# 6 Einführung in die Digitaltechnik

In diesem Kapitel werden die Funktionen von grundlegenden Bauelementen der Digitaltechnik erklärt und wichtige Grundschaltungen der Digitaltechnik analysiert.

Digitale Signale benutzen nur eine begrenzte (diskrete) Anzahl von Zuständen. Sofern Störeinflüsse daher unter einer bestimmten Schwelle liegen, können digitale Signale fehlerfrei und ohne Qualitätsverlust übertragen und gespeichert werden. Weiters ist es möglich, digitale Aufzeichnungen beliebig oft zu kopieren und vervielfältigen zu können. Die für uns bedeutsamste Form ist die binäre Digitaltechnik, die nur zwei diskrete Zustände umfasst (log. Null, 0, Low, L; log. Eins, 1, High, H).

Weitere Vorteile der digitalen Signalverarbeitung gegenüber der Analogtechnik sind relativ geringe Bauteilkosten, und eine sehr hohe Integrationsdichte. Viele analoge Funktionen können im Digitalen durch Software realisiert werden. Dadurch werden Adaptierungen einfach durchführbar, ohne dass die Hardware an sich geändert werden muss. Eines der dafür in Europa unterrichteten und verwendeten Systeme ist die „Very High Speed Integrated Circuit Hardware Description Language (VHDL)“.

Einfache digitale Schaltungen, die die Grundlage aller digitalen Komponenten bilden, bestehen im Wesentlichen aus Gattern. Diese werden wir nun im Detail näher analysieren.

## 6.1 NOT

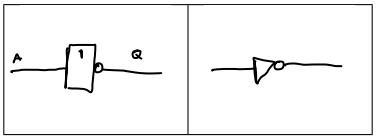
Inverter (NOT, Negation, Nicht-Funktion) invertieren das Eingangssignal. Hier wird X für das Eingangssignal verwendet, Y für den Ausgang:

$$Y = \overline{X} = \neg X$$

Daraus ergibt sich folgende Wahrheitstabelle:

| $X$ | 0 | 1 |
|---|---|---|
| $Y$ | 1 | 0 |

Der Ausgang eines Inverters wird 0, wenn der Eingang 1 ist; er wird 1, wenn der Eingang 0 ist. Für den Inverter werden folgende Symbole verwendet (links EN 60617-12 Standard, rechts alternatives Symbol nicht normgemäß, aber in manchen Programmen eingesetzt):




## 6.2 AND

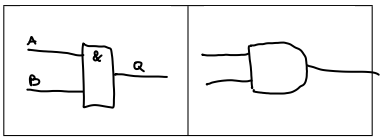
Der Ausgang der Und-Funktion (Konjunktion, AND) eines Gatters wird dann 1, wenn alle Eingänge 1 sind. Der Ausgang wird 0, wenn mindestens ein Eingang 0 ist. Dieses Gatter ist auf beliebig viele Eingänge erweiterbar.

$$Y = X_0 \wedge X_1 = X_0 \cdot X_1$$

Daraus ergibt sich folgende Wahrheitstabelle:

| $X_0$ | 0 | 0 | 1 | 1 |
|---|---|---|---|---|
| $X_1$ | 0 | 1 | 0 | 1 |
| $Y$ | 0 | 0 | 0 | 1 |

Die Symbole sind üblicherweise:




Ein AND-Gatter kann zur Sperre bzw. zur Freigabe eines Signals benützt werden, in dem ein Eingang als Sperreingang eingesetzt wird. Im gesperrten Zustand liegt der Ausgang des Gatters auf 0.

## 6.3 OR

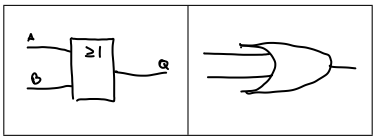
Der Ausgang der Oder-Funktion (Disjunktion, OR) eines Gatters wird dann 0, wenn alle Eingänge 0 sind. Der Ausgang wird 1, wenn mindestens ein Eingang 1 ist. Auch dieses Gatter ist auf beliebig viele Eingänge erweiterbar.

$$Y = X_0 \vee X_1 = X_0 + X_1$$

Daraus ergibt sich folgende Wahrheitstabelle:

| | | | | |
|---|---|---|---|---|
| $X_0$ | 0 | 0 | 1 | 1 |
| $X_1$ | 0 | 1 | 0 | 1 |
| $Y$ | 0 | 1 | 1 | 1 |

Üblich sind unten stehende Symbole:




Ein OR-Gatter kann zur Sperre bzw. zur Freigabe eines Signals benützt werden. Im gesperrten Zustand liegt der Ausgang des Gatters auf 1.

