http://en.wikipedia.org/wiki/List_of_7400_series_integrated_circuits

# List of 7400 series integrated circuits

From Wikipedia, the free encyclopedia

The following is a **list of 7400 series digital logic integrated circuits**. The SN7400 series originated with TTL integrated circuits made by Texas Instruments. Because of the popularity of these parts, they were second-sourced by other manufacturers who kept the 7400 sequence number as an aid to identification of compatible parts. As well, compatible TTL parts originated by other manufacturers were second sourced in the TI product line under a 74xxx series part number.

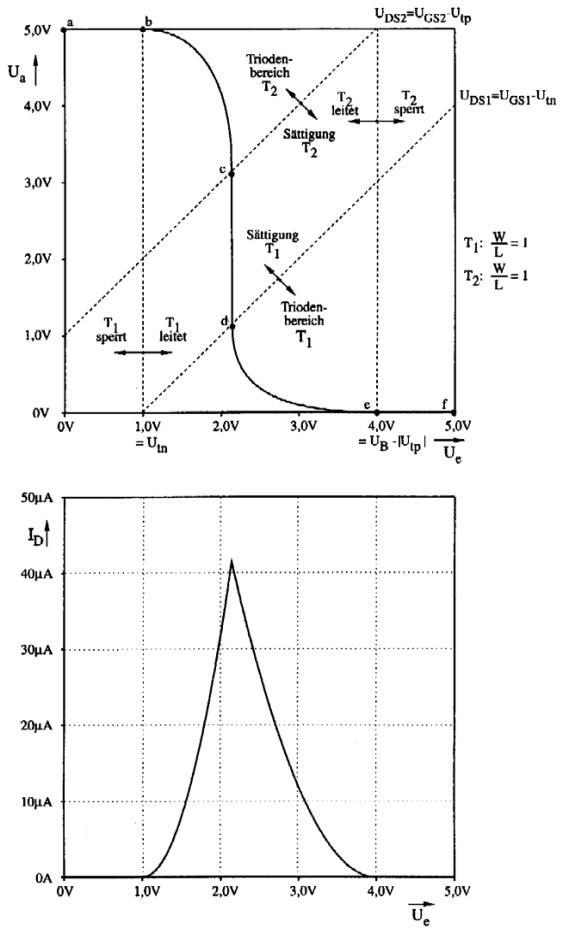
Just the *base numbers* are listed below, that is: parts are listed here as if made in the basic, standard power and speed, TTL form, although many later parts were never manufactured with that technology.

| Part number | Description | Datasheet |
|---|---|---|
| 7400 | quad 2-input NAND gate | HC/HCT |
| 741G00 | single 2-input NAND gate | |
| 7401 | quad 2-input NAND gate with open collector outputs | |
| 741G01 | single 2-input NAND gate with open drain output | |
| 7402 | quad 2-input NOR gate | HC/HCT |
| 741G02 | single 2-input NOR gate | |
| 7403 | quad 2-input NAND gate with open collector outputs | HC/HCT |
| 741G03 | single 2-input NAND gate with open drain output | |
| 7404 | hex inverter | HC/HCT |
| 741G04 | single inverter | |
| 7405 | hex inverter with open collector outputs | HC |
| 741G05 | single inverter with open drain output | |
| 7406 | hex inverter buffer/driver with 30 V open collector outputs | |

CMOS- Eingänge sind sehr empfindlich gegenüber statischer Aufladung! CMOS Ein- und Ausgänge müssen abgeschlossen sein. (Eingang hat undefinierten Zustand) auch unbenutzte Gatter eines Chips stets auf wohldefiniertes Potential bringen! (Verlustleistung!)

Bereiche:

