssignale tatsächlich unabhängig voneinander erzeugen. Hierzu braucht man – von Stufe zu Stufe – Gatter mit immer mehr Eingängen. Wenn diese Gatter als Binärbaumstrukturen aufgebaut werden, wächst die Schaltungstiefe nur gemäß O(ld n). Der Aufwand wächst aber gemäß $O(n^2)$. Und gegen quadratisch wachsende Komplexität kann letzten Endes auch die neueste Schaltkreistechnologie nicht helfen ...

Beispiel: Aus 32 Variablen $a_{31}...a_0$ sind 31 Ergebnisbits $r_{31}...r_1$ zu bilden (Bit r_1 durch Verknüpfung von a_0 und a_1, Bit r_2 durch Verknüpfung von Bit r_1 mit a_2 usw.; vgl. Abb. 3.22c). Jede einzelne Verknüpfung r_k (k = 0...31) entspricht einer Gatterfunktion mit k + 1 Eingängen. Die gesamte Schaltung ist mit Gattern aufzubauen, die zwei Eingänge haben.

- Die Kettenschaltung kommt mit 31 Gattern aus. Für Bit r_{31} ergibt sich aber eine Schaltungstiefe s = 31.
- Werden die Ausgangssignale unabhängig voneinander gebildet (vgl. Abb 3.23a) und die einzelnen Gatter als Baumstrukturen aufgebaut (vgl. Abb. 3.22b), so braucht man für Bit r_1 ein Gatter (zwei Variable), für Bit r_2 zwei Gatter (drei Variable) usw. Bit r_{31} erfordert somit 31 Gatter (32 Variable $a_0...a_{31}$). Insgesamt braucht man also 1 + 2 + 3 +... + 31 Gatter, also allgemein für n Variable[2)]:

$$\frac{(n-1)^2 + n - 1}{2} \text{ Gatter.}$$

  Für 32 Variable ergeben sich somit 496 Gatter.

2: Gemäß Summenformel der arithmetischen Reihe 1 + 2 + 3 +...+ (n-1).

Der Aufwand wächst also offensichtlich mit dem Quadrat der Anzahl der Variablen. Dieses Wachstum gemäß $O(n^2)$ setzt der Anwendbarkeit von Baumstrukturen Grenzen, so dass typischerweise Kompromißlösungen zwischen Baum- und Kettenstrukturen gewählt werden. Beispielsweise liegt es nahe, für die Verknüpfung von acht Variablen in Abb. 3.21b (unten) die Verknüpfung der ersten vier Variablen (oben) mitzunutzen (die untere Verknüpfung der Variablen $a_0...a_3$ könne somit entfallen). In der Schaltungspraxis sind die Grenzen dieser Mitnutzung u. a. durch Leitungslängen, kapazitive Belastung und verfügbare Verbindungsressourcen gegeben.

3.2 Zweistufige Schaltnetze und Normalformen

Jede Schaltfunktion kann mit zweistufigen Schaltnetzen dargestellt werden, deren Struktur einer disjunktiven oder konjunktiven Normalform entspricht. Die Signalleitungen der Variablen sind direkt oder invertiert an die Gatter der ersten Stufe angeschlossen. Die zweite Stufe besteht aus einem einzigen Gatter, das eingangsseitig mit den Ausgängen aller Gatter der ersten Stufe verbunden ist. Die disjunktive Normalform entspricht den Schaltungsstrukturen UND-ODER und NAND-NAND; die konjunktive Normalform entspricht den Schaltungsstrukturen ODER-UND und NOR-NOR.

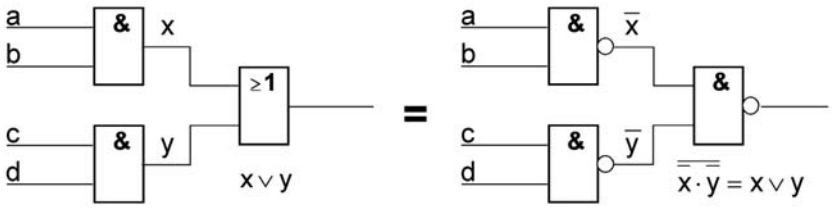


Abb. 3.23 Verknüpfung in disjunktiver Normalform. a) UND-ODER-, b) NAND-NAND-Struktur. Das ausgangsseitige NAND wirkt als disjunktive Verknüpfung für invertierte Signale. Diese werden von den vorgeschalteten NAND-Gattern geliefert.

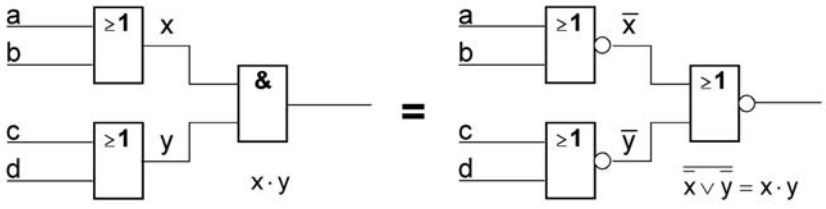


Abb. 3.24 Verknüpfung in konjunktiver Normalform. a) ODER-UND-, b) NOR-NOR-Struktur. Das ausgangsseitige NOR wirkt als konjunktive Verknüpfung für invertierte Signale. Diese werden von den vorgeschalteten NOR-Gattern geliefert.

Von der Schaltfunktion, Wahrheitstabelle oder Belegungsliste zur disjunktiven Normalform:

1. Für jeden Minterm oder für jeden Eintrag, dem der Funktionswert Eins zugeordnet ist, ein konjunktiv wirkendes Gatter vorsehen (UND oder NAND).
2. Die Signale, die mit einer invertierten Variablen oder mit Null belegt sind, invertiert anschließen.
3. Die Signale, die mit einer nicht invertierten Variablen oder mit Eins belegt sind, direkt anschließen.
4. Die Ausgänge aller konjunktiv wirkenden Gatter an ein disjunktiv wirkendes Gatter anschließen (ODER oder NAND).

Von der Schaltfunktion, Wahrheitstabelle oder Belegungsliste zur konjunktiven Normalform:

1. Für jeden Maxterm oder für jeden Eintrag, dem der Funktionswert Null zugeordnet ist, ein disjunktiv wirkendes Gatter vorsehen (ODER oder NOR).
2. Die Signale, die mit einer invertierten Variablen oder mit Null belegt sind, direkt anschließen.
3. Die Signale, die mit einer nicht invertierten Variablen oder mit Eins belegt sind, invertiert anschließen.
4. Die Ausgänge aller Gatter an ein konjunktiv wirkendes Gatter (UND oder NOR) anschließen.

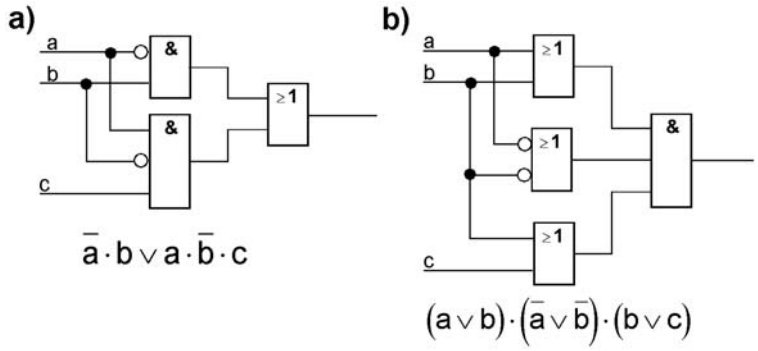


Abb. 3.25 Implementierungsbeispiele. a) disjunktive, b) konjunktive Normalform.

Disjunktive und konjunktive Normalformen mit invertierten Signalen

Ein NAND wirkt als disjunktive, ein NOR als konjunktive Verknüpfung invertierter Signale. Somit kann man mit NAND-Gattern konjunktive und mit NOR-Gattern disjunktive Normalformen darstellen, wobei sich invertierte Ausgangssignale ergeben.

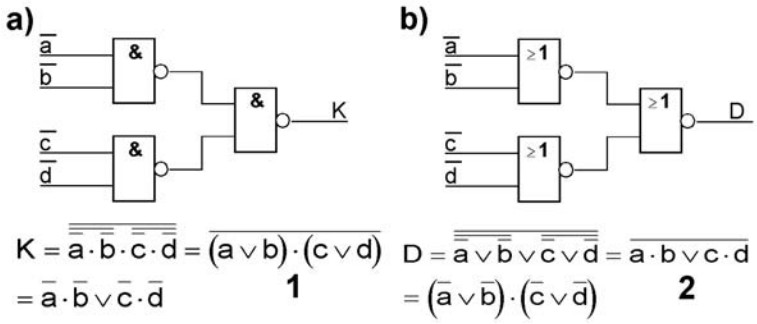


Abb. 3.26 Verknüpfungen invertierer Signale. a) mit NAND; b) mit NOR. 1 - invertierte KNF; 2 - invertierte DNF.