## 6.4 XOR

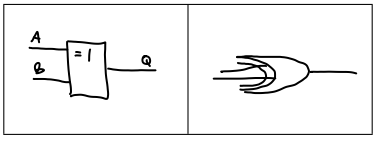
Der Ausgang der Exklusiv-Oder-Funktion (Antivalenz, EXOR, XOR) ist prinzipiell nur für zwei Eingänge definiert. Der Ausgang wird dann 1, wenn die Eingänge ungleich sind; er wird dann 0, wenn die Eingänge gleich sind.

$$Y = X_0 \oplus X_1 = X_0 \veebar X_1$$

Daraus ergibt sich folgende Wahrheitstabelle:

| | | | | |
|---|---|---|---|---|
| $X_0$ | 0 | 0 | 1 | 1 |
| $X_1$ | 0 | 1 | 0 | 1 |
| $Y$ | 0 | 1 | 1 | 0 |

Für das XOR-Gatter werden folgende Symbole eingesetzt:




Ein XOR-Gatter kann zur gesteuerten Negation eines Signales verwendet werden.

## 6.5 NAND

Eine NAND Funktion (Nicht-UND oder Sheffer-Funktion) entspricht einer Und-Verknüpfung mit anschließender Negation. Diese Funktion kann daher für beliebig viele Eingänge definiert werden.

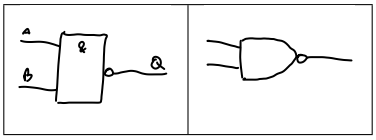
Der Ausgang des NAND Gatters wird 0, wenn alle Eingänge 1 sind. Er wird 1, wenn mindestens ein Eingang 0 ist.

$$Y = \overline{X_0 \wedge X_1} = X_0 \overline{\wedge} X_1 = \overline{X_0 X_1} = A|B$$

Daraus ergibt sich folgende Wahrheitstabelle:

| | | | | |
|---|---|---|---|---|
| $X_0$ | 0 | 0 | 1 | 1 |
| $X_1$ | 0 | 1 | 0 | 1 |
| $Y$ | 1 | 1 | 1 | 0 |

Für das NAND-Gatter werden folgende Symbole eingesetzt:



Ein NAND-Gatter kann zur Sperre bzw. Freigabe eines Signals mit anschließender Negation verwendet werden. Im gesperrten Zustand liegt dabei der Ausgang auf 1.

## 6.6 NOR

Die NOR Verknüpfung (Nicht-ODER oder Pierce-Funktion) entspricht einer Oder-Verknüpfung mit anschließender Inversion. Diese Funktion kann daher wieder für eine größere Anzahl von Eingängen gebaut werden.

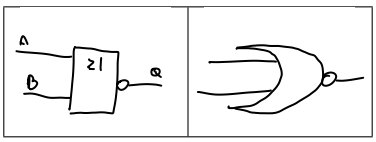
Der Ausgang des NOR Gatters wird 1, wenn alle Eingänge 0 sind. Er wird 0, wenn mindestens ein Eingang 1 ist.

$$Y = \overline{X_0 \vee X_1} = X_0 \overline{\vee} X_1 = \overline{X_0 + X_1}$$

Daraus ergibt sich folgende Wahrheitstabelle:

| | | | | |
|---|---|---|---|---|
| $X_0$ | 0 | 0 | 1 | 1 |
| $X_1$ | 0 | 1 | 0 | 1 |
| $Y$ | 1 | 0 | 0 | 0 |

Für das NOR-Gatter werden folgende Symbole eingesetzt:



Ein NOR-Gatter kann auch wieder zur Sperre bzw. Freigabe eines Signals mit anschließender Negation verwendet werden. Im gesperrten Zustand liegt der Ausgang dabei auf 0.