- a-b → T1 sperrt
- e-f → T2 sperrt
- c-d → T1, T2 in Sättigung





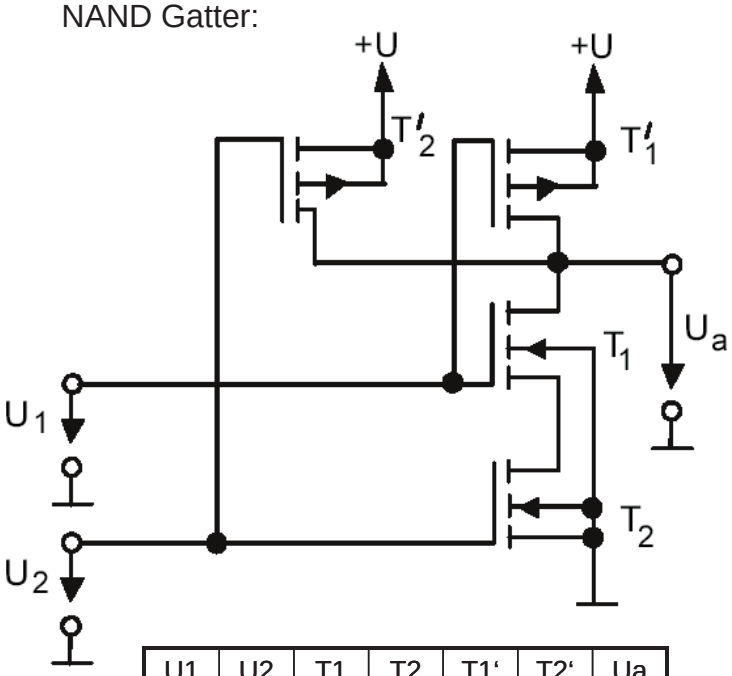


**Elektronik II 23.04.2018 E. Riedle**

Abbildung 3.31: Übertragungskennlinie und Stromaufnahme eines CMOS–Gatters

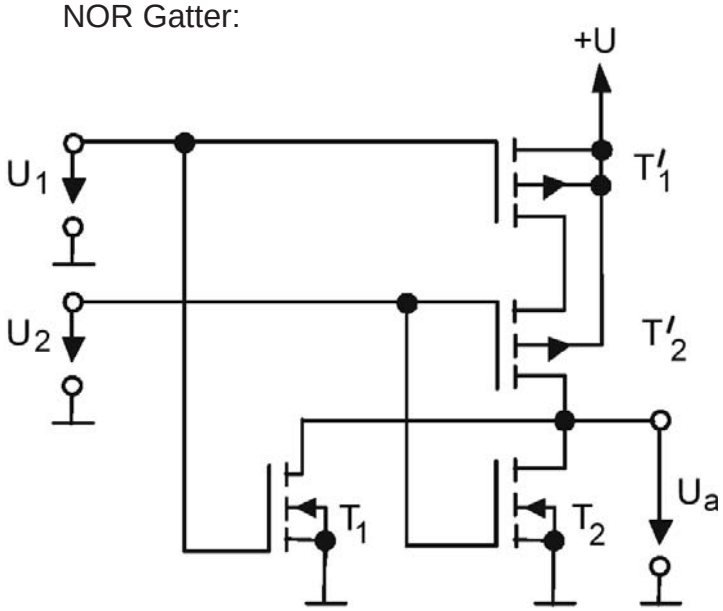
Beispiel: CMOS „NAND“ und CMOS „NOR“

NAND Gatter:




| U1 | U2 | T1 | T2 | T1‘ | T2‘ | Ua |
|---|---|---|---|---|---|---|
| 0 | 0 | s | s | l | l | 1 |
| 0 | 1 | s | l | l | s | 1 |
| 1 | 0 | l | s | s | l | 1 |
| 1 | 1 | l | l | s | s | 0 |

NOR Gatter:




| U1 | U2 | T1 | T2 | T1‘ | T2‘ | Ua |
|---|---|---|---|---|---|---|
| 0 | 0 | s | s | l | l | 1 |
| 0 | 1 | s | l | l | s | 0 |
| 1 | 0 | l | s | s | l | 0 |
| 1 | 1 | l | l | s | s | 0 |



**High-speed-CMOS (HC):**

Anfang der 80er Jahre kam diese Familie auf den Markt

- Verringern von Streukapazitäten
- Verringerung des Flächenbedarfs durch Ionenimplantation
- Geringere Spannungsfestigkeit (max. 6V)

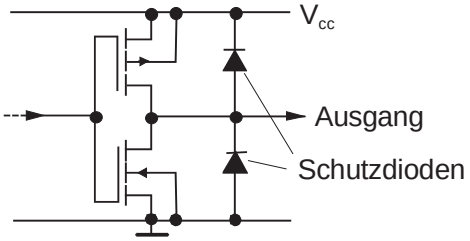
Betriebsspannung kann 2-6 V betragen (batteriebetriebene Geräte)

Low: 0V –20% der Betriebsspannung
High: 70%-100% der Betriebsspannung

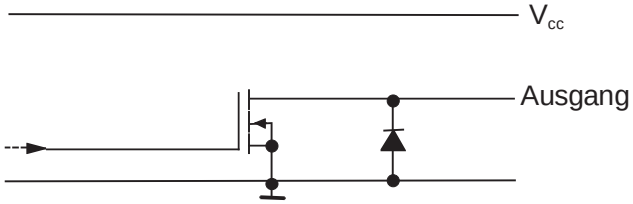
Open drain Ausgang zur Pegelumsetzung! (0 oder hochohmig, $R_{pullup}$)

**HCT-Familie:** TTL-kompatibler Eingang

a) Typische Ausgangsstufe der HCMOS - Bauelemente




c) Open - Drain - Ausgang




(da TTL worst case kleiner als 70% der Betriebsspannung sein kann)

→ Ankopplung der Eingangs-FET über Diode an Betriebsspannung

## III. Boolesche Algebra und logischer Schaltkreisentwurf:

Logik:

- Alle Aussagen sind entweder wahr oder falsch:

mathematische Beschreibung: 1 oder 0

*George Boole (1815- 1864), Queens College (Cork)*
*"The mathematical analysis of logic" (1847)*
*"An investigation of the laws of thought" (1854)*

--> Boole´sche Algebra (Schaltalgebra)

Aufgabe der Boole´schen Algebra:

Formulierung von Bedingungen in einem zweiwertigen System und daraus logische Entscheidungen zu treffen.

- Diese mathematischen Verknüpfungen führen schaltungstechnisch zu bestimmten logischen Bausteinen, die die Operationen ausführen.
- Logische Bausteine: Haben immer einen oder mehrere Eingänge, aber nur einen Ausgang.



**Boolesche Konstante:**

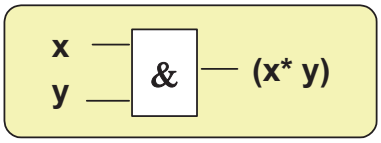
In der Booleschen Algebra gibt es nur zwei Konstanten (0, 1). Technisch muß diesen Konstanten ein eindeutiger Wert (z.B. Potential) zugeordnet werden.