3.3 Decodieren und Codieren

Ein Code ist im Grunde eine Vorschrift, die Bitmustern oder Signalbelegungen bestimmte Bedeutungen zuweist. Die Bitmuster oder Signalbelegungen, die zu einem Code gehören, heißen auch Codewörter. So entspricht im BCD-Code das Bitmuster 1001B der Ziffer 9 und im ASCII-Code das Bitmuster 41H dem Buchstaben A. Viele Codes werden durch Codetabellen beschrieben, die alle zum Code gehörenden Codewörter und deren Bedeutung enthalten. Decodieren heißt, bestimmte Bitmuster – mit anderen Worten: Belegungen von Signalleitungen – zu erkennen. Codieren heißt, Bitmuster zu bilden, die bestimmten Bedeutungen entsprechen.

3.3.1 Decodieren

Die einfachste Decodierschaltung erkennt, ob ein bestimmtes Bitmuster oder eine bestimmte Signalbelegung vorliegt. Es ist beispielsweise zu erkennen, ob drei Signalleitungen a, b, c mit dem Bitmuster 1, 0, 1 belegt sind. Das ist offensichtlich dann gegeben, wenn Leitung a mit Eins UND Leitung B mit Null UND Leitung c mit Eins belegt ist ($a \cdot \bar{b} \cdot c$). Die Aufgabe der Decodierung kann also mit UND-Verknüpfungen gelöst werden. Jede Signalbelegung läßt sich durch eine UND-Verknüpfung erfassen[3)], in die mit Eins belegte Signale direkt und mit Null belegte invertiert einbezogen werden.

Der 1-aus-n-Code

Ein UND-Gatter hat nur dann eine Ausgangsbelegung Eins, wenn das betreffende Bitmuster an seinen Eingängen anliegt. Sind mehrere solcher UND-Gatter vorgesehen, so kann zu jedem Zeitpunkt nur eine Ausgangsbelegung den Wert Eins haben. Auch die von den UND-Verknüpfungen einer Decodierschaltung gebildeten Signalbelegungen kann man als Codewörter betrachten. Da sie höchsten eine einzige Eins enthalten, bezeichnet man einen solchen Code als 1-aus-n-Code (One Hot Encoding OHE). Ein Decoder mit k Eingängen und n Ausgängen heißt auch k-zu-n-Decoder. Demgemäß zeigt Abb. 3.27d einen 3-zu-8-Decoder.

3: Jede Belegung von k Signalen entspricht einem Punkt im Booleschen Raum B^k. Dieser kann über eine UND-Verknüpfung ausgewählt werden.

Die vollständige Decodierung

k Signalleitungen können bis zu 2^k verschiedene Belegungen aufweisen. Eine Decodierschaltung (Decoder), die alle Belegungen erkennen kann (Binary to One Hot), hat k Eingänge und $n = 2^k$ Ausgänge. Sie enthält 2^k UND-Verknüpfungen. Es ist jeweils nur einer der Ausgänge aktiv.

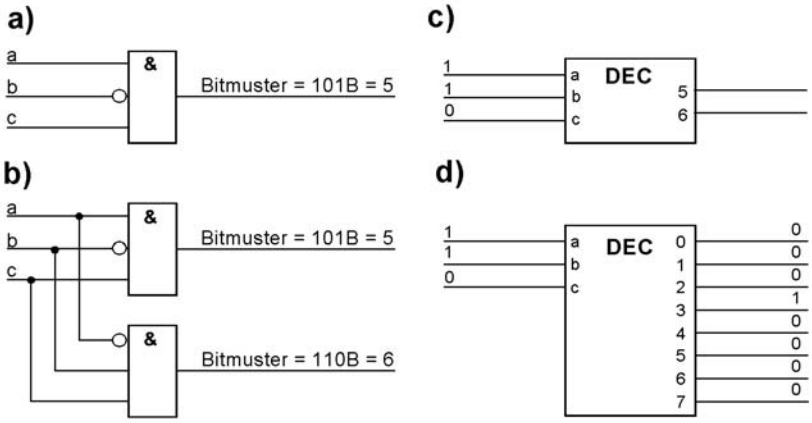


Abb. 3.27 Decoder. Elementare Schaltungen, Schaltsymbole und Belegungsbeispiele. a) Decodierung einer einzigen Signalbelegung. b) diese Decodierschaltung spricht nur auf zwei Signalbelegungen an (unvollständige Decodierung). c) Schaltsymbol dieses Decoders. Bei der gezeigten Eingangsbelegung wird keiner der Ausgänge aktiv. d) vollständige Decodierung. Dieser Decoder kann jede der acht Signalbelegungen der drei Eingänge erkennen (Binär-zu-OHE- oder 3-zu-8-Decoder).

Generische Schaltungen

Die einfachste generische Schaltung zum Decodieren einer einzelnen Signalbelegung besteht aus einem UND-Gatter, an das die betreffenden Signale gemäß der zu decodierenden Belegung direkt oder invertiert angeschlossen sind. Das Gatter hat einen weiteren Eingang, der zu Steuer- und Kaskadierungszwecken genutzt werden kann (Erlaubnis- oder Erweiterungseingang E). Ein Decoder, der alle 2^n Belegungen von n Signalen erkennen kann (Binary to One Hot), enthält 2^n UND-Gatter oder universelle Decodierstufen, deren Erlaubniseingänge miteinander verbunden sind.

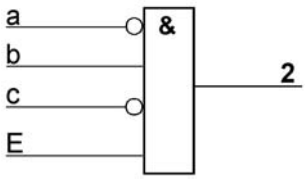


Abb. 3.28 Das UND-Gatter als generische Decodierschaltung. Beispiel für drei Signale a, b, c, deren Belegung 010B = 2 zu decodieren ist. E - Erlaubnis- oder Erweiterungseingang (Enable).

a)

Codes:

	Binär			1-aus-n							
dez.	c	b	a	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	1	0	0
3	0	1	1	0	0	0	0	1	0	0	0
4	1	0	0	0	0	0	1	0	0	0	0
5	1	0	1	0	0	1	0	0	0	0	0
6	1	1	0	0	1	0	0	0	0	0	0
7	1	1	1	1	0	0	0	0	0	0	0

b)

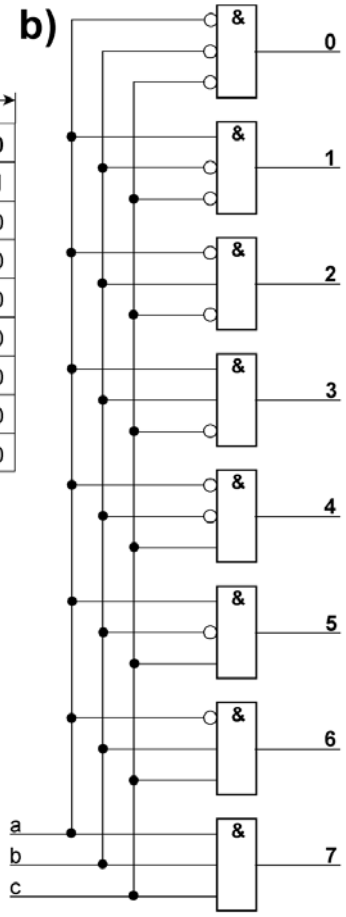
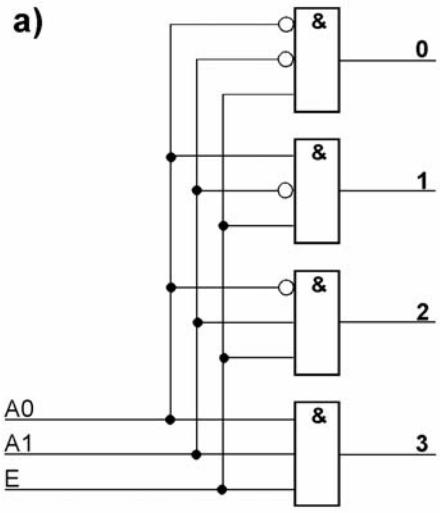


Abb. 3.29 Decoder für dreistellige Binärzahlen. a) Wahrheitstabelle; b) Schaltung.

a)



b)

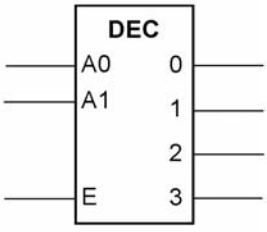


Abb. 3.30 Generische Decodierschaltung für die vollständige Decodierung der Belegungen zweier Signale a, b. a) Schaltung; b) Schaltsymbol.