## 6.7 XNOR

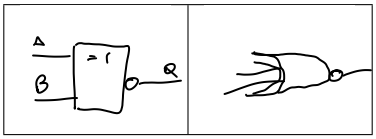
Der Ausgang der Exklusiv-Nicht-Oder-Funktion (Äquivalenz, EXNOR, XNOR) ist prinzipiell nur für zwei Eingänge definiert. Der Ausgang wird dann 1, wenn die Eingänge gleich sind; er wird dann 0, wenn die Eingänge ungleich sind.

$$Y = \overline{X_0 \oplus X_1} = \overline{X_0 \veebar X_1} = X_0 \barwedge X_1$$

Daraus ergibt sich folgende Wahrheitstabelle:

| | | | | |
|---|---|---|---|---|
| $X_0$ | 0 | 0 | 1 | 1 |
| $X_1$ | 0 | 1 | 0 | 1 |
| $Y$ | 1 | 0 | 0 | 1 |

Für das XNOR-Gatter werden folgende Symbole eingesetzt:



Ein XNOR-Gatter kann zur gesteuerten Negation eines Signales verwendet werden.

## 6.8 Verknüpfungen mit mehreren Eingängen in der Praxis

Üblicherweise sind NAND, NOR, AND und OR als Bausteine mit mehreren Eingängen verfügbar. Oft werden NAND und NOR Gatter verwendet, da durch diese die anderen Gatterfunktionen erzeugt werden können (siehe Konjunktive- und Disjunktive-Normalform im GINF-Teil von Systemtechnik). Die Inverterfunktion kann durch ein 2-fach NAND oder NOR-Gatter realisiert werden, indem beide Eingänge zusammengeschlossen werden.

Üblich sind NAND und NOR Gatter mit 3, 4, 5, 8 und 13 Eingängen.

## 6.9 Logikfamilien

Als Logikfamilie wird in der Digitaltechnik eine Serie von Gattern oder Bauteilen zusammengefasst, die in ähnlich gefertigter Herstellungstechnologie erzeugt worden sind. Daraus ergeben sich also vergleichbare elektrische, zeitliche, mechanische und thermische Eigenschaften. Es ist daher immer möglich, Bausteine einer Logikfamilie miteinander zu kombinieren. Beim Verwenden unterschiedlicher Familien in einer Schaltung wird es komplizierter.

### 6.9.1 74LS-Serie

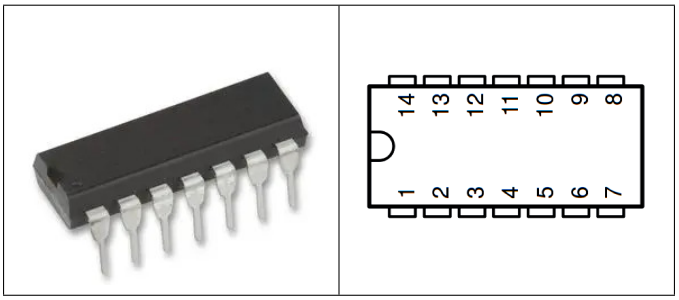
Die 74er-Serie wurde in der sogenannten Transistor-Transistor-Logik (TTL) gefertigt. Hier werden Dioden, Widerstände und Transistoren mit mehreren Emitteranschlüssen auf einem Chip integriert. Die Familie des Herstellers Texas Instruments (TI) setzte sich als Industriestandard durch. Die Generation 74xy ist veraltet und wird kaum mehr hergestellt.

Eine der wichtigsten Logikfamilien ist die 74LS-Reihe. Die Gatter werden in Low Power Schottky Technologie gefertigt, daher die Bezeichnung 74LS... . Buchstabenkombinationen davor kennzeichnen in der Regel die Herstellerfirma. Die anschließende Zahlen-/Buchstabenkombination beschreibt die Logikfunktion der Schaltung.

Es wurden veschiedene Logikfamilien entwickelt, die sich neben der Herstellungsmethode vor allem in den Spannungspegeln, dem Timingverhalten und dem Energieverbrauch unterscheiden.