**Boolesche Variable:**

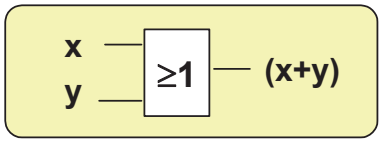
A= 0, 1

**Boolesche Operation:**

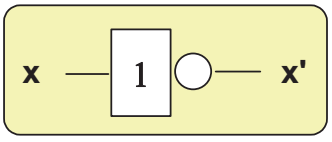
AND, OR, NOT




AND




OR




NOT

x = A * B * C * ... * N
Konvention auch: A * B * C = ABC

Die AND Operation ist kommutativ
AB = BA

Die AND Operation ist assoziativ
A(BC) = (AB)C = B(AC)

x = A + B + C

Die OR Operation ist kommutativ
A + B + C = A + C + B = B + C + A

Die OR Operation ist assoziativ
A + B + C = A + (B + C) = (A + C) + B

Wirkt auf einzelne Konstante wie Vorzeichenwechsel

$\overline{1} = 0$

$\overline{0} = 1$

## Grundlegende Rechenregeln der Boole´schen Algebra

| | |
|---|---|
| $A + 0$ | $= A$ |
| $A + 1$ | $= 1$ |
| $A + A$ | $= A$ |
| $A + \overline{A}$ | $= 1$ |
| $A * A$ | $= A$ |
| $A * 1$ | $= A$ |
| $A * 0$ | $= 0$ |
| $A * \overline{A}$ | $= 0$ |
| $A$ | $= \overline{\overline{A}}$ |

### Gesetze von De Morgan:

1.) $\overline{A + B + C} = \overline{A} \cdot \overline{B} \cdot \overline{C}$

2.) $\overline{A \cdot B \cdot C} = \overline{A} + \overline{B} + \overline{C}$

Beispiele

| A | B | A + B | $\overline{A + B}$ | $\overline{A} * \overline{B}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 |

| A | B | A * B | $\overline{A * B}$ | $\overline{A} + \overline{B}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |



### Negative und positive Logik:

"0" und "1" werden in der Praxis bestimmten Spannungen zugeordnet.

=> positive Logik: U (1) > U (0)
=> negative Logik: U (0) > U (1)

==> Damit kann ein und derselbe Baustein eine OR oder eine AND Funktion ausführen
(nur Definition der Art der Logik ist wichtig!)

Beispiel: Sei N der negative Wert, P der positive Wert der Spannungen

| | | | positive Logik OR | | | negative Logik AND | | |
|---|---|---|---|---|---|---|---|---|
| A | B | x | A | B | x | A | B | x |
| N | N | N | 0 | 0 | 0 | 1 | 1 | 1 |
| P | N | P | 1 | 0 | 1 | 0 | 1 | 0 |
| N | P | P | 0 | 1 | 1 | 1 | 0 | 0 |
| P | P | P | 1 | 1 | 1 | 0 | 0 | 0 |

Einer AND Funktion in negativer Logik entspricht eine OR Funktion in positiver Logik!

Übergang positive Logik <----> negative Logik durch Inversion (NOT)

Mit der Negation kann man einen Zusammenhang zwischen Konjunktion (AND) und Disjunktion (OR) herstellen, entsprechend den Gesetzen von De Morgan.

**De Morgan´sches Theorem:**

- NOR ist identisch mit AND bei invertierten Eingangsgrößen bzw. NAND Logik ist identisch zu OR Logik bei invertierten Eingangsgrößen!

Inversion durch Substitution:
Falls bei längeren Ausdrücken das De Morgan´sche Theorem nicht direkt anwendbar ist => Substitution Boolescher Ausdruck (oder Teil davon) so, daß das De Morgan´sche Theorem anwendbar wird:

Beispiele:

$$\overline{A+BCD} = \overline{A+S} = \overline{A}\cdot\overline{S}$$
$$dann \quad \overline{A}\cdot\overline{S} = \overline{A}\cdot(\overline{BCD}) = \overline{A}\cdot\left(\overline{B}+\overline{C}+\overline{D}\right)$$

$$\overline{ABC+A\overline{C}+\overline{C}D+\overline{A}\overline{D}} = \overline{S+T+U+V} = \overline{S}\cdot\overline{T}\cdot\overline{U}\cdot\overline{V} =$$
$$\overline{ABC}\cdot\overline{A\overline{C}}\cdot\overline{\overline{C}D}\cdot\overline{\overline{A}\overline{D}} = \left(\overline{A}+\overline{B}+\overline{C}\right)\cdot\left(\overline{A}+C\right)\cdot\left(C+\overline{D}\right)\cdot(A+D)$$



Damit kann ein Gleichungssystem an die gegebenen Voraussetzungen angepasst werden:

- Vorgabe der Bauelemente (Konjunktion oder Disjunktion)
- Vorgabe der Eingangsvariable oder der Ausgangsvariable (negiert, nicht negiert)

**Erweiterung der Rechenregeln:**

Multiplikation zweier Ausdrücke:

A ( B + C ) = AB + AC

A + AB = A

A ( A + B ) = A

A + ( $\overline{A}$ B ) = A + B

( A + $\overline{B}$ ) B = AB

Wegen "+" und "*" haben Boolesche Ausdrücke formal die Gestalt von Summen oder Produkten. Um Verwechslungen mit der "normalen" Algebra zu vermeiden bezeichnet man die **Summe als disjunktive Form** und das **Produkt als konjunktive Form**.

Seien A, B, C, ..., N Variable der Booleschen Algebra. Alle Terme, die AND Verknüpfungen enthalten heißen **Minterme** oder **Vollkonjunktionen**. Alle Terme, die OR Verknüpfungen enthalten heißen **Maxterme** oder **Volldisjunktionen**.

Mit N Variablen können $2^N$ Minterme gebildet werden.

Jede Form, die **alle** Variablen enthält, heißt **Normalform**."

**Normalform:**

Jede logische Funktion läßt sich auf zwei Grundformen zurückführen:

1. die disjunktive Normalform (Boolesche Summe, Standard Sum)
2. die konjunktive Normalform (Boolesches Produkt, Standard Product)



Die disjunktive Normalform entsteht dadurch, daß alle UND-Verknüpfungen, bei denen die Ausgangsvariable X den Wert 1 annimmt (Minterme), disjunktiv (mit ODER) verknüpft werden.

In der konjunktiven Normalform werden alle ODER-Verknüpfungen der negierten Eingangs-variablen, die zum Funktionswert X = 0 führen (Maxterme), konjunktiv (mit UND) verknüpft.