Viele Bits decodieren

Um Belegungen von n Signalleitungen zu erkennen, sind UND-Verknüpfungen mit n Eingängen erforderlich (wobei meist wenigstens ein Erlaubniseingang (E) hinzukommt). Vor allem die Decodierung von Adressen führt auf ziemlich große Werte für n. Typische Werte reichen – gemäß den üblichen Speicherkapazitäten und der Auslegung der Bussysteme – von 8...16 bis 32 Bits und mehr.

Kaskadierung mehrerer Decoder

Die Decoder werden in zwei oder mehr Ebenen angeordnet. Die erste Ebene liefert die 2^n Ausgangssignale. Hat der einzelne Decoder 2^k Ausgänge, so sind in der ersten Ebene 2^{n-k} Decoder erforderlich. Zu einer Zeit kann nur einer dieser Decoder aktiv sein. Um den jeweils aktiven Decoder der ersten Ebene auszuwählen, ist ein Decoder der zweiten Ebene vorgesehen, dessen Ausgänge mit den Erlaubniseingängen (E) der Decoder der ersten Ebene verbunden sind.

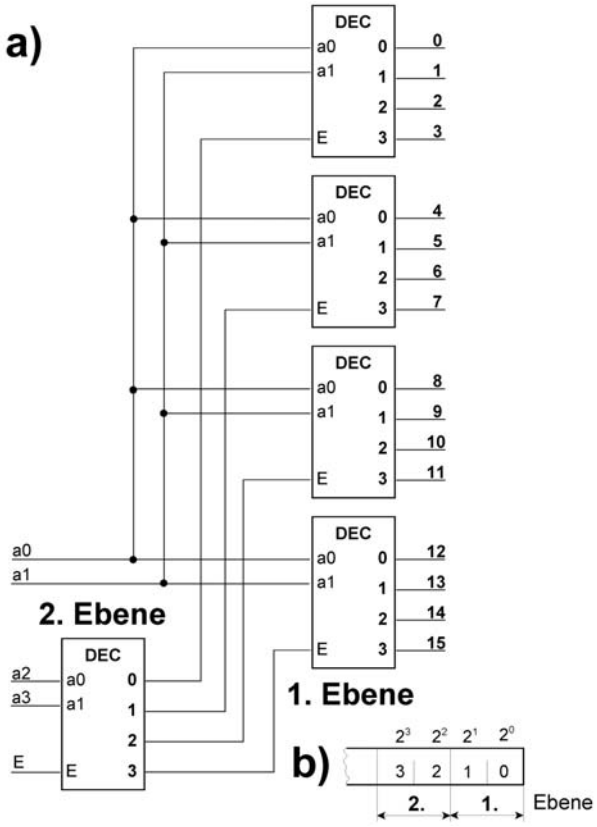


Abb. 3.31 Kaskadierung von Decodern. a) Schaltung; b) die Aufteilung der Eingangssignale auf die beiden Ebenen. Die erste Ebene liefert der Ausgangssignale, die zweite Ebene aktiviert die Decoder der ersten Ebene.

Es sind sämtliche Belegungen von vier Signalen zu decodieren. Die 16 Ausgänge erfordern vier Decoder der ersten Ebene, deren Erlaubniseingänge (E) den Ausgängen eines fünften Decoders nachgeschaltet werden (zweite Ebene). Es ist offensichtlich, dass das Prinzp auf mehr Ebenen erweitert werden kann. Dabei sind die Erlaubniseingänge (E) der jeweils

niederen Ebene mit den Ausgängen der jeweils höheren Ebene verbunden. Die niedrigstwertigen der zu decodierenden Signale werden an die Decoder der ersten Ebene angeschlossen, die nächst-höherwertigen an die der zweiten Ebene usw.

Matrixorganisation
Das Prinzip soll anhand einer sechs Bits langen Binärzahl gezeigt werden, die vollständig zu decodieren ist. Es ist also eine Decodierschaltung zu bauen, die sechs Eingänge und 64 Ausgänge hat (6-zu-64-Decoder). Die zu decodierenden sechs Bits werden in zwei Abschnitte (Spalte und Zeile) zu je drei Bits zerlegt. Beide Abschnitte werden unabhängig voneinander decodiert (3-zu-8; das erfordert jeweils acht UND-Gatter mit drei Eingängen). Soll der Decoder einen Erlaubniseingang haben, sind UND-Gatter mit vier Eingängen erforderlich. Die Zeilen- und Spaltensignale werden mit insgesamt 64 UND-Gattern mit zwei Eingängen zu den eigentlichen Ausgangssignalen verknüpft. Die Matrixorganisation braucht insgesamt nicht weniger Gatter. Die Aufwandsersparnis ergibt sich vielmehr daraus, dass man mit Gattern auskommt, die weniger Eingänge haben.

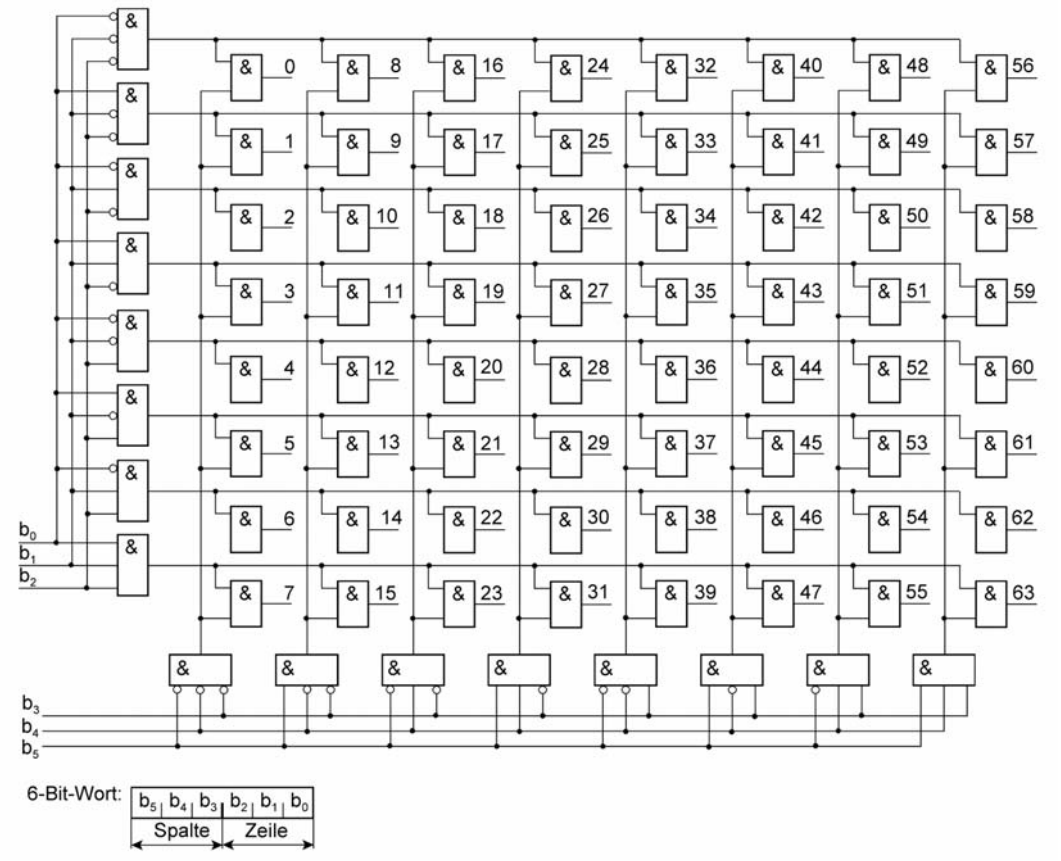


Abb. 3.32 6-Bit-Decoder (6 zu 64) in Matrixorganisation.