Üblicherweise werden diese Bauteile im Kunststoffgehäuse für die Durchsteckmontage oder in SMD-Versionen gefertigt. Erste haben einen Pinabstand von 2,54 mm (aus dem Zollraster abgeleitet) und passen in Steckbretter.

Surface Mounted Devices werden per Oberflächenlötmontage auf Leiterplatten befestigt. Die linke Abbildung zeigt ein 74LS... Gatter im DIL Gehäuse. Rechts davon sehen Sie das typische Pining. Pin Nummer 1 liegt links unten, gekennzeichnet durch eine Einkerbung oder einen aufgedruckten Punkt nahe des Pins. Die weitere Nummerierung läuft entgegen dem Uhrzeigersinn.




### 6.9.2 4000er-Serie

Die 4000er Serie ist in CMOS-Technologie gefertigt. Hier werden anstelle bipolarer Transistoren MOSFETs (Metal Oxyde Semiconductor Field Effect Transistors) eingebaut. CMOS bedeutet Complimentary MOS, hier werden also P- und N-Kanal MOSFETS gemeinsam eingebaut.

Diese Logikfamilie zeichnet sich durch einen geringen Stromverbrauch und einen großen erlaubten Spannungsversorgungsbereich aus (z.B. 3V - 15V). Die Schaltvorgänge sind aber eher langsam und es können keine großen Ausgangsströme erreicht werden. Die Urserie war relativ empfindlich gegenüber elektrostatischen Entladungen. Deswegen wurde die zweite Generation 4000B mit Eingangsschutzschaltungen ausgeführt.

### 6.9.3 74HC-Serie

HC steht hier für High Speed CMOS. Diese Reihe verwendet die Gatterbezeichnung der 74er Serie und ist mit diesen pinkompatibel. Sie kombiniert die Vorteile des geringen Leistungsverbrauches, eines geringen erforderlichen Eingangsstromes und der Schnelligkeit. Typischerweise ist der Versorgungsspannungsbereich 2 V bis 6 V.

## 6.10 Komplexere Verknüpfungen

### 6.10.1 Halbaddierer (Half Adder)

Der Halbaddierer hat zwei Eingänge (A, B) und zwei Ausgänge: Einen für die Summe (S), einen für den Übertrag (C wie carry).

Wahrheitstabelle:

| | | | | |
|---|---|---|---|---|
| $A$ | 0 | 1 | 0 | 1 |
| $B$ | 0 | 0 | 1 | 1 |
| $S$ | 0 | 1 | 1 | 0 |
| $C$ | 0 | 0 | 0 | 1 |

Der Halbaddierer hat keinen Eingang für den Übertrag einer Stelle mit niedrigerer Wertigkeit und ist daher nur für das LSB brauchbar. An allen anderen Stellen benötigt man den Volladdierer. Übungsaufgabe: Skizzieren Sie aufgrund der Wahrheitstabelle das Innenleben der Halbaddiererschaltung.

### 6.10.2 Volladdierer (Full Adder)

Ein Volladdierer besteht aus drei Eingängen (A, B, CI für carry in) und zwei Ausgängen (S für Summe, C für carry out). Daraus ergibt sich folgende Wahrheitstabelle:

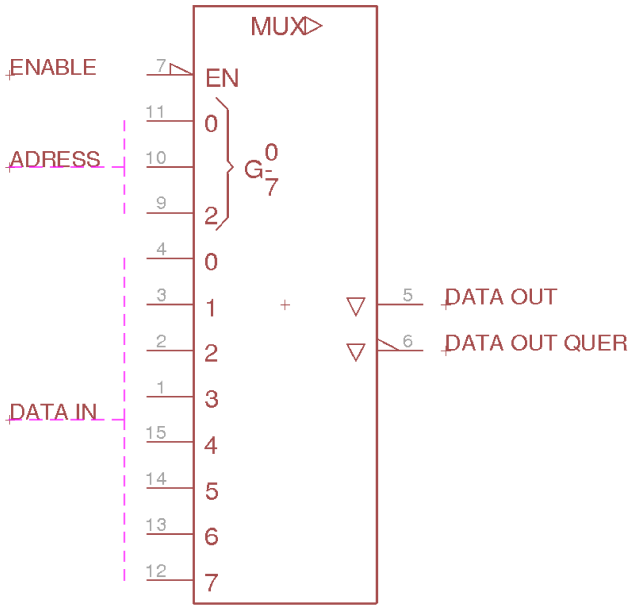
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $B$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $C_{in}$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $S$ | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| $C_{out}$ | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