Beispiel: Exklusive OR

| A | B | XOR | $\overline{XOR}$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | => $\overline{A}\ \overline{B}$ |
| 0 | 1 | 1 | 0 | => $\overline{A}$ B |
| 1 | 0 | 1 | 0 | => A $\overline{B}$ |
| 1 | 1 | 0 | 1 | => A B |

Disjunktive Normalform: $X = \overline{A}\,B + A\,\overline{B}$ (XOR)

$X = \overline{A}\,\overline{B} + A\,B$ $\overline{(XOR)}$

Konjunktive Normalform: $X = (A + B) * (\overline{A} + \overline{B})$ (XOR)

$X = (A + \overline{B}) * (\overline{A} + B)$ $\overline{(XOR)}$

| A | B | C | X | Minterm | Maxterm |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | $A + B + C$ |
| 0 | 0 | 1 | 1 | $\overline{A}\ \overline{B}\ C$ | |
| 0 | 1 | 0 | 1 | $\overline{A}\ B\ \overline{C}$ | |
| 0 | 1 | 1 | 0 | | $A + \overline{B} + \overline{C}$ |
| 1 | 0 | 0 | 1 | $A\ \overline{B}\ \overline{C}$ | |
| 1 | 0 | 1 | 0 | | $\overline{A} + B + \overline{C}$ |
| 1 | 1 | 0 | 0 | | $\overline{A} + \overline{B} + C$ |
| 1 | 1 | 1 | 1 | $A\ B\ C$ | |

Die disjunktive Normalform lautet in diesem Fall:

$$X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC \quad \text{(disj. Normalform)}$$

Die konjunktive Normalform lautet:

$$X = (A + B + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C) \quad \text{(konj. Normalform)}$$

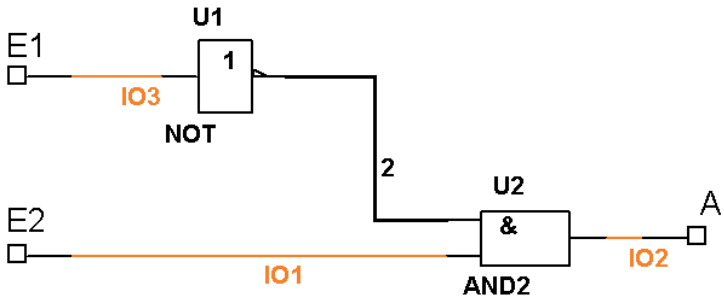


**Schaltungsanalyse:**

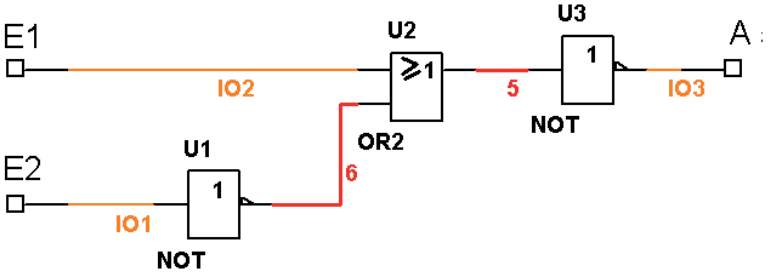
Bei der Analyse eines Schaltnetzes ist die logische Funktion gesucht!

Die Verknüpfung, die erzeugt werden soll, ist die sogenannte **Soll-Verknüpfung.**
Die Verknüpfung, welche die Schaltung tatsächlich erzeugt, wird **Ist-Verknüpfung** genannt.
Die Soll-Verknüpfung ermittelt man anhand einer Wahrheitstabelle:

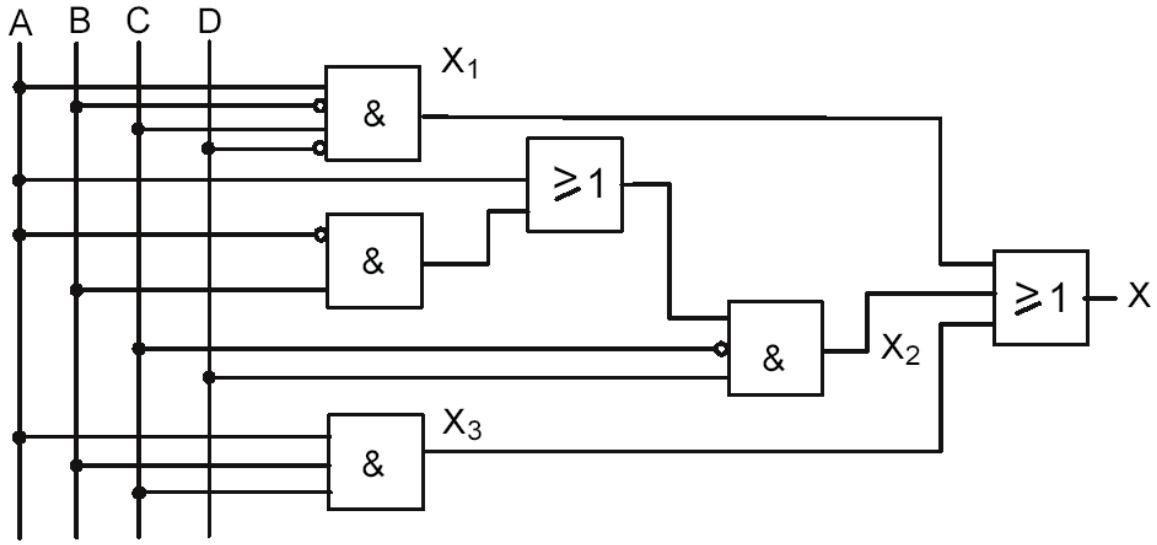
| E1 | E2 | Z= $\overline{E1}$ | A=E2*Z |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |




| E1 | E2 | Z= $\overline{E2}$ | X=E1+Z | A= $\overline{X}$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |




Beide Schaltungen ergeben die gleiche Sollverknüpfung wieder!