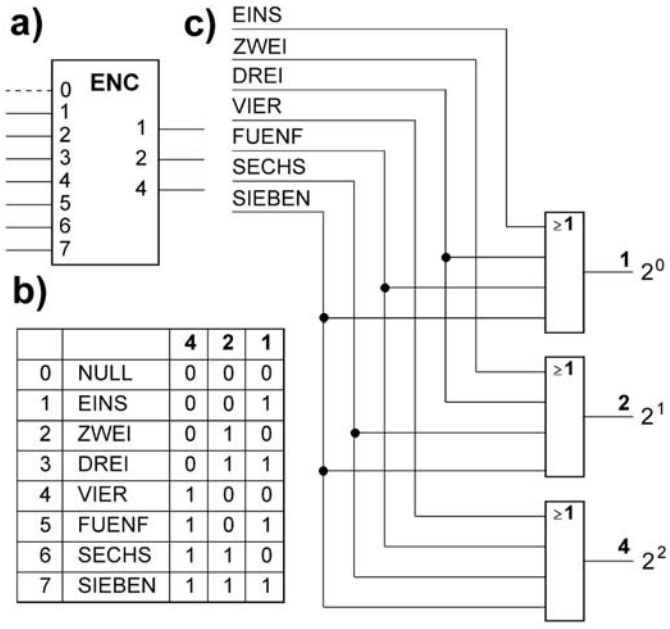
3.3.2 Codieren

Eine Codierschaltung (Codierer, Encoder) gibt Bitmuster aus, die einem bestimmten Code entsprechen. Der Begriff des Codierens (Encoding) soll hier auf Einrichtungen beschränkt

bleiben, die für jedes zu bildende Bitmuster einen eigenen Eingang haben, wobei zu einer Zeit nur einer dieser Eingänge erregt ist (1-aus-n-Code; OHE). Ist die Anzahl k der Eingänge eine Zweierpotenz, so lassen sich maximal n verschiedene Codeworte von jeweils ld n Bits Länge bilden (One Hot to Binary Encoding). Eine solche Codierschaltung hat $k = 2^n$ Eingänge und n Ausgänge. Sie enthält n ODER-Verknüpfungen mit 2^{n-1} Eingängen.

Generische Schaltungen

Für jeden Ausgang ist eine ODER-Verknüpfung vorgesehen, an die all jene Eingänge angeschlossen sind, bei deren Aktivierung der betreffende Ausgang aktiv werden muss. Aus der Codetabelle ergibt sich die Schaltung auf naheliegende Weise. Beispiel: das Ausgangssignal $2 = 2^1$ ist zu aktivieren, wenn eingangsseitig eine der Binärzahlen Zwei ODER Drei ODER Sechs ODER Sieben anliegt.



		4	2	1
0	NULL	0	0	0
1	EINS	0	0	1
2	ZWEI	0	1	0
3	DREI	0	1	1
4	VIER	1	0	0
5	FUENF	1	0	1
6	SECHS	1	1	0
7	SIEBEN	1	1	1

Abb. 3.33 Codierer (Encoder) für binär codierte Dezimalzahlen. a) Schaltsymbol; b) Codetabelle; c) Schaltung.

Umcodieren (Codewandlung)

Beliebige Codewandlungen von einem Code A in einen Code B lassen sich durch Hintereinanderschalten eines Decoders und eines Codierers (Encoders) verwirklichen (erste Wandlung von A nach OHE, zweite Wandlung von OHE nach B).

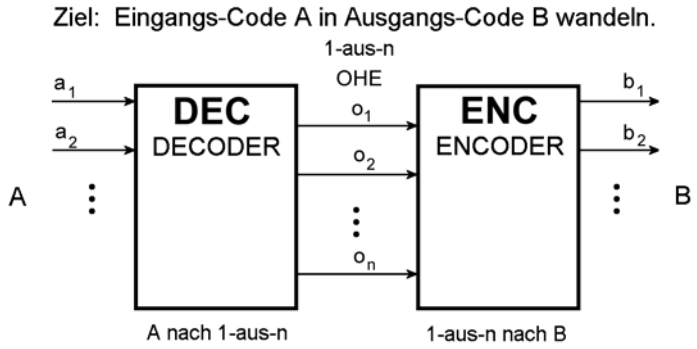


Abb. 3.34 Codewandlung (Prinzip).

Die Belegungen der OHE-Signale o_i (i = 1...n) ergeben sich aus den Eingangssignalen a_1, a_2 usw.:

$$o_i = f_i\,(a_1, a_2,...) \qquad (3.17)$$

Die Ausgangssignale b_1, b_2 usw. werden wiederum aus den OHE-Signalen gebildet:

$$b_j = g_j\,(o_1, o_2,.., o_n) \qquad (3.18)$$

Setzt man in (3.18) anstelle der OHE-Variablen (o_1, o_2 usw.) die f_i-Ausdrücke aus den Gleichungen (3.17) ein, ergeben sich die direkten Abhängigkeiten der Ausgänge von den Eingängen, die zumeist schaltalgebraisch vereinfacht werden können. Der universelle Codewandler ist der adressierbare Speicher. Für a Eingangs- und b Ausgangssignale wird eine Anordnung mit einer Speicherkapazität von 2^a Worten benötigt, die b Bits lang sind. Das eingangsseitige Bitmuster wird als Adresse an den Speicher angelegt, der die Wandlungstabelle enthält.

Das allgemeine Schaltnetz ist eigentlich ein Codewandler

Jedes Schaltnetz beruht im Grunde auf Decodier- und Codierschaltungen. Werden die Belegunge der Eingangs- und Ausgangssignale als Codewörter aufgefasst, so leistet das Schaltnetz nichts anderes als eine Umcodierung. Die UND-Gatter bilden den Decoder. Sie decodieren alle Eingangsbelegungen, die die betreffende Schaltgleichung erfüllen (mit anderen Worten, die zur Eins-Menge gehören). Die ODER-Gatter entsprechen dem Codierer, der die ausgangsseitige Codebelegung bildet.

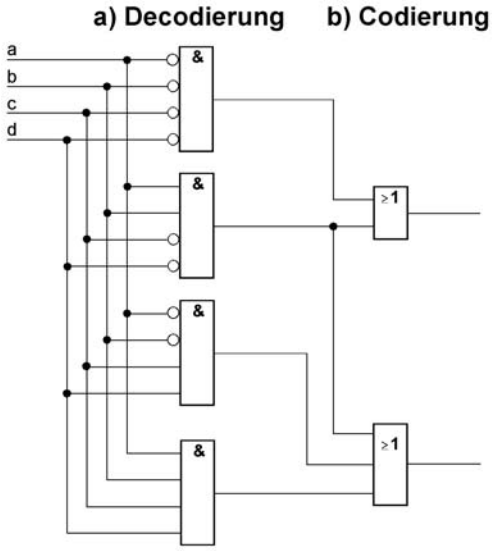


Abb. 3.35 Das Schaltnetz als Codewandler (Beispiel). a) Decodierung aller zur Eins-Menge gehörenden eingangsseitigen Signalbelegungen mit UND-Gattern; b) Codieren der ausgangsseitigen Signalbelegungen mit ODER-Gattern.

3.4 Auswählen

Die Aufgabe besteht darin, einen von mehreren Eingängen zu einem Ausgang durchzuschalten. Das heißt: am Ausgang muss eine Eins erscheinen, wenn der betreffende Eingang eine Eins führt UND wenn er ausgewählt ist. Man muss also jeden Eingang mit einem Auswahlsignal konjunktiv verknüpfen und alle diese Verknüpfungen disjunktiv zusammenfassen. Die Auswahlschaltungen unterscheiden sich darin, wie die Auswahlsignale codiert sind. Es gibt zwei Grundschaltungen:

d) Datenselektor. Für jeden Datenweg ist ein einzelnes Auswahlsignal vorgesehen (1-aus-n-Auswahl).
e) Multiplexer. Die Belegung der Auswahlsignale entspricht dem Binärcode (Auswahladresse).

Eine Auswahlschaltung kann mehrere gemeinsam auszuwählende Bitpositionen umfassen. Im Beispiel von Abb. 3.36 gibt es vier eingangsseitige Signalwege A, B, C, D. Jedem dieser Signalwege ist ein Auswahlsignal zugeordnet (SELECT A, SELECT B usw.). Je Bitposition ist eine UND-ODER-Schaltung vorgesehen. Ist beispielsweise das Auswahlsignal SELECT A aktiv, so werden die Eingänge A_0, A_1 usw. zu den Ausgängen S_0, S_1 usw. durchgeschaltet. Sind mehrere Auswahlsignale gleichzeitig aktiv, so ergibt sich am Ausgang eine ODER-Verknüpfung der jeweils ausgewählten Eingangsbelegungen. Das ist manchmal gewünscht, manchmal ein Fehler (inkorrekte Ansteuerung).

Der Multiplexer ist eine Kombination aus Adressdecoder und Datenselektor. Um einen von n Eingängen auszuwählen, braucht man ld n Auswahlsignale (Auswahladresse). Deren

Belegung muss decodiert werden, um die UND-Gatter des Datenselektors anzusteuern. Typische Multiplexer ermöglichen es, einen von 2, 4, 8 oder 16 Eingängen auszuwählen. Übliche Bezeichnungen: 2-zu-1 (2 to 1), 4-zu-1 (4 to 1) usw. In Abb. 3.36b sind zwei Bitpositionen eines 4-zu-1-Multiplexers dargestellt.

