11 Digitale Signalumsetzung

11.1 Logische Grundfunktionen

11.1.1 Boole'sche Algebra

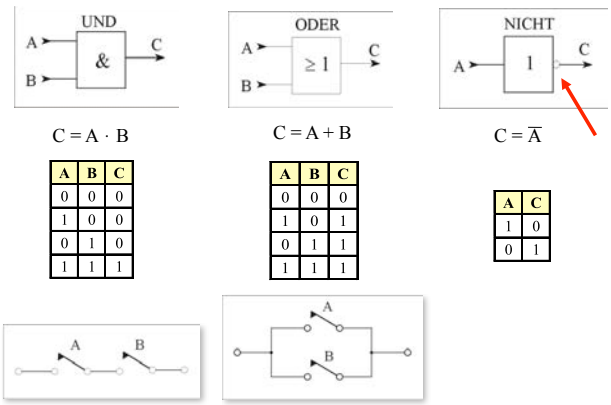


Abbildung 11.1: Die Grundoperationen der Boole'schen Algebra.

George Boole (1815–1864) machte 1847 Vorschläge, wie logische Operationen formalisiert werden könnten, auf der Basis von Operationen wie AND, OR und NOT, welche auf binären Zahlen operieren.

Die wichtigsten Grundrechenoperationen sind UND, ODER und NICHT; die ersten beiden benötigen zwei bits am Eingang, NICHT ein einzelnes. Alle drei haben ein einzelnes Bit als Ausgabewert. Dies bedeutet, dass die beiden Operationen UND und ODER Information verlieren.

Abb. 11.1 stellt diese Operationen mit Hilfe von Tabellen dar, welche für die verschiedenen möglichen Eingangswerte den Ausgangswert angeben.

In der untersten Zeile ist dargestellt, wie diese Operationen mit einfachen Schaltkreisen realisiert werden können: ein geschlossener Schalter entspricht einem logischen 1, ein offener Schalter einer 0.

Dual (binär)	Dezimal	Oktal	Hexadezimal
000 000 = 0000	0	0	0
000 001 = 0001	1	1	1
000 010 = 0010	2	2	2
000 011 = 0011	3	3	3
000 100 = 0100	4	4	4
000 101 = 0101	5	5	5
000 110 = 0110	6	6	6
000 111 = 0111	7	7	7
001 000 = 1000	8	10	8
001 001 = 1001	9	11	9
001 010 = 1010	10	12	A
001 011 = 1011	11	13	B
001 100 = 1100	12	14	C
001 101 = 1101	13	15	D
001 110 = 1110	14	16	E
001 111 = 1111	15	17	F

Abbildung 11.2: Umrechnung zwischen Zahlsystemen.

11.1.2 Zahlsysteme

Neben dezimalen und binären Zahlen werden auch oktale und hexadezimale Zahlen verwendet. In jedem Zahlensystem mit Basis b wird eine Zahl dargestellt als eine Reihe

$$z = \sum_{i=0}^{n} a_i b^i,$$

wobei i den Index der Stelle darstellt und a_i den Wert der Ziffer an dieser Stelle.

Abb. 11.2 vergleicht die Darstellung der dezimalen Zahlen 0 bis 16 in Zahlensystemen mit der Basis 2, 8, 10 und 16.

11.1.3 Addition von Binärzahlen

Komplexere Operationen können auf die Boole'schen Grundoperationen zurückgeführt werden.

Als Beispiel betrachten wir die Addition von zwei Binärzahlen. Das Resultat besteht aus zwei bits, von

x1	x2	x1+x2	Übertrag
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

C = carry bit
= Übertrag

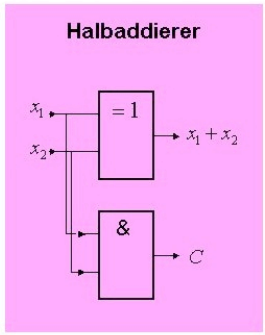


Abbildung 11.3: Wahrheitstabelle und Schaltschema eines Halbaddierers.

denen eines den Wert $x_1 + x_2 \mod 2$ annimmt, und ein Übertragsbit. Diese Schaltung wird als Halbaddierer bezeichnet; das "Halb-" bezieht sich darauf, dass er nicht berücksichtigt, dass die vorangehende Stufe einen Übertrag bringen kann.

a_i	b_i	c_i	Σ_i	c_{i+1}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	0	1	1

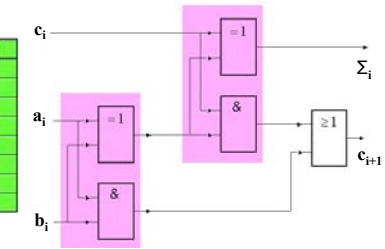


Abbildung 11.4: Wahrheitstabelle und Schaltschema eines 1-bit Volladdierers.

Einen Volladdierer erhält man durch einen weiteren Halbaddierer, der den Übertrag aus der vorhergehenden Stelle dazuaddiert.

Um mehrstellige Binärzahlen zu addieren, müssen mehrere Volladdierer kaskadiert werden. Das Resultat ist in Abb. 11.5 dargestellt: Jeder Volladdierer erhält als Eingangswert die entsprehenden bits der Summanden sowie den Übertrag aus der niedrigeren Stufe und berechnet den Summenwert und den Wert des Übertragsbits.