Zur Aufstellung der Wahrheitstafel lesen wir folgende Verknüpfungen ab:

$$X = X_1 + X_2 + X_3$$

$$X_1 = A\ \overline{B}\ C\ \overline{D}$$

$$X_2 = (A + (\overline{A}\ B))\ \overline{C}\ D$$

$$X_3 = A\ B\ C$$



Die weitere Rechnung hat zum Ziel, alle Min-Terme zu finden. Dazu werden Terme, die nicht alle Variablen enthalten mit der 1-Funktion erweitert. Der Term $X_3$ ist z.B. von $D$ unabhängig. Die notwendige Erweiterung lautet:

$$X_3 = A\ B\ C = A\ B\ C\ (D + \overline{D}) = A\ B\ C\ D + A\ B\ C\ \overline{D}$$

$$X_2 = (A + (\overline{A}\ B))\ \overline{C}\ D = A\ \overline{C}\ D + \overline{A}\ B\ \overline{C}\ D = A\ B\ \overline{C}\ D + A\ \overline{B}\ \overline{C}\ D + \overline{A}\ B\ \overline{C}\ D$$

$$\Rightarrow\ X = A\ \overline{B}\ C\ \overline{D} + A\ B\ \overline{C}\ D + A\ \overline{B}\ \overline{C}\ D + \overline{A}\ B\ \overline{C}\ D + A\ B\ C\ D + A\ B\ C\ \overline{D}$$

Zugehörige Wahrheitstabelle:

| A | B | C | D | X | Minterm |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 1 | $\overline{A}$ B $\overline{C}$ D |
| 0 | 1 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | 1 | A $\overline{B}$ $\overline{C}$ D |
| 1 | 0 | 1 | 0 | 1 | A $\overline{B}$ C $\overline{D}$ |
| 1 | 0 | 1 | 1 | 0 | |
| 1 | 1 | 0 | 0 | 0 | |
| 1 | 1 | 1 | 0 | 1 | A B C $\overline{D}$ |
| 1 | 1 | 0 | 1 | 1 | A B $\overline{C}$ D |
| 1 | 1 | 1 | 1 | 1 | A B C D |



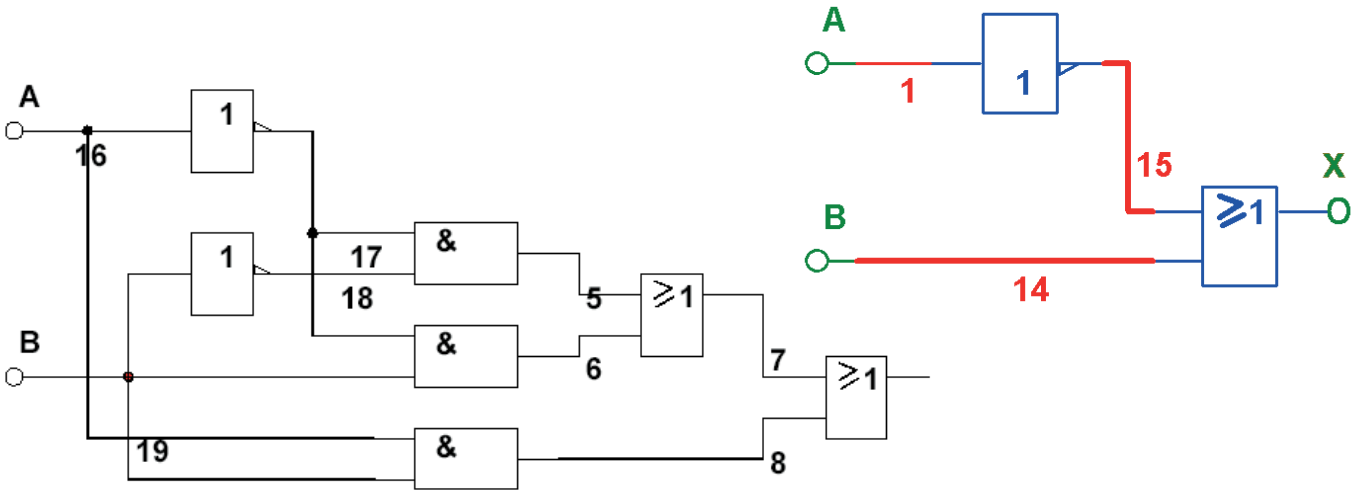
**Synthese von Digitalschaltungen:**

Bei der Synthese eines Schaltnetzes geht man von der bekannten logischen Funktion aus, d.h. man stellt die Wahrheitstabelle auf. Beide Normalformen lassen sich dann sofort realisieren.

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Disjunktive Normalform: X = $\overline{A}$ $\overline{B}$ + $\overline{A}$ B + A B

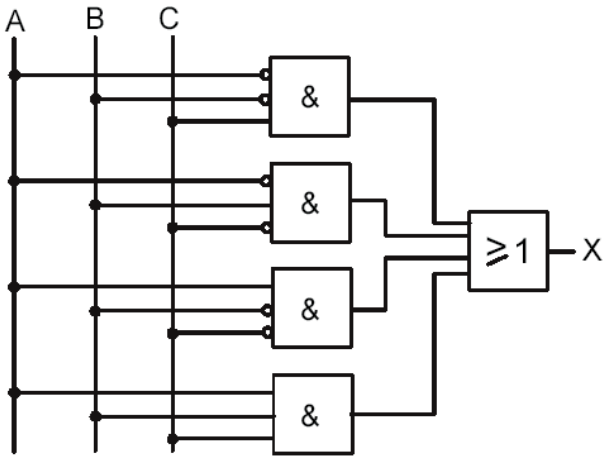
Konjunktive Normalform: X = $\overline{A}$ + B