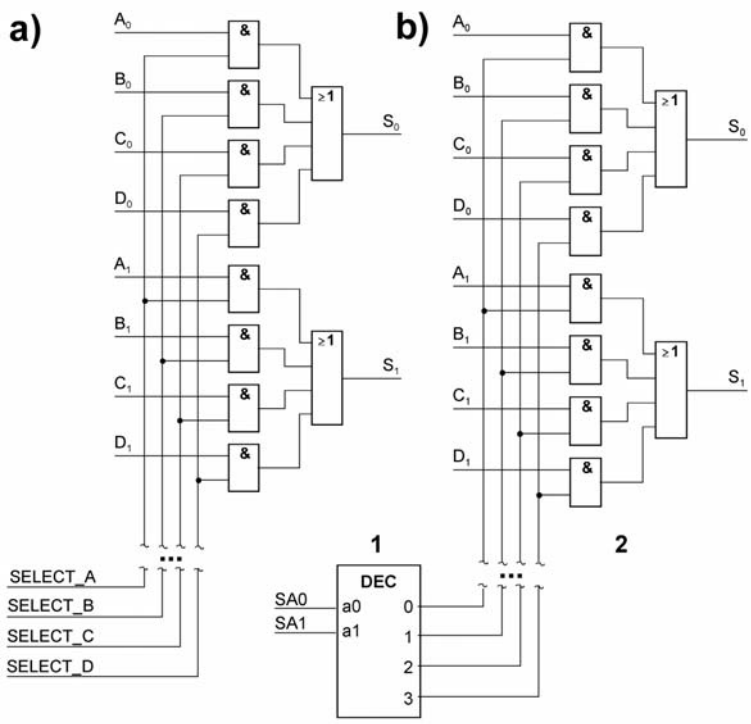


Abb. 3.36 Auswahlschaltungen. a) Datenselektor; b) Multiplexer.1 - Adressdecoder; 2 - Datenselektor.

Generische Schaltungen

Der einfachste Datenselektorist ein UND-ODERNetzwerk mit der Schaltfunktion

$$a \cdot SELECT\_a \vee b \cdot SELECT\_b .$$

Sind mehr als zwei Signale auszuwählen, so wird je Signal eine UND-Verknüpfung hinzugefügt, und die ODER-Verknüpfung wird entsprechend erweitert (z. B. durch Kaskadierung).

Der einfachste Multiplexer ergibt sich aus dem Datenselektor durch Ansteuerung mit einem einzigen Auswahlsignal A0, das mit dem einen Datensignal invertiert und mit dem anderen direkt verknüpft wird (Auswahladresse):

$$D0 \cdot \overline{A0} \vee D1 \cdot A0$$

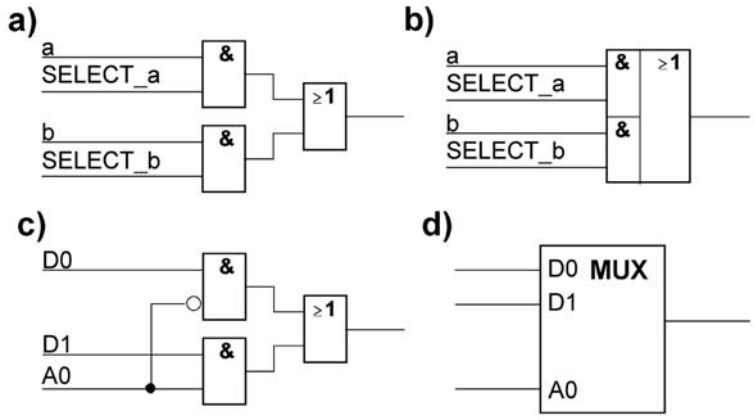


Abb. 3.37 Einfachste Datenselektoren und Multiplexer. a) Datenselektor als UND-ODER-Verknüpfung; b) zusammengefasstes Schaltsymbol nach DIN 40 900; c) Multiplexer 2 zu 1; d) Schaltsymbol (vereinfacht).

Erlaubnis- oder Steuereingänge

Gelegentlich ist es von Vorteil, zusätzliche Erlaubnis- oder Steuereingänge einzuführen, um den Ausgang freizugeben oder zu sperren. Die Wirkung hängt vom Typ der Ausgangsstufe ab:

- Zweiwertige Ausgangsstufe (Gegentaktschaltung): ein direkter (nicht-invertierter) Ausgang verharrt auf Low, ein invertierter Ausgang auf High.
- Tri-State-Ausgangsstufe: der Ausgang wird hochohmig.
- Open Collector / Drain / Emitter: der Ausgang wird inaktiv.

Tri-State-Typen sind vor allem zum Anschließen an Bussysteme vorgesehen. Der Einsatz in rein zweiwertigen Schaltungen ist nur dann unproblematisch, wenn das Erlaubnissignal stets aktiv gehalten wird.

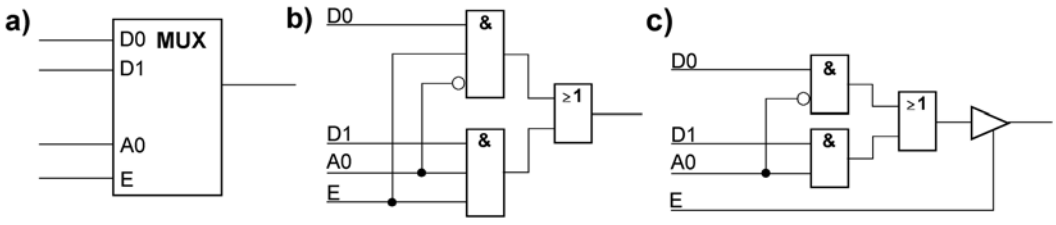


Abb. 3.38 Einfachste Multiplexer mit Erlaubniseingang (E). a) Schaltsymbol (vereinfacht); b) mit zweiwertigem Ausgang (wird Low, wenn E = Low); c) mit Tri-State-Ausgang (wird hochohmig, wenn E = Low).

Der Multiplexer mit mehr als zwei Dateneingängen ist ein Datenselektor, dessen Auswahlsignale an einen Adressdecoder angeschlossen sind. Aus Geschwindigkeitsgründen werden in vielen Multiplexerschaltungen die Decodier- und Auswahlgatter (es sind beides UND-Verknüpfungen) zusammengefasst (Verringerung der Schaltungstiefe).

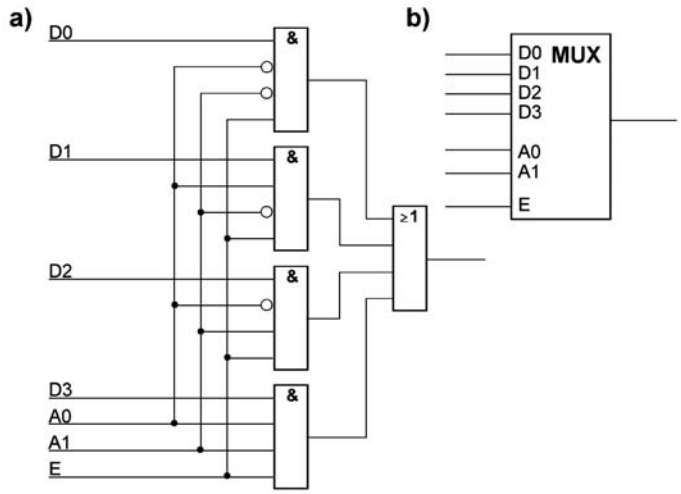


Abb. 3.39 Multiplexer mit vier Dateineingängen (4 zu 1) und Erlaubniseingang (E). a) Schaltung; b) Schaltsymbol (vereinfacht).

Multiplexer kaskadieren

Es ist eine größere Anzahl von Signalen auszuwählen. Hat der einzelne Multiplexer 2^k Eingänge, so sind für maximal 2^n Signale ($n > k$) 2^{n-k} Multiplexer erforderlich. Sie sollen im folgenden als Multiplexer der ersten Ebene bezeichnet werden. Deren Ausgänge werden wiederum von Multiplexern der zweiten Ebene ausgewählt usw. Die niedrigstwertigen Bitpositionen der Auswahladresse werden an die Multiplexer der ersten Ebene angeschlossen, die nächst-höherwertigen an die der zweiten Ebene usw.

Wenn die Anzahl der Dateneingänge keine Zweierpotenz ist

Es gibt zwei Alternativen:

- Wahl eines Typs mit der nächst-höheren Anzahl an Dateneingängen. Die ungenutzten Eingänge werden mit einem Festwert beschaltet.
- Kaskadierung mit Multiplexern, die weniger Eingänge haben; im Extremfall mit 2-zu-1-Typen. Mit diesen kann man beliebige Multiplexeranordungen ohne “Verschnitt” (= ungenutzte Eingänge) aufbauen.