11.1.4 Zweierkomplement und Subtraktion

Negative Zahlen werden im binären Zahlensystem dadurch markiert, dass das erste (most significant)

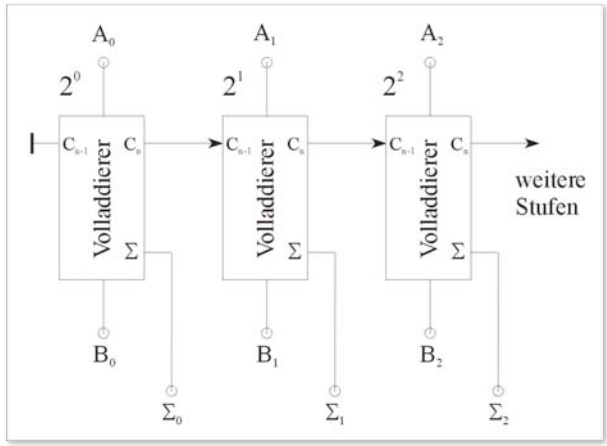


Abbildung 11.5: Addition von Binärzahlen mit Hilfe von kaskadierten Volladdierern.

	Bitdarstellung								
2er Komplement	b3	b2	b1	b0					
0	0	0	0	0	-8	1	0	0	0
1	0	0	0	1	-7	1	0	0	1
2	0	0	1	0	-6	1	0	1	0
3	0	0	1	1	-5	1	0	1	1
4	0	1	0	0	-4	1	1	0	0
5	0	1	0	1	-3	1	1	0	1
6	0	1	1	0	-2	1	1	1	0
7	0	1	1	1	-1	1	1	1	1

Abbildung 11.6: Zahlen von -8..+7 in binärer Darstellung.

bit den Wert 1 annimmt. Die binäre Darstellung einer negativen Zahl x erhält man, indem man das binäre Komplement von $|x|$ bildet, d.h. jedes bit mit Wert 0 durch 1 und jede 1 durch 0 ersetzt, und zum Resultat 1 addiert. Als Beispiel: 5 = 0101; -5 = 1011.

Diese Darstellung erlaubt eine relativ einfache Implementation der Subtraktion: $A - B$ wird in 3 Schritten berechnet:

1. Man bildet das Zweierkomplement von B und addiert 1
2. Man addiert dies zu A
3. Man streicht die höchste Stelle.

Als Beispiel rechnen wir 6-4, wobei wir die Zahlen durch 4 bits codieren:

1. -4 = 1100

2. 0110 + 1100 = (1)0010
3. 6-4 = 0010 = 2.

11.2 DTL-, TTL-, ECL- und CMOS-Logik

11.2.1 Dioden-Logik

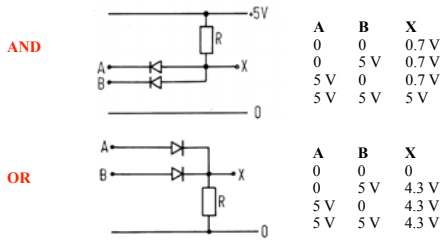


Abbildung 11.7: AND und OR als Dioden-Logik.

Die Dioden-Logik kann keine Negation implementieren. Die Flussspannung der Diode von ca. 0,7 V verringert den Abstand zwischen H und L. Bei hintereinander geschalteten Gattern sind diese bald nicht mehr unterscheidbar.

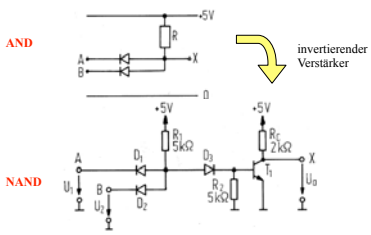


Abbildung 11.8: NAND in DTL-Logik.

Das Problem kann umgangen werden, indem man an den Ausgang einen invertierenden Transistor-Verstärker hängt. Hier kompensiert die Diode D_3 die Flussspannung von D_1 und D_2. Aufgrund der Inversion ist das resultierende Gatter ein NAND-Gatter. Der Nachteil dieses DTL-Gatters ist, dass der Übergang $L \rightarrow H$ relativ langsam ist.

11.2.2 TTL

Die Geschwindigkeit ist wesentlich höher wenn nur Transistoren verwendet werden. Abb. 11.9 zeigt

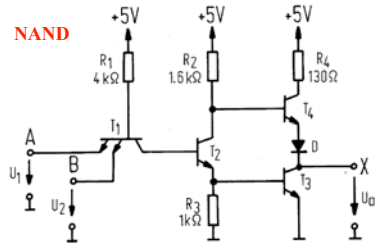


Abbildung 11.9: NAND in TTL-Logik.

wie mit TTL-Logik ein NAND-Gatter implementiert werden kann. Hier wird ein Multiemitter-Transistor verwendet. Man unterscheidet zwischen unterschiedlichen Familien von TTL-Gattern, wie z.B. 'advanced low-power Schottky TTL', 'advanced Schottky TTL' oder 'fast TTL'; diese unterscheiden sich vor allem in der Gatterlaufzeit (meist zwischen 1 und 5 ns), sowie im Stromverbrauch (1 mW bis 10 mW).

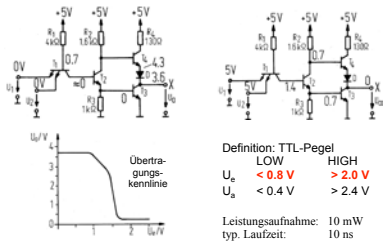


Abbildung 11.10: Pegel in TTL-Logik.

Sind beide Eingänge tief (<0.8 V), so sind die Basisspannunen von T_2 und T_3 niedrig, so dass T_3 sperrt, und die Ausgangsspannung liegt bei etwa 3.6 V. Sind die beiden Eingänge hoch, so werden T_2 und T_3 durchgeschaltet und die Ausgangsspannung liegt bei Null. Zwischen den beiden Werten macht die Ausgangsspannung einen Übergang. Man wählt deshalb die beiden logischen Werte so, dass der Zustand L einem Bereich von <0,8 V entspricht und H ist > 2.0 V. Am Ausgang sollte L < 0.4 V sein, H > 2.4 V.

11.2.3 ECL

Emittergekoppelte Logik ist noch schneller, da Sperrverzögerung vermieden wird. Hier fließt Im stationären Zustand ($U_e = 0$ V) durch beide Transistoren der gleiche Strom ($I_E/2$). Wird $U_e > 0$ V, so

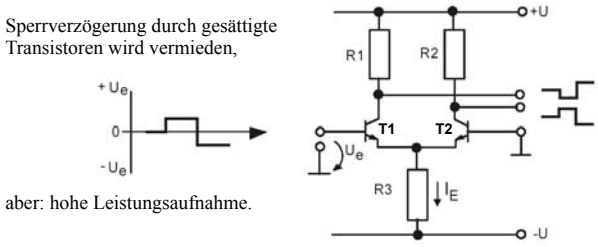


Abbildung 11.11: NAND in ECL-Logik.

wird T1 mehr und mehr leitend, während T2 mehr und mehr sperrt. Wird Ue < 0 V, so gilt das Umgekehrte. NOR-Gatter durch Zuschalten eines Transistors parallel zu T1 und einer invertierenden Verstärkerstufe am Ausgang realisierbar. : Schnell, da keine Verzögerung durch gesättigte Transistoren, aber hohe Leistungsaufnahme

11.2.4 CMOS

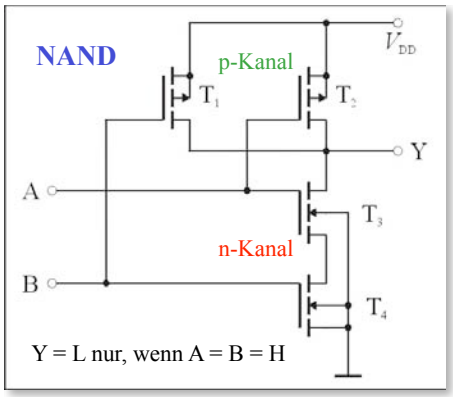


Abbildung 11.12: NAND in CMOS-Technik.

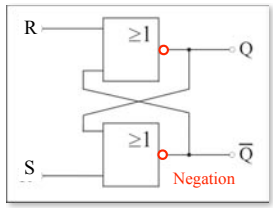
CMOS-Logik arbeitet mit MOSFETs; erlaubt hohe Integrationsdichte; Strom nur beim Umladen

11.3 Flip-Flops : Sequentielle Logik

Die Basis der Flip-Flop Schaltung hatten wir im Kapitel 9.2.2 vorgestellt. Im Folgenden soll sie als Basis vieler digitalen logischen Schaltungen nochmals diskutiert werden.

11.3.1 Aufbau und Funktion

Die Eingänge R und S (positive Logik) sind zum Setzen (S) bzw. Rücksetzen (R) des Flip-Flops.



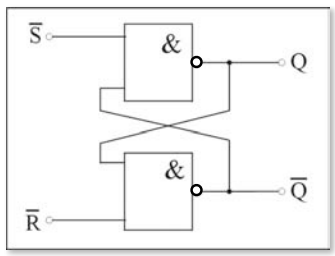
Flip-Flop ist 1-Bit-Speicher

R	S	Q	$\bar{Q}$
0	0	Q_{-1}	$\bar{Q}_{-1}$
1	0	0	1
0	1	1	0
1	1	---	---

Abbildung 11.13: RS Flip-Flop aus NOR Gattern.

Als logische Schaltung kann man ein Flip-Flop aus zwei NOR-Gattern zusammensatzen, deren Ausgänge jeweils auf den Eingang des anderen NOR Gatters zurückgekoppelt sind. Das NOR Gatter erzeugt am Ausgang eine 0, falls eines der Eingänge =1 ist und sonst eine 1. Ist der Ausgang des einen NOR Gatters hoch, ist somit der andere automatische tief. Die beiden entsprechenden Zustände sind stabil. Man kann zwischen den beiden Zuständen umschalten, indem man an den einen oder den anderen Eingang (R, S) ein hohes Signal anlegt: das entsprechende NOR Gatter wird dann am Ausgang zu 0 gesetzt, das andere entsprechend auf 1.

Der Zustand des RS Flip-Flops hängt von der Vorgeschichte ab. Schaltungen mit Flip-Flops werden deshalb als sequentielle Logik bezeichnet. Da gleichzeitige Pulse auf R und S zu einem unbestimmten Verhalten führen, sollte diese Situation vermieden werden.



$\bar{R}$	$\bar{S}$	Q	$\bar{Q}$
0	0	---	---
1	0	1	0
0	1	0	1
1	1	Q_{-1}	$\bar{Q}_{-1}$

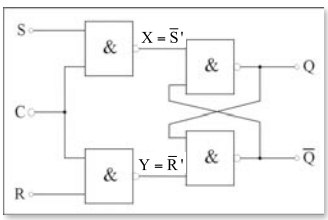
Abbildung 11.14: RS-Flip-Flop aus NAND Gattern

Anstelle von NOR Gattern kann man ein Flip-Flop auch aus NAND-Gattern aufbauen. In diesem Fall müssen beide Eingänge des NAND Gatters auf 1

sein, damit der Ausgang auf 0 gesetzt wird. Die Logik ist in diesem Fall invertiert: sind R und S auf 1 gesetzt, so bleibt der alte Zustand erhalten: ist z. B. gleichzeitig Q = 1, so ist $\bar{Q}$=0 und umgekehrt.

Zum Umschalten wird ein negativer Puls auf einen der beiden Eingänge gelegt. Diese invertierte Logik wird durch die Balken ($\bar{R}$, resp. $\bar{S}$) markoert. Ein solcher negativer Puls auf (z.B. $\bar{S}$ = 0), führt dazu dass der Ausgang des entsprechenden NAND-Gates auf Q=1 gesetzt wird. Damit sind beide Eingänge am unteren NAND Gatter =1 und dessen Ausgang wechselt auf 0.

11.3.2 Getaktete Flip-Flops



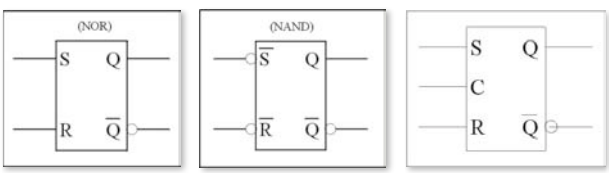
positive Logik

C	R	S	Q
1	0	0	Q_{-1}
1	1	0	0
1	0	1	1
1	1	1	---
0	belie	-big	Q_{-1}

Abbildung 11.15: Getacktetes RS Flip-Flop.

Manchmal möchte man, dass das Flip-Flop nur zu bestimmten Zeiten geschaltet werden kann. Dies erreicht man, indem vor die eigentlichen Eingänge je ein weiteres NAND Gatter geschaltet wird. Nur wenn am C-Eingang ein Taktimpuls liegt, können die R- und S-Pulse an die Eingänge X und Y des eigentlichen Flip-Flops gelangen. Dabei werden sie invertiert, d.h. X = S' und Y = S'.

Kurzsymbole der RS-Flip-Flops

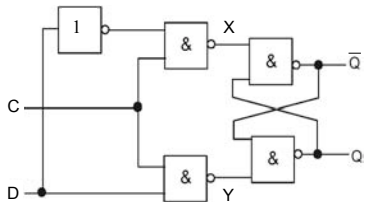


RS-Flip-Flop aus NOR-Gattern | RS-Flip-Flop aus NAND-Gattern | Getaktetes RS-Flip-Flop

Abbildung 11.16: Kurzsymbole für RS Flip-Flops.

Abb. 11.16 zeigt die Symbole für die wichtigsten Typen von RS Flip-Flops.

11.3.3 Data-Latch Flip-Flops



C	D	Q
0	0	Q_{-1}
0	1	Q_{-1}
1	0	0
1	1	1

Abbildung 11.17: Data-Latch oder transparentes D-Flip-Flop.

Eine Alternative ist das "Data Latch" oder "Transparentes D-Flip-Flop": Hier dient ein Eingang (C) als Kontrolleingang: Wenn C hoch ist, wird der Zustand des anderen Eingangs (D) auf den Ausgang Q übertragen, sonst bleibt der vorherige Zustand. Ist z.B. D gleichzeitig mit C hoch, so wird X hoch und Y niedrig. Somit wird auch Q hoch und $\bar{Q}$ niedrig.

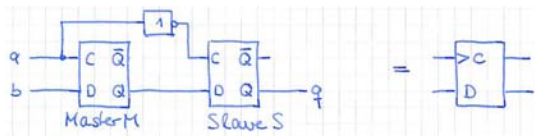


Abbildung 11.18: Aufbau eines Master-Slave-Flip-Flops.

Eine weitere Möglichkeit zur Vermeidung von Kollisionen bei der Eingabe basiert auf dem Master-Slave Prinzip. Hier wird der Eingangszustand zwischengespeichert und erst an den Ausgang weiter gegeben, wenn der Eingang wieder verriegelt ist. Dazu sind zwei Flip-Flops erforderlich, von denen der erste der Master und der zweite der Slave ist. Beide werden vom gleichen Taktimpuls gesteuert, wobei jedoch der Slave den invertierten Taktimpuls erhalt.

Die beiden Flip-Flops besitzen ein C- und einen D Eingang. Ist C hoch, wird das bit D weiter gegeben, für C=0 ist es verriegelt. Im Master-Slave Betrieb wird das Signal a auf den C-Eingang des Masters gegeben und zusätzlich invertiert auf den C-Eingang des Slaves. Wird a=1 gesetzt, so wird b im Master eingelesen, aber der Slave ist blockiert. Wird a=0 gesetzt, so übernimmt der Slave die Daten vom Master, gleichzeitig wird dieser verriegelt. Beim Master-Slave FF wird die eingehende Information zunächst nur in den Master übernommen und

erst wenn das Kontroll-Bit wieder in den “Low”-Zustand übergeht, wird die Information über den Slave weiter gegeben. Dies kann z.B. genutzt werden, um einen Zähler zu implementieren.

Ein solches Master-Slave FF FF kann auch als Teilerschaltung (divide by 2) verwendet werden.

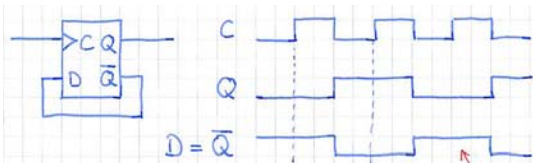
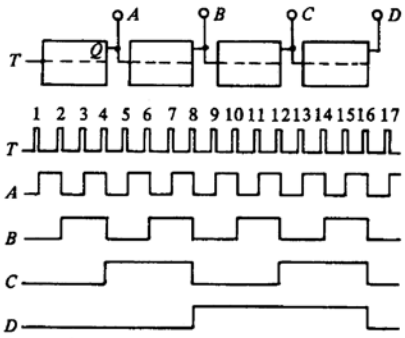


Abbildung 11.19: Teilerschaltung mit Flip-Flop.

Dafür wird der $\bar{Q}$ Ausgang auf den D-Eingang zurückgeführt, das Ganze wird nur noch durch den C-Eingang kontrolliert. Der Wert von $D = \bar{Q}$ wird jeweils auf der fallenden Flanke von C auf den Ausgang Q übertragen. Gleichzeitig wird der Wert von $D = \bar{Q}$ invertiert. Somit wird bei jedem zweiten Zyklus des Eingangssignals das Ausgangssignal invertiert und die resultierenden Ausgangssignale haben die halbe Frequenz des C Eingangssignals.



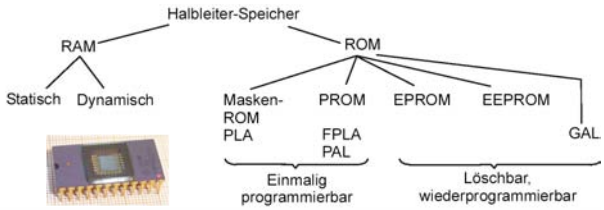
T	A	B	C	D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1
16	0	0	0	0

Abbildung 11.20: Binärzähler aus kaskadierten FlipFlops.

Das Prinzip lässt sich kaskadieren und auf diese Weise kann ein Binärzähler augebaut werden. Hier wird bei jeder aufeinander folgenden Stelle die Frequenz halbiert.

11.4 Speicher

11.4.1 Typenunterscheidung



RAM	Random Access Memory	Speicher mit wahlfreiem Zugriff
ROM	Read-Only-Memory	Nur-Lese-Speicher
PROM	Programmable ROM	Programmierbares ROM
EPROM	Erasable PROM	löschbares PROM
EEPROM	Electrically erasable PROM	elektrisch löschbares PROM
PLA	Programmable Logic Array	programmierbares logisches Feld
FPLA	Field PLA	durch Anwender programmierbar
GAL	Generic Array Logic	löschbares und wiederprogammierbares logisches Feld

Abbildung 11.21: Unterschiedliche Typen von Halbleiter-Speichern.

Halbleiter-Bauteile werden in unterschiedlichen Arten von Speichern verwendet. Man unterscheidet zunächst zwischen RAM (=Random Access Memory, Lese- und Schreibe Speicher) und ROM (Read-Only Memory) Speicher. Bei RAM wird zwischen statischem und dynamischem Speicher unterschieden, bei ROM wird unterschieden, ob er löschbar ist und wenn ja, wie.

Die Bezeichnung RAM deutet darauf hin, dass man beliebige Speicheradressen in beliebiger Reihenfolge ansteuern kann, im Gegensatz z.B: zu Festplatten, wo Daten in Blocks, resp. seriell ausgelesen werden.

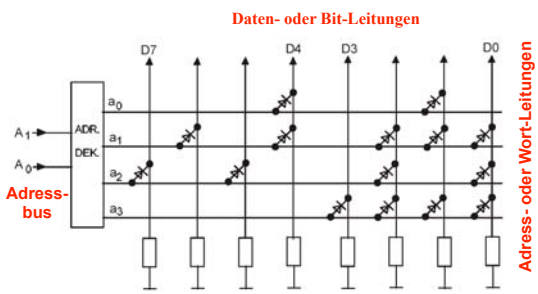


Abbildung 11.22: Adressierung von Speicherzellen in einem ROM.

Die Speicherzellen sind in einem zweidimensionalen Muster angeordnet. Beim Auslesen muss jeweils

die Spalten- und die Zeilenadresse aktiviert werden. Je nach Speicher können über eine einzelne Adresse auch mehrere bits (z.B. ein Byte) angesprochen werden.

11.4.2 ROM

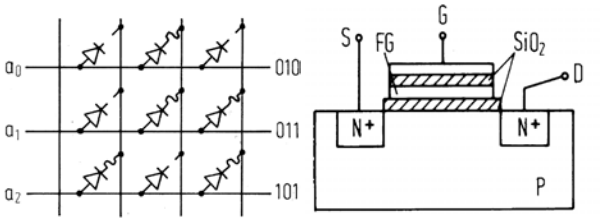


Abbildung 11.23: PROM (links) und EPROM (rechts).

In einem programmierbaren Speicher (PROM) kann die Sicherung für Zellen, die eine Null enthalten sollen, mit einem kurzen Stromimpuls weggebrannt werden. Danach ist Inhalt dieser Zellen nicht mehr änderbar.

Bei einem EPROM (=Erasable PROM) befinden sich an den Kreuzungen MOS-FETs mit einer schwimmenden Gate-Elektrode (Floating Gate MOS-FET). Die Programmierung erfolgt durch Aufladen dieses Gates mit 'heißen' Elektronen, die durch eine Überspannung erzeugt werden. Das Löschen erfolgt durch UV-Licht oder elektrisch (Dauer typ. 10 ms). Elektrisch löschbare Speicher erlauben teilweise auch eine sektionenweise Löschung. Sie werden auch als Flash-Speicher bezeichnet.

11.4.3 RAM

RAM=Random Access Memory = Schreib-Lese Speicher existiert in zwei Varianten, als statisches und als dynamisches RAM. Das dynamische RAM speichert die Information auf einem Kondensator, das statische RAM in einem D-Flip-Flop. Das D-RAM ist einfacher und benötigt weniger Platz, es muss jedoch ständig wieder aufgefrischt und nach jedem Lesezugriff erneut aufgeladen werden. Dies liegt daran, dass der Kondensator eine relativ kleine Kapazitat von ~ 50 fF besitzt. Deshalb bewirkt

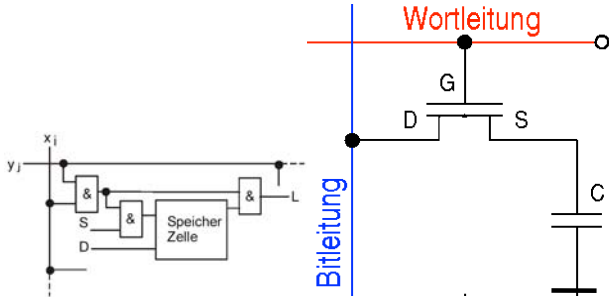


Abbildung 11.24: Statisches RAM (SRAM, links) und dynamisches RAM (DRAM, rechts).

das Auslesen eine so starke Entladung, daß anschließend alle längs der decodierten Zeile liegenden Kondensatoren in einem internen Schreibvorgang wieder auf den vorherigen Wert gebracht werden müssen.

Ein Schreibvorgang läuft in entsprechender Weise ab: Die Zeilenadresse wird decodiert, so daß wieder längs einer Zeile die MOS-Schalter geöffnet werden. Vom Eingangsverstärker wird das Bit auf alle Schreibverstärker gelegt. Mit der decodierten Spalte wird ein Schreibverstärker ausgewählt und das Bit in die entsprechende SpeicherzelIe geschrieben. Gleichzeitig werden auch alle Kondensatoren längs der decodierten Zeile automatisch wieder auf den alten Wert gebracht, da jedes Öffnen der MOS-Schalter bereits eine Teilentladung bedeutet.

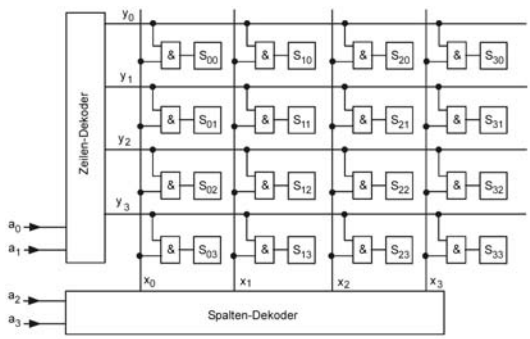


Abbildung 11.25: Adressierung von RAM.

Auch beim RAM werden die einzelnen Speicherzellen durch Zeilen- und Spalten adressiert. Der Zugriff wird durch ein UND-Gatter kontrolliert, welches durch die Zeilen- und Spalten Dekoder angesteuert wird.

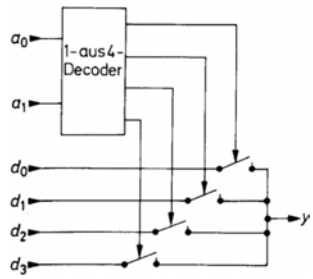


Abbildung 11.26: Decodierung von Adressleitungen.

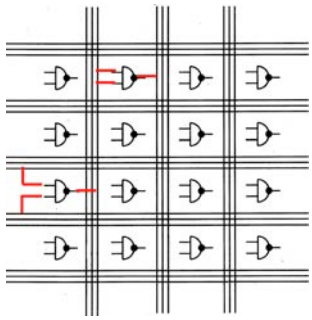


Abbildung 11.27: Field programmable gate array.

Dabei werden jeweils Adressleitungen decodiert: aus den n bits, welche die Adresse codieren, wird jeweils bestimmt, welche der 2^n Leitungen durchgeschaltet wird. So kann mit 2 Adressleitungen in einem 1-aus-4 Dekoder eine von vier Zeilen ausgewählt werden. Dieser Decoder kann z.B. auch in einem Multiplexer verwendet werden: man legt z.B. mit 2 Adressleitungen fest, welche von 4 Datenleitungen auf den Ausgang durchgeschaltet werden.

11.4.4 Programmierbare Logikbausteine

Das Ändern von Werten in adressierbaren Speicherbausteinen kann auch auf Logikbausteine ausgedehnt werden. Damit erhält man wehr viel mehr Möglichkeiten beim Aufbau von logischen Schaltungen als wenn man jede auf der Basis von einzelnen diskreten Elementen realisieren muss. Bausteine, welche diese Möglichkeit bieten, werden als ASIC (application-specific IC) bezeichnet, oder als PLDs (programmable logic devices). Das Grundprinzip ist dabei, dass man eine große Zahl von logischen Schaltelementen (UND, ODER etc.) auf einem Chip realisiert, wobei die logischen Verknüpfungen nachträglich, vom Anwender noch erstellt werden können. Geschieht das Erstellen der Verbindungen bei der Herstellung, so spricht man von MPGA (mask programmable gate array).

Beim FPGA (field-programmable gate array) können die Verbindungen zwischen den Logikgattern mit programierbaren Verbindungsleitungen hergestellt werden. Werden für die Verbindungen Transistorzellen verwendet, so kann die Logik jederzeit neu programmiert werden. Man spricht in diesem Fall von SRAM Technik. Verwendet man statt dessen Schmelzsicherungen, (Antifuse-Technik), so wird die Programmierung fest eingebrannt.

11.5 Digital-Analog Wandler (DAC)

11.5.1 Grundlagen

In unserer Umwelt oder im Labor fallen die meisten Signale in analoger Form an, d.h. es handelt sich um kontinuierlich variierende Signale. Es gibt aber eine Reihe von Gründen, diese Signale in digitale Werte zumzusetzen: dies erleichtert z.B. die Speicherung, Übermittlung und Verarbeitung, und es reduziert viele mögliche Fehlerquellen. Grundsätzlich haben analoge Signale einen belibig hohen Dynamikumfang. Allerdings wird dieser in der Praxis gegen unten durch unvermeidliches Rauschen und Störsignale limitiert, gegen oben durch die maximal zur Verfügung stehende Spannung oder Leistung. Digitale Signale hingegen haben grundsätzlich eine endliche Präzision, welche durch die Zahl der Stellen begrenzt wird, welche zu ihrer Darstellung zur Verfügung stehen. Die Zahl der Stellen kann aber heute in fast allen Fällen so groß gewählt werden, dass die Präzision dadurch nicht mehr beschränkt wird. Sind die Signale einmal digitalisiert, so können sie mit beliebiger Präzision und praktisch verschwindender Fehlerrate verarbeitet und gespeichert werden.

Eine wesentliche Motivation für die digitale Verarbeitung von Signalen ist der Dynamikumfang. So werden auf einer CD (Compact Disc) die Musik-

daten mit 16 bit gespeichert. Bei einer maximalen Spannung von 2V entsprechen 16 bit einer Auflösung von

$$2\text{V}\frac{1}{2^{16}} = \frac{2\text{V}}{65536} = 30\mu\text{V}.$$

In dB ausgedrückt sind dies 96,3 dB, d.h. dies deckt beinahe den gesamten Hörbereich des Menschen ab.

Aus diesen Gründen werden Daten immer häufiger in digitaler Form gespeichert und verarbeitet. Um diese digitale Welt mit der analogen Welt zu verbinden, benötigt man Schnittstellen, welche analoge Signale (z.B. Spannungen) in digitale Signale umwandeln und umgekehrt. Diese werden als Analog-zu-Digital Konverter (ADC), resp. Digital-zu-Analog Konverter (DAC) bezeichnet. Wir diskutieren zuerst die Umsetzung aus der digitalen in die analoge Welt.

11.5.2 Parallele vs. serielle Umsetzung

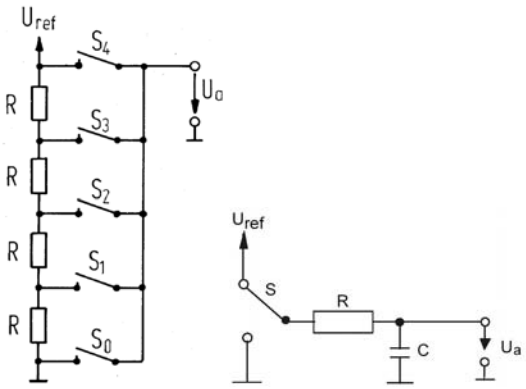


Abbildung 11.28: Parallelverfahren (links) und Zählverfahren (rechts) für DACs.

Für die digital-analoge Umwandlung werden verschiedene Verfahren verwendet. Beim Parallelverfahren verwendet man ein digitales Potentiometer. Die Ansteuerung erfolgt über einen 1-aus-n-Dekoder. Für N bit Auflösung braucht man hier 2^N Schalter, für 8 bit Auflösung also 256 Schalter.

Ein weiteres mögliches Verfahren ist das Zählverfahren mit Referenzfrequenz: in diesem Fall braucht man nur einen Schalter. Dieser wird periodisch geschlossen und geöffnet. Wegen des Integriergliedes ist diese Schaltung langsam. Sie findet Anwendung in Frequenzzählern (z.B. in einem Drehzahlmesser).

11.5.3 Wägeverfahren

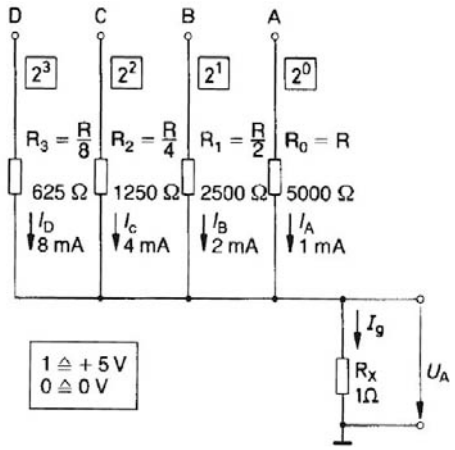


Abbildung 11.29: Wägeverfahren für DACs.

Beim Wägeverfahren werden gewichtete Ströme addiert. Für N Bit Auflösung braucht man N Schalter. Um die Ströme zu addieren, gibt am entsprechenden Eingang eine Spannung von z.B. 5 Volt, multipliziert mit dem Wert a_i des Bits (0 oder 1) auf einen Widerstand, der den Wert $R_i = R_0/2^i$ hat. Über diesen Widerstand fließt somit ein Strom $I_0 \cdot 2^i a_i$. Diese Ströme werden addiert und über dem Ausgangswiderstand in eine Spannung umgewandelt. Im Beispiel wird angenommen, dass ein logischer Wert von 1 durch eine Spannung von 5 V dargestellt wird. Verwendet man für den Widerstand $R_0 = 5k\Omega$, so fließt auf der LSB Linie ein Strom von 1 mA (falls das Bit gesetzt ist), auf der zweiten 2 mV etc. Über dem Ausgangswiderstand werden diese Ströme in Spannungen mit dem gleihen numerischen Wert umgewandelt.

Abb. 11.30 zeigt eine realistischere Variante des Wägeverfahrens: Die Eingangswerte werden auf die Basis eines Transistors gegeben. Dadurch werden die Ströme der einzelnen Bits voneinander entkoppelt. Da kein Strom in den Eingang des OPs fließt, muss die Summe der Ströme über den Rückkopplungsiderstand R_N abfließen. Die Ausgangsspannung nimmt damit den gewünschten Wert an.

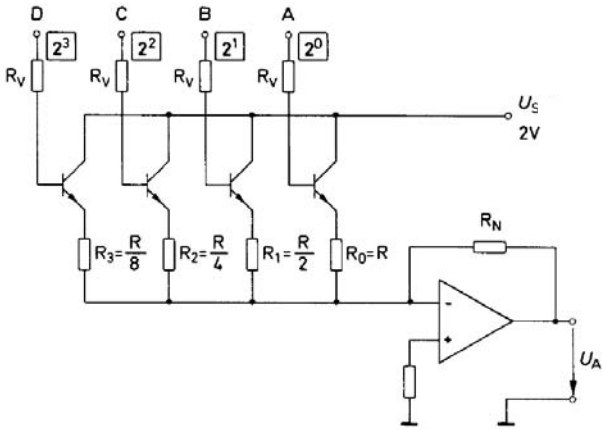


Abbildung 11.30: Eine realistischere Variante für das Wägeverfahren.

11.6 Analog-Digital Wander (ADC)

11.6.1 Übersicht

ADC-Typen: analog zu DACs	Parallel- Flash- Wandler	Wäge- Sukzessive Approximation	Zähl-Verfahren
Schrittzahl	1	N	bis zu 2^N
Referenzspannungen	2^N	N	1
Merkmale	aufwendig, schnell	oft guter Kompromiss	einfach, langsam
Anwendungen	Video	Audio	Präzisionsmess-technik, DVM
typische Auflösung	4-8 Bit	10-14 Bit	12-18 Bit
Frequenzbereich	10 MHz - 100 MHz	10 kHz - 1 MHz	< 1 kHz

Abbildung 11.31: Übersicht über und Vergleich der verschiedenen Verfahren.

Die einzelnen Verfahren haben unterschiedliche Vor- und Nachteile.

11.6.2 Flash ADC

Ein Analog-zu-Digital Converter (ADC) ist im Wesentlichen ein DAC mit einem Vergleich. Entsprechend gibt es die gleichen Methoden: Flash Wandler sind schnell aber aufwändig, das Zähl-Verfahren ergibt eine hohe Präzision, ist einfach, aber langsam.

Beim Parallelverfahren wird das analoge Eingangssignal mit 2^N Referenzsignalen verglichen, welche

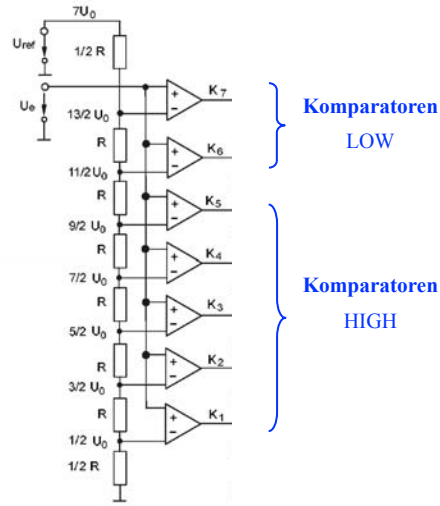


Abbildung 11.32: ADC mittels Parallelverfahren.

über Spannungsteiler erzeugt werden. Dies wird als Flash-ADC bezeichnet. Im Beispiel von Abb. 11.32 beträgt das Signal 5 U_0. Somit sind die Komparatoren K_6 und K_7 niedrig, während die Komparatoren K_1 bis K_5 hoch sind (das Signal liegt über ihrer Referenzspannung). Diese Ausgangswerte der Komparatoren werden (getaktet mit einer Clock) auf einen Dekoder gegeben, welcher daraus die Werte für die einzelnen Bits berechnet.

Dieses Verfahren ist sehr schnell, da es nur einen Takt-Schritt benötigt; es ist aber auch sehr aufwändig, da es für ein N-bit Signal 2^N Komparatoren benötigt.

11.6.3 Zähl- und Wägeverfahren

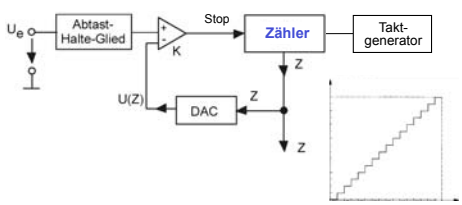


Abbildung 11.33: ADC mittels Zählverfahren.

Beim Zählverfahren erzeugt ein Zähler Pulse, mit denen ein DAC jeweils um eine Stufe hochgezählt wird, bis sein Ausgang dem Wert des Eingangssignals erreicht. Stellt der Komparator fest, dass der

Ausgangswert des DACs den Eingangswert des Signals erreicht hat, so stoppt er den Zähler.

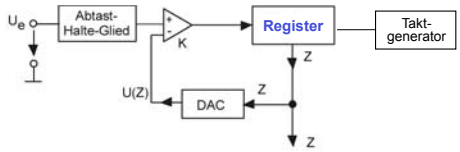


Abbildung 11.34: ADC mittels Wägeverfahren.

Beim Wägeverfahren wird der Vergleich bitweise vorgenommen: zunächst wird das MSB = 1 gesetzt und der DAC erzeugt die entsprechende Spannung. Liegt diese unterhalb der Eingangsspannung, $U_e > U(Z)$, so bleibt das bit gesetzt, sonst wird es gelöscht. Danach wird die Spannung, welche dem zweiten Bit ensrpicht, addiert, und das Verfahren wird widerholt. Somit wird der Eingangswert bei jedem Takt um ein Bit angenähert, in sukzessive abnehmenden Schritten. Insgesamt benötigt man damit N Taktzyklen.