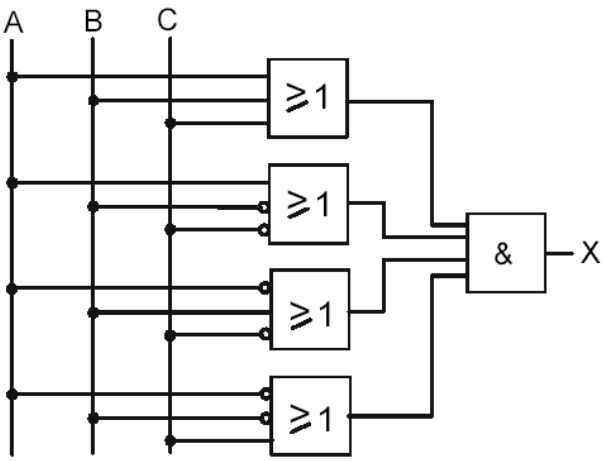
Beispiel für drei Eingänge:

| A | B | C | X | Minterm | Maxterm |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | $A + B + C$ |
| 0 | 0 | 1 | 1 | $\overline{A}\ \overline{B}\ C$ | |
| 0 | 1 | 0 | 1 | $\overline{A}\ B\ \overline{C}$ | |
| 0 | 1 | 1 | 0 | | $A + \overline{B} + \overline{C}$ |
| 1 | 0 | 0 | 1 | $A\ \overline{B}\ \overline{C}$ | |
| 1 | 0 | 1 | 0 | | $\overline{A} + B + \overline{C}$ |
| 1 | 1 | 0 | 0 | | $\overline{A} + \overline{B} + C$ |
| 1 | 1 | 1 | 1 | $A\ B\ C$ | |

Disjunktive Normalform:




Konjunktive Normalform:

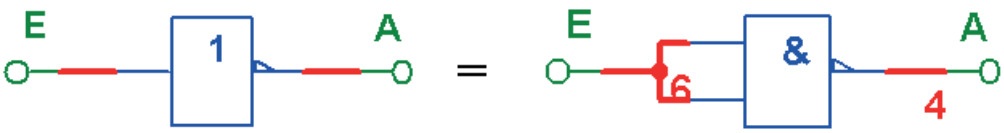



**Schaltungssynthese mit NAND-Bausteinen:**

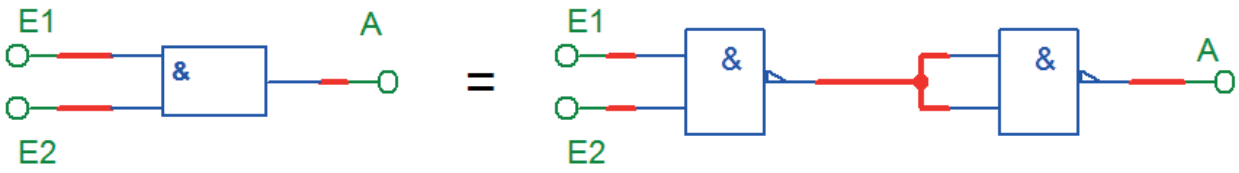
Stehen für die Schaltungssynthese nur NAND-Glieder zur Verfügung, dann ist die Aufgabe mit den NAND-Gattern die logischen Verknüpfungen AND, OR und NOT zu realisieren:

**NOT-Verknüpfung:**

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



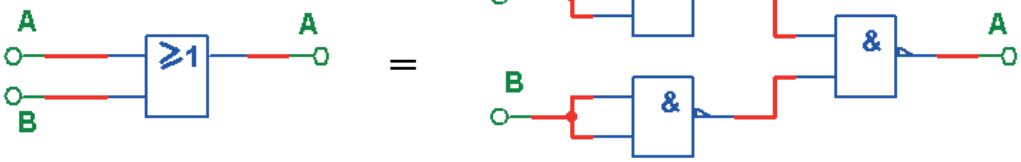

**AND-Verknüpfung:**



**OR-Verknüpfung:**

X = A + B = $\overline{\overline{A} \cdot \overline{B}}$

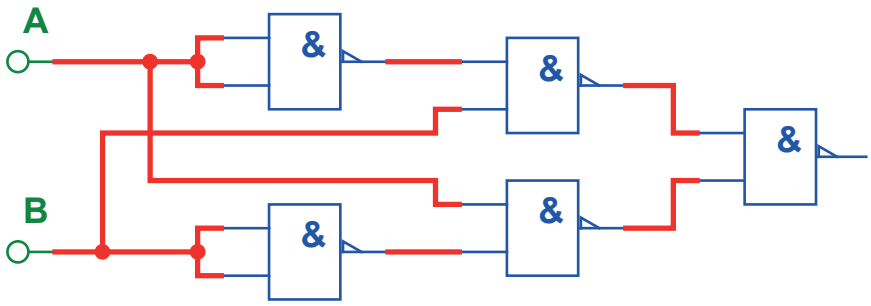
| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