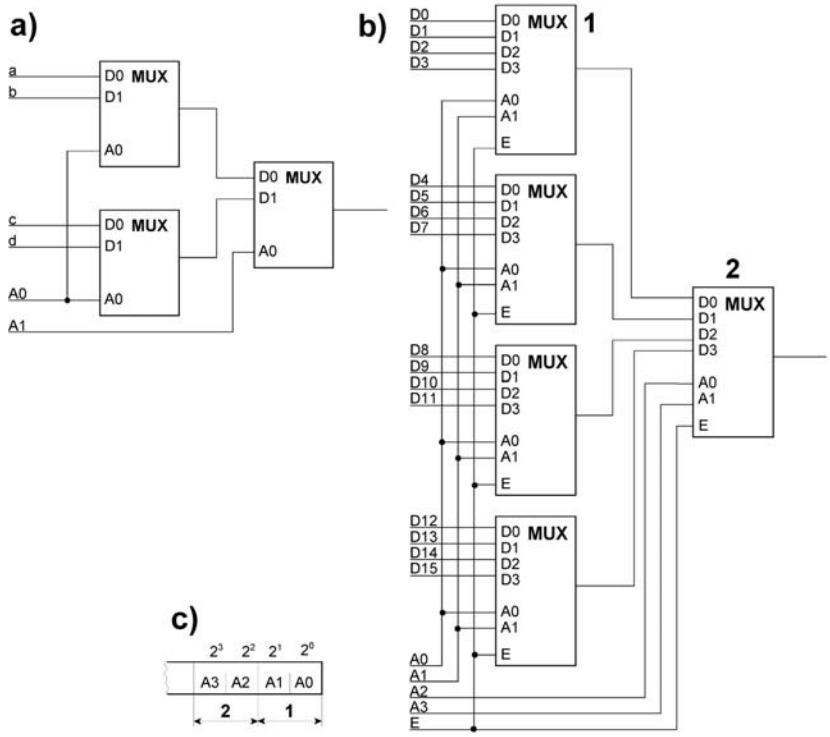


Abb. 3.40 Kaskadierung von Multiplexern (Multiplexer-Pyramide). a) mit 2-zu-1-Multiplexern. Aus diesen einfachen Multiplexern können durch Kaskadierung Auswahlnetzwerke für beliebig viele Signale aufgebaut werden. b) mit 4-zu-1-Multiplexern; c) die Aufteilung der Auswahladresse von b) auf die beiden Ebenen. 1 - erste Ebene (hier werden die auszuwählenden Signale (D15...D0) angeschlossen); 2 - zweite Ebene (wählt jeweils einen der Ausgänge der ersten Ebene aus).

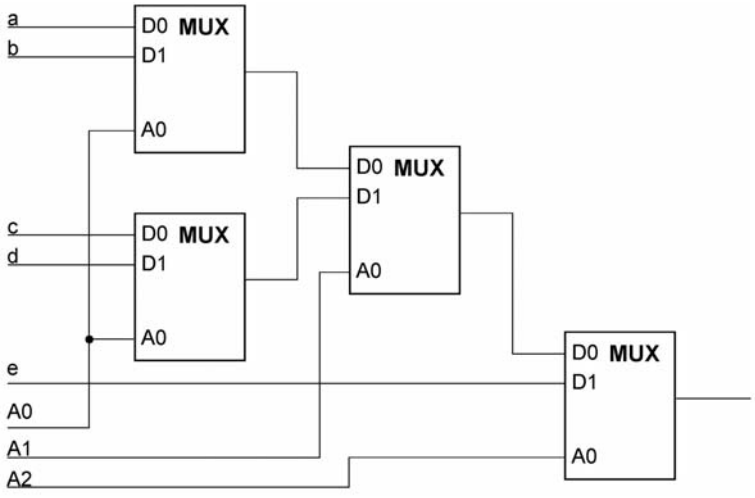


Abb. 3.41 Mit 2-zu-1-Multiplexern kann man Auswahlschaltungen mit beliebig vielen Dateneingängen aufbauen. Hier ein 5-zu-1-Multiplexer. Die Adressbelegungen 100B bis 111B bewirken, dass der Eingang e ausgewählt wird.

Multiplexer mit Schalterbauelementen
Abb. 3.42 zeigt einen 8-zu-1-Multiplexer mit FET-Schaltern. Die Besonderheiten:

- Die Signallaufzeit zwischen ausgewähltem Eingang und Ausgang ist praktisch vernachlässigbar (z. B. maximal 250 ps).
- Die Signale können in beide Richtungen fließen (bidirektionaler Betrieb). Damit ist der Schaltkreis auch als Verteiler (Demultiplexer) einsetzbar.
- Die Ausgänge werden nicht aktiv getrieben; somit muss die Treibfähigkeit von den jeweils eingangsseitig vorgeordneten Schaltkreisen erbracht werden.
- In bestimmten Spannungsbereichen eignen sich solche Bauelemente auch als Analogschalter.

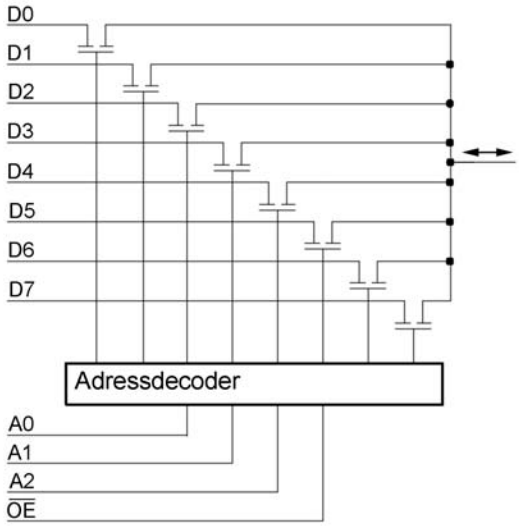


Abb. 3.42 Multiplexer als Schalterbauelement (nach [[2.36] bis [2.39]). A0...A2 - Auswahladresse; OE - Steuersignal (sperrt oder erlaubt Datendurchleitung).

3.5 Abfragen und Verteilen (Multiplexing / Demultiplexing)

Der Ursprung dieser Begriffsbildungen liegt im Prinzip der Zeitmultiplexübertragung. Der Grundgedanke: anstelle vieler einzelner Signalwege einen einzigen vorzusehen, über den alle Signale nacheinander übertragen werden. Naheliegende Anwendungsbeispiele sind u. a. Signalwege in Flugzeugen und Chemieanlagen (zwischen den Motoren, Messwertaufnehmern usw. und der Kanzel oder dem Leitstand). Hierfür sind zwei Funktionen zu erbringen:

- Auf der Sendeseite (an den Signalquellen) sind die Signale nacheinander abzufragen, und die abgefragten Werte sind auf den Signalweg zu geben (Serialisierung, Multiplexing).
- Auf der Empfangsseite sind die nacheinander ankommenden Signalwerte auf die jeweiligen Einrichtungen (Anzeigen, Messwerke, Rechengeräte, Stellglieder usw.) zu verteilen (Deserialisierung, Demultiplexing).

Der Demultiplexer

Die Abfrage (Serialisierung) ist mit den bereits erläuteren Datenselektoren und Multiplexern zu erledigen. Die Deserialisierung erfordert eine weitere Funktionseinheit – den Demultiplexer. Der Demultiplexer ist eine Verteilerschaltung. Im einfachsten Fall wird die Belegung eines einzigen Eingangs auf einen von mehreren Ausgängen gegeben. Das läuft offensichtlich auf konjunktive Verknüpfungen hinaus: An einem Ausgang muss dann eine Eins erscheinen, wenn der Eingang mit Eins belegt UND wenn der betreffende Ausgang ausgewählt ist. Die Verteilerschaltung ist das Gegenstück zum Datenselektor. Der binär adressierten Demultiplexer als Gegenstück zum Multiplexer ergibt sich, wenn man die Weitergabesteuersignale an einen Adressdecoder anschließt. Es liegt nahe, die UND-Verknüpfungen der Decodier- und der Weitergabefunktion in jeweils einem Gatter zusammenzufassen. Diese Schaltung ist aber nichts anderes als ein Decoder mit Erlaubniseingang. Man kann jeden Decoder mit Erlaubniseingang als Demultiplexer einsetzen, indem man den Erlaubniseingang als Signaleingang verwendet.