Folgendes Symbol kann für den Volladdierer eingesetzt werden, sein Innenleben könnte man sich aus folgenden Komponeten zusammengesetzt denken:

In dieser Struktur steht das Summenergebnis S nach zwei Gatterlaufzeiten zur Verfügung. Allerdings benötigt der Übertrag Cout drei Gatterlaufzeiten, da das Zwischenergebnis $A \oplus B = A \veebar B$ an den Eingang des zweiten Halbaddierers zurückgeführt wird. Das nicht gleichzeitige Anliegen der beiden Ergebnisausgänge kann daher relativ leicht zu Instabilitäten und ungewollten Zuständen führen. Abhilfe schafft eine andere Struktur ohne innere Signalrückführung:

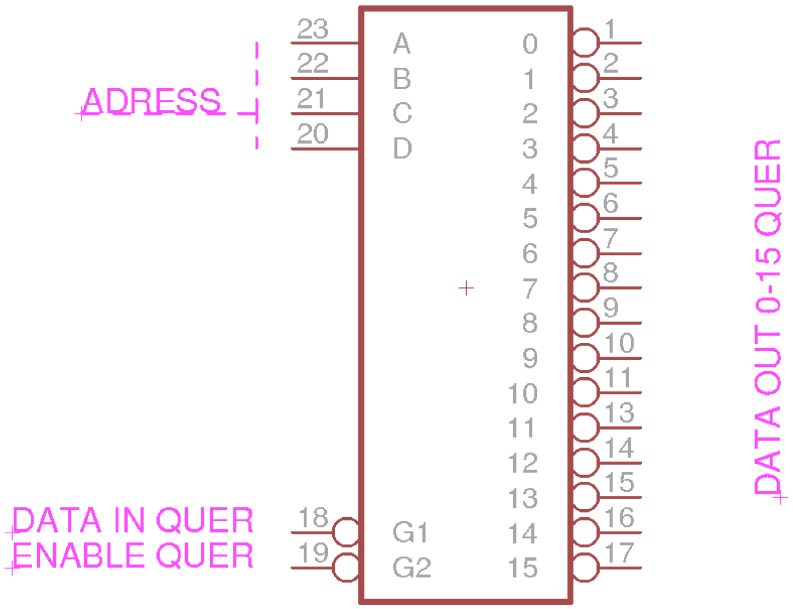
### 6.10.3 Multiplexer

Einer der $2^n$ Eingänge DATA IN wird an den Ausgang DATA OUT durchgeschaltet, abhängig von der Adresse ADRESS und dem Enable oder Chipselect PIN (Freigabeeingang). Hier ein Beispiel:




### 6.10.4 Demultiplexer

Je nach Adressierung wird der Dateneingang an einen der Ausgänge durchgelegt (abhängig vom Enable oder Chipselect Eingang). In diesem Beispiel kann der Eingang DATA IN QUER an einen der 16 Ausgänge DATA OUT QUER mittels ENABLE QUER und der Adressierungspins A - D geschalten werden:

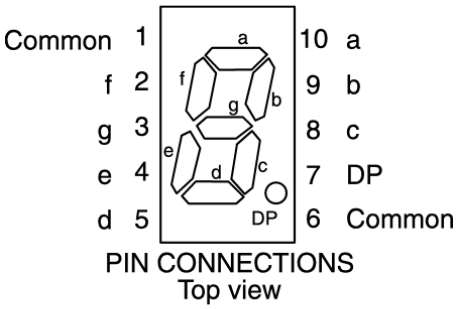



### 6.10.5 Codierer (Coder, Encoder)

Typisches Beispiel ist die Umsetzung eines “1 aus x - Codes” in entsprechende Dualzahlen. Meist wird hier ein Prioritätsencoder eingesetzt, das heißt der höchst priorisierte aktive Eingang bestimmt den Wert des Ausgangssignales.