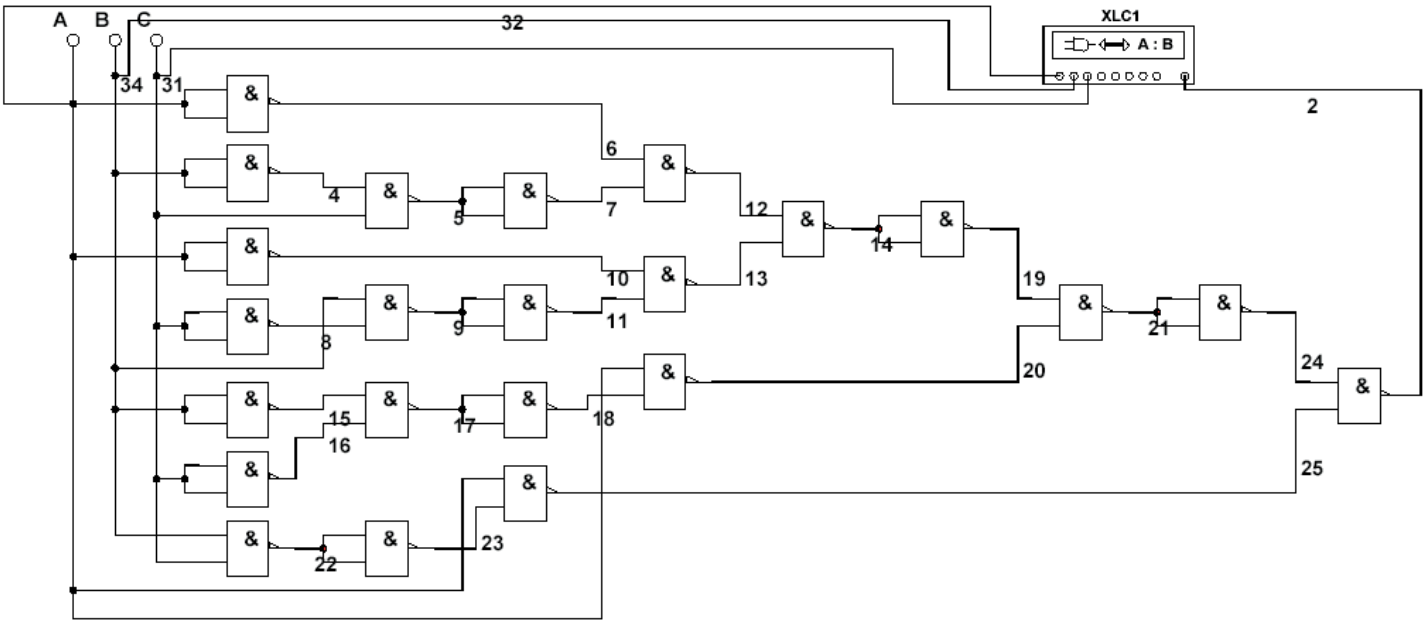

Somit kann man alle Schaltungen mit NAND-Gatter ausdrücken:

**XOR-Verknüpfung:** X = $\overline{A}$ B + A $\overline{B}$

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |






Die zugehörige Logikfunktion lautet:

$$X = \overline{A}\ \overline{B}\ C + \overline{A}\ B\ \overline{C} + A\ \overline{B}\ \overline{C} + A\ B\ C = \overline{\overline{A}\overline{B}C} \cdot \overline{\overline{A}B\overline{C}} \cdot \overline{A\overline{B}\overline{C}} \cdot \overline{ABC}$$

**Optimierung (Vereinfachung) von Schaltnetzen**

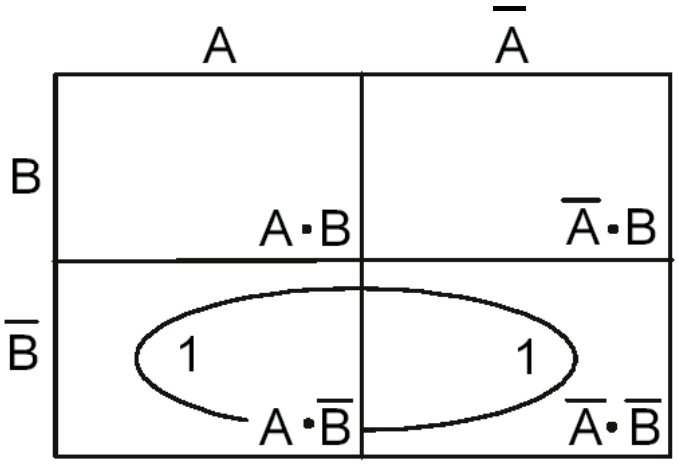
Bei der Optimierung von Schaltnetzen geht es darum, eine logische Funktion zu finden, in der möglichst wenige Verknüpfungen stehen. In der Praxis: Das gesuchte Schaltnetz besteht aus der geringsten Anzahl von Verknüpfungsgliedern (elektrischen Schaltungen).

- Rechenregeln der Schaltalgebra (rechnerisches Verfahren)
- Optimierung mit Hilfe des Karnaugh-Diagramms

Das Karnaugh-Diagramm ist lediglich eine andere Darstellung der Wahrheitstafel.

Bsp: Das Karnaugh-Diagramm der Funktion $X = A\,\overline{B} + \overline{A}\,\overline{B}$

- besteht aus 4 Feldern (mögliche Min-Terme)
- Min-Terme, die zu X = 1 führen werden gekennzeichnet
- Funktion hängt nicht von A ab, da sich zwei benachbarte Min-Terme sich im Wert von A unterscheiden






Die Rechnung bestätigt das:

$$X = A\,\overline{B} + \overline{A}\,\overline{B} = (A + \overline{A})\,\overline{B} = \overline{B} \quad \text{(Absorptionsgesetz)}$$

Weiteres Beispiel:

$$X = \overline{A}\;\overline{B}\;\overline{C} + \overline{A}\;B\;\overline{C} + A\;B\;\overline{C} + \overline{A}\;\overline{B}\;C + \overline{A}\;B\;C$$