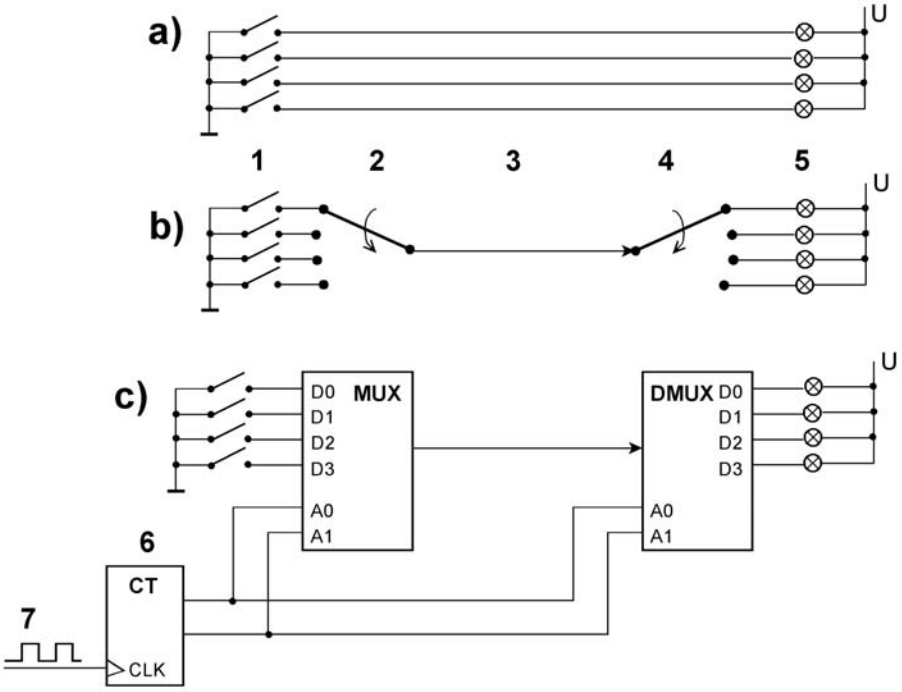


Abb. 3.43 Abfragen und Verteilen (Multiplexing / Demultiplexing; Serialisierung / Deserialisierung) am einfachen Anwendungsbeispiel. 1 - Signalquellen; 2 - Abfrage (Multiplexing); 3 - Signalübertragung; 4 - Verteilung (Demultiplexing); 5 - Signalausgabe; 6 - Abfragezähler; 7 - Abfragetakt. a) das Anwendungsproblem. Werden die Signalquellen 1 direkt mit der Signalausgabe 2 verbunden (parallele Übertragung), so braucht man viele Leitungen. b) Grundsatzlösung der Übertragung über eine einzige Signalleitung (serielle Übertragung). Die Signalquellen werden nacheinander abgefragt (Serialisierung, Multiplexing). Am anderen Ende werden die übertragenen Signale zwecks Ausgabe wieder verteilt (Deserialisierung, Demultiplexing). c) Schaltungslösung mit Multiplexer und Demultiplexer, die beide von einem Abfragezähler 6 adressiert werden.

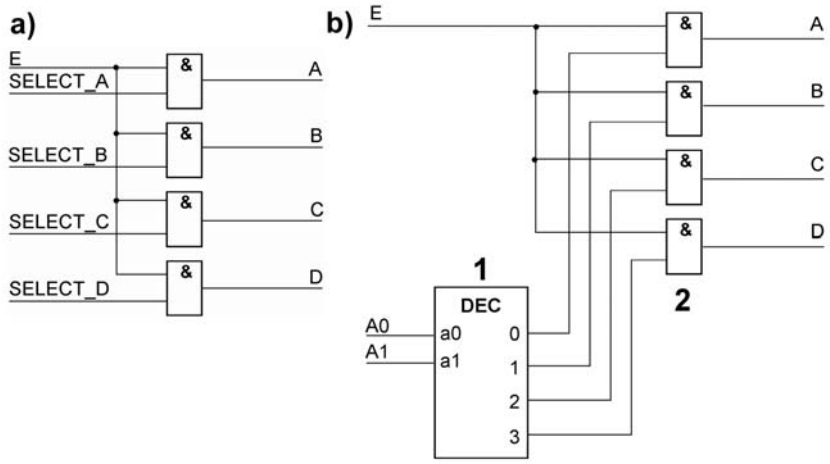


Abb. 3.44 Verteilerschaltungen. a) eine einfache Verteilerschaltung. E - Dateneingang; A…D - Ausgänge; SELECT_A…SELECT_D - Weitergabesteuersignale. b) Demultiplexer 1-zu-4. 1 - Adressdecoder; 2 - Verteiler.

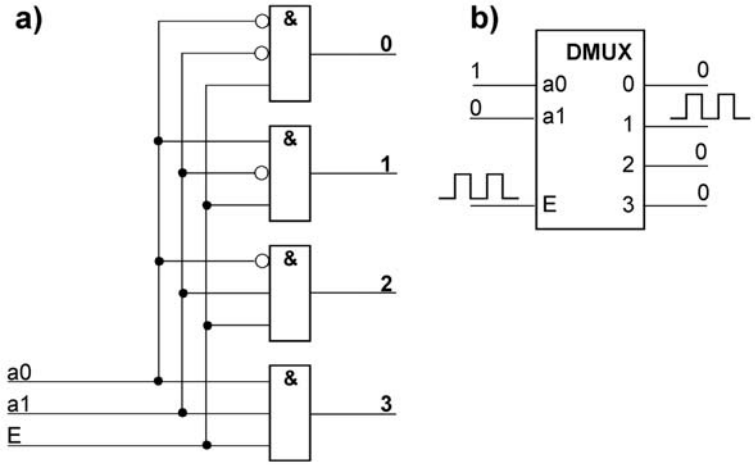


Abb. 3.45 Der Demultiplexer ist nichts anderes als ein Decoder mit Erlaubniseingang. Der Erlaubniseingang dient hier als Signaleingang. a) Demultiplexer mit zusammengefassten UND-Verknüpfungen; b) Schaltsymbol (vereinfacht) mit Betriebsbeispiel (Weitergabe eines Eingangssignals zum Ausgang 1).

3.6 Universelle Logik

Universelle Logikschaltungen sind Funktionselemente, die verschiedene kombinatorische Verknüpfungen darstellen können. Die jeweils auszuführende Verknüpfungen ergeben sich durch entsprechendes Belegen von Anschlüssen und Speicherzellen.

Universelle kombinatorische Schaltungen

Die Aufgabe besteht darin, universelle Schaltungen anzugeben, die für beliebige anwendungsspezifische Verknüpfungen eingerichtet werden können, ohne dass hierzu Fertigungsschritte der Halbleitertechnologie erforderlich sind. Die theoretische Grundlage ist der Erweiterungssatz von Boole und Shannon. Aus (1.62) lässt sich ablesen, wie ein universelles Schaltnetz verwirklicht werden kann. Es werden alle 2^n Produktterme implementiert und disjunktiv verknüpft. Jedes UND-Gatter erhält einen zusätzlichen Steuereingang. Um eine bestimmte Schaltfunktion darzustellen, ist jeder Steuereingang mit dem Funktionswert (0 oder 1) zu beschalten, der der jeweiligen Variablenbelegung zugeordnet werden soll.

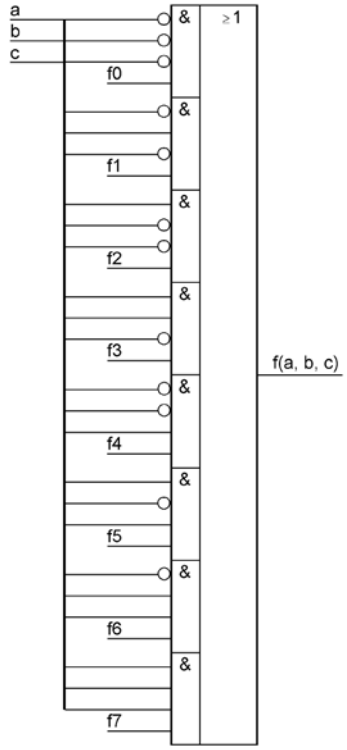


Abb. 3.46 Eine universelle kombinatorische Schaltung auf Grundlage des Entwicklungssatzes von Boole und Shannon (vgl. (1.62) auf S. ****). f0...f7 sind feste Funktionswerte, die den Produkttermen $\overline{a} \cdot \overline{b} \cdot \overline{c} \ldots a \cdot b \cdot c$ zugeordnet werden.

Es liegt nahe, nach bereits vorhandenen Funktionselementen zu suchen, in die 2^n Produktterme eingebaut sind. Das ist bei drei Grundtypen der Fall:

- beim Decoder (binär zu 1 aus n),
- beim Multiplexer,
- beim adressierbaren Speicher.

Der Decoder als Grundlage einer universellen kombinatorischen Schaltung

Jede der 2^n möglichen Eingangsbelegungen aktiviert einen der 2^n Ausgänge. Damit sind an den Ausgängen alle Produktterme verfügbar. Die zur jeweiligen Schaltfunktion gehörenden Produktterme können disjunktiv verknüpft werden. Es ist möglich, die Produktterme für mehrere Schaltfunktionen auszunutzen.

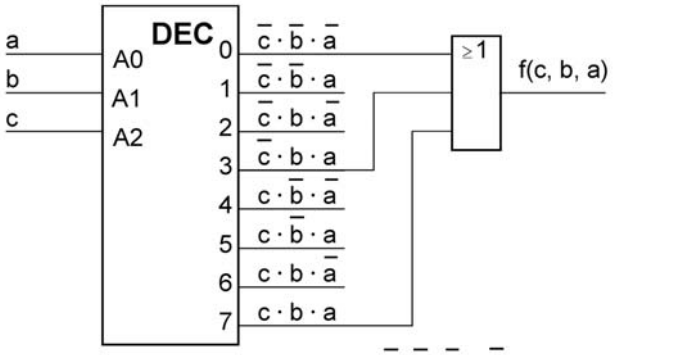


Abb. 3.47 Der Decoder als Grundlage einer universellen kombinatorischen Schaltung.

Der Multiplexer als universelles Funktionselement

Der Multiplexer ist eine Kombination aus einem Auswahlnetzwerk und einem Decoder. Ein Multiplexer mit n Adresseingängen erlaubt es, einen von 2^n Dateneingängen auf den Ausgang durchzuschalten. Somit kann man einen 2^n-zu-1-Multiplexer verwenden, um jede beliebige Schaltfunktion von n Variablen zu verwirklichen. Hierzu genügt es, die Dateneingänge mit den Festwerten 0 oder 1 beschalten, je nach den Werten in der Wahrheitstabelle.

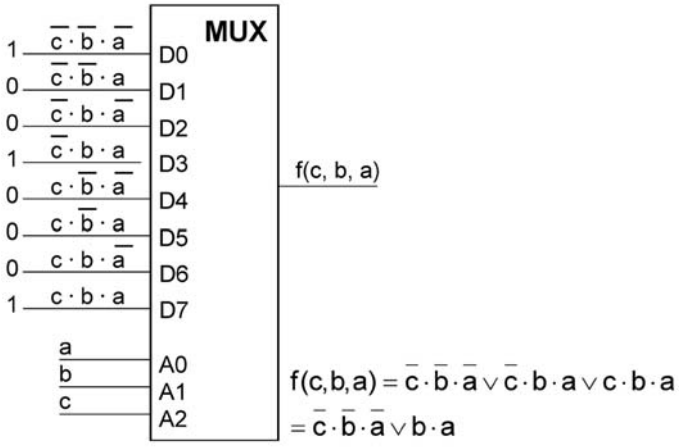


Abb. 3.48 Darstellung einer Schaltfunktion mittels Multiplexer (1).

Ein so beschalteter Multiplexer ist im Grunde ein kleiner Festwertspeicher (ROM). Geht man von der Festbeschaltung ab, kann ein Multiplexer mit n Adresseingängen beliebige Schaltfunktionen mit n + 1 Variablen darstellen. Voraussetzung dafür ist, dass die Dateneingänge mit jeweils einem von vier Werten belegt werden:

- mit dem Festwert 0,
- mit dem Festwert 1,
- mit einer Variablen,
- mit einer invertierten Variablen.

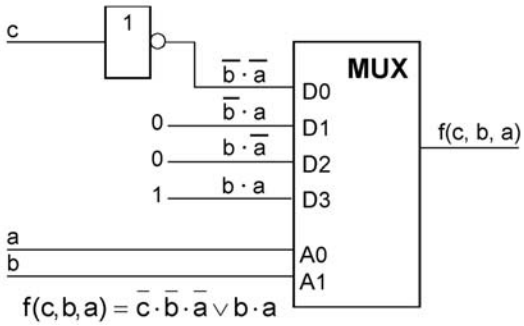


Abb. 3.49 Darstellung einer Schaltfunktion mittels Multiplexer (2).

Der adressierbare Speicher als universeller Funktionszuordner

Jede Speicherzelle wird durch Adressierung ausgewählt. n Adressbits entsprechen 2^n Speicherzellen. Die Adressdecdierung ergibt die Produktterme, der Lesesignalweg (an den alle Speicherzellen angeschlossen sind) ergibt die disjunktive Verknüpfung, der Inhalt der Speicherzellen ergibt die Zuordnung der Wahrheitswerte. Die Adresseingänge des Speichers werden mit den Variablen der Schaltfunktion belegt, die Speicherzellen entsprechend der Wahrheitstabelle mit Nullen oder Einsen gefüllt. Auf diese Weise lässt sich jede beliebige Schaltfunktion im Rahmen der gegebenen Speicherkapazität verwirklichen.

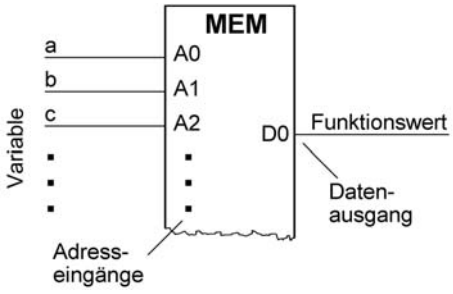


Abb. 3.50 Darstellung einer Schaltfunktion mittels eines adressierbaren Speichers.

Adressierbare Speicher sind als Speicherschaltkreise und als Funktionselemente in programmierbaren Schaltkreisen verfügbar. Die Anzahl der Variablen ist auf ca. 20...24

begrenzt. Diese Größenordnung ist aber als Extremfall anzusehen, da hierfür Speicherkapazitäten von 1M...16M benötigt werden. Die typischen Zuordnerspeicher haben Speicherkapazitäten zwischen 16 und einigen k Speicherworten (1 k = 2^{10}), so dass sie Schaltfunktionen von 4 bis ca. 12...16 Variablen aufnehmen können. Der Zuordnerspeicher ist vor allem dann das Mittel der Wahl, wenn es nicht allzu viele Variable sind, wenn die Verknüpfungen kompliziert sind und wenn mehrere Schaltfunktionen untergebracht werden müssen. Das betrifft u. a. Aufgaben der Codewandlung und des Rechnens.

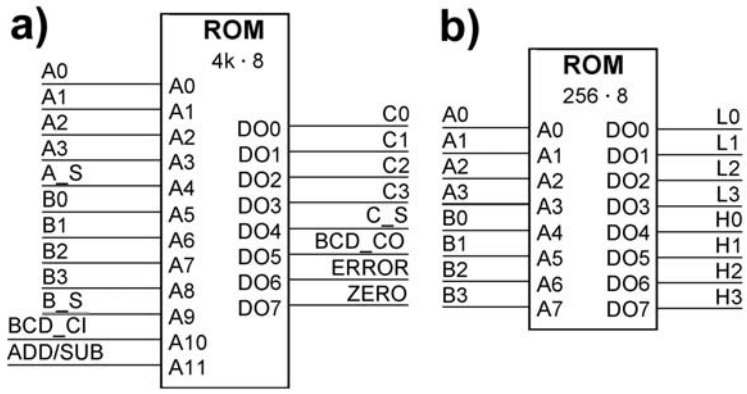


Abb. 3.51 Der Zuordnerspeicher macht Kompliziertes sehr einfach[4)]. Hier beim Rechnen mit binär codierten Dezimalzahlen. a) Addition und Subtraktion; b) Multiplikation. A3...0, B3...0 - Operanden; A_S, B_S - Operandenvorzeichen; BCD_CI - Eingangsübertrag; C3...0 - Ergebniswert; C_S - Ergebnisvorzeichen; BCD_CO - Ausgangsübertrag; ERROR - Operandenfehler (unzulässige Belegung); ZERO - Nullbedingung; L3...0 - niedere Produkttetrade; H3...0 - höhere Produkttetrade. Ein Speicher der Organisationsform 4k • 8 erledigt die Addition und Subtraktion von BCD-Ziffern einschließlich der Vorzeichen- und Übertragsbehandlung. Auch für die Fehlererkennung ist noch Platz. Für das kleine Einmaleins genügt sogar ein Speicher der Organisationsform 256 • 8.

Der Logikentwurf mit Zuordnerspeichern erscheint einfach. Es ist aber auch an den folgenden einfachen Zusammenhang zu denken:

$$\text{Ressourcenbedarf für n Variable} = O(2^n) \qquad (3.19)$$

Bei mehr als 10...12 Variablen ist typischerweise Schluss (vom ausgesprochenen Sonderlösungen abgesehen). Demgegenüber ist die Anzahl der Ausgänge (= Schaltfunktionen) weniger von Bedeutung, da der Ressourcenbedarf nur linear zunimmt (für i Ausgänge gemäß O(i)). Braucht man mehr Ausgänge, werden mehrere Funktionselemente oder Schaltkreise parallel adressiert.

4: Zumindest im Schaltungsentwurf. Es muss aber auch der Speicherinhalt bestimmt werden...