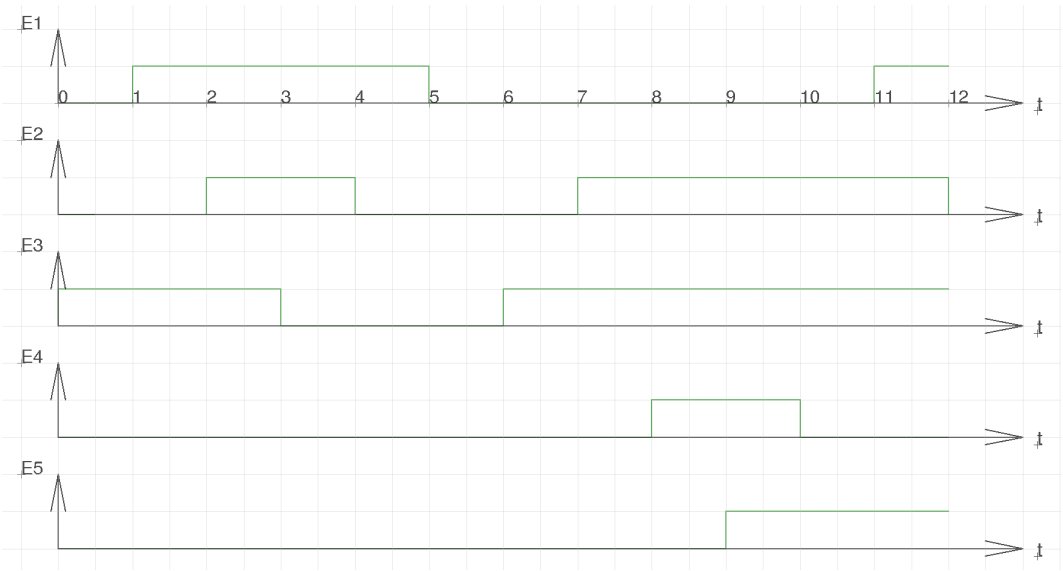
### 6.10.6 Decodierer (Decoder)

Anwendungsbeispiele sind oft 4 Bit Zahlen auf “1 aus 16 Code” oder 4 Bit auf eine 7-Segmentanzeige (BCD, binary coded decimal). Mit einer Siebensegmentanzeige können die Ziffern 0 - 9, ein Dezimalpunkt und teilweise Buchstaben angezeigt werden:



## 6.11 Übungsbeispiele

Verwenden Sie für die unten angeführten Beispiele folgende Eingangssignale über der Zeit:




### 6.11.1 Beispiel

Sie haben folgende Funktionsgleichung gegeben:

$$Y = (E_1 \oplus E_2) \vee (\overline{(E_4 \vee E_5)} \wedge E_3)$$

- Zeichnen Sie die Schaltung.
- Skizzieren Sie den Zeitverlauf des Ausgangssignales Y. Verwenden Sie dazu Hilfsvariablen an den Gatterausgängen.
- Entwickeln Sie eine vollständige Wahrheitstabelle.
- Simulieren Sie die Wahrheitstabelle in einem Simulationsprogramm mittels idealisierter Gatter.
- Simulieren Sie die Wahrheitstabelle in einem Simulationsprogramm mit Bausteinen aus der 74er Familie.

### 6.11.2 Beispiel

Sie haben folgende Funktionsgleichung gegeben:

$$Y = \overline{(E_2 \oplus E_3)} \wedge \overline{(\overline{(E_1 \vee E_5)} \wedge (E_4 \vee E_1))}$$

- Zeichnen Sie die Schaltung.
- Skizzieren Sie den Zeitverlauf des Ausgangssignales Y. Verwenden Sie dazu Hilfsvariablen an den Gatterausgängen.
- Entwickeln Sie eine vollständige Wahrheitstabelle.
- Simulieren Sie die Wahrheitstabelle in einem Simulationsprogramm mittels idealisierter Gatter.
- Simulieren Sie die Wahrheitstabelle in einem Simulationsprogramm mit realen Bausteinen aus der CMOS Familie.