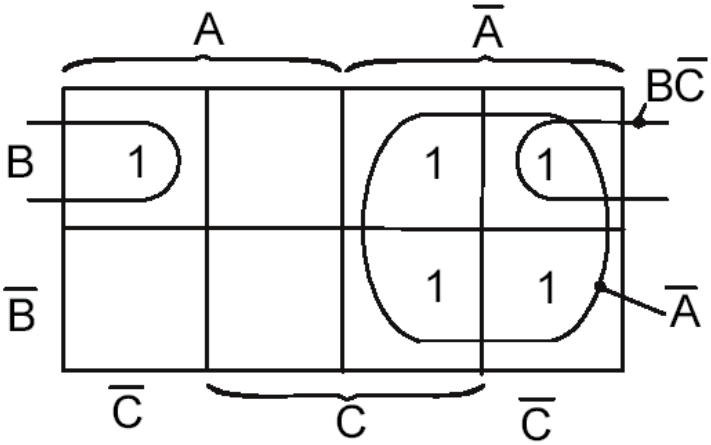
Vierer-Block von Min-Termen, die *B* und *C* enthalten. Hier ist $X = \overline{A}$

Zweier-Block von Min-Termen, die Variable *A* enthalten (Blöcke werden über die Ränder fortgesetzt)

Hier ist $X = B\,\overline{C}$

Insgesamt ergibt sich für die Funktion:

$X = \overline{A} + B\,\overline{C}$



Regeln zu den Karnaugh-Diagrammen:

1. Es dürfen nur Felder zusammengefasst werden, die direkt aneinander stoßen.
2. Es können nur 2, 4, 8, allgemein $2^n$ Felder zusammengefasst werden.
3. Der Inhalt dieser Gruppe ergibt sich aus den Koordinaten des Karnaugh-Diagramms. Alle Koordinaten, die negiert und nicht negiert auftreten können entfallen.
4. Die Ausgangsvariable wird durch die ODER-Normalform aller Gruppen dargestellt.
5. Die größte Vereinfachung erhält man, indem man möglichst große Gruppen bildet.
6. Die erweiterte Nachbarschaftsbeziehung erlaubt das Zusammenfassen von Gruppen über den Rand hinweg.
7. Eine Bildung von Gruppen über Ecken ist nicht zulässig.
8. Werden nicht alle Vollkonjunktionen in einem Karnaugh-Diagramm benötigt, so können diese, um größtmögliche Zusammenfassungen zu erreichen mit einer „1“ oder eine „0“ besetzt werden.



Beispiel zur Karnaugh-Veitch-Minimierung:

Verknüpfungsgleichungen für einen Kodewandler, der einen BCD-Kode in einen Gray-Kode umwandelt:

BCD-CODE (Eingang)

| A | B | C | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

Gray-Kode (Ausgang)

| O | P | Q | R |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |

$O = (A\,\overline{B}\,\overline{C}\,\overline{D}) + (A\,\overline{B}\,\overline{C}\,D)$

$P = (\overline{A}\,B\,\overline{C}\,\overline{D}) + (\overline{A}\,B\,\overline{C}\,D) + (\overline{A}\,B\,C\,\overline{D}) + (\overline{A}\,B\,C\,D) + (A\,\overline{B}\,\overline{C}\,\overline{D}) + (A\,\overline{B}\,\overline{C}\,D)$

$Q = (\overline{A}\,\overline{B}\,C\,\overline{D}) + (\overline{A}\,\overline{B}\,C\,D) + (\overline{A}\,B\,\overline{C}\,\overline{D}) + (\overline{A}\,B\,\overline{C}\,D)$

$R = (\overline{A}\,\overline{B}\,\overline{C}\,D) + (\overline{A}\,\overline{B}\,C\,\overline{D}) + (\overline{A}\,B\,\overline{C}\,D) + (\overline{A}\,B\,C\,\overline{D}) + (A\,\overline{B}\,\overline{C}\,D)$

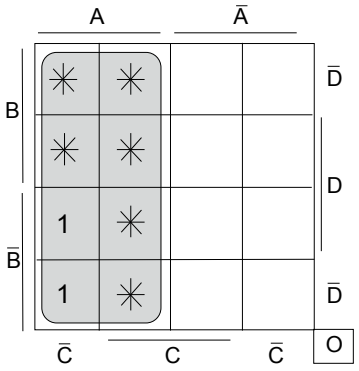
Zugehörige Diagramme ergeben:

O = A

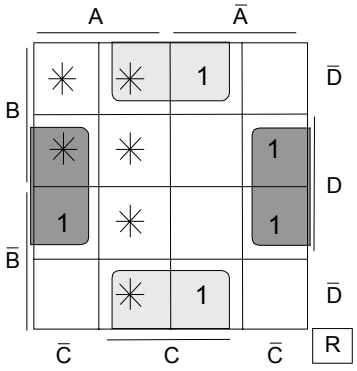
P = A + B

Q = (C* $\overline{B}$ )+( $\overline{C}$ *B)

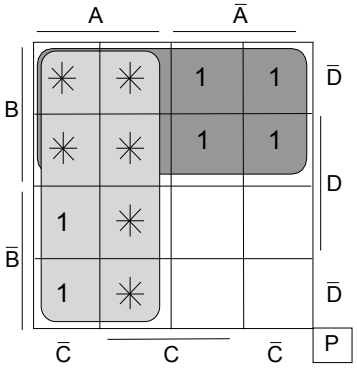
R = (C* $\overline{D}$ )+( $\overline{C}$ *D)



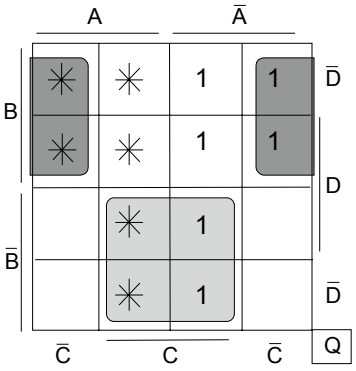

Zusammenfassung : O=A




Zusammenfassung : R=(C*$\overline{D}$) + ($\overline{C}$*D)
R=C ⊕ D




Zusammenfassung : P=A + B




Zusammenfassung : Q=(C*$\overline{B}$) + ($\overline{C}$*B)
Q=C ⊕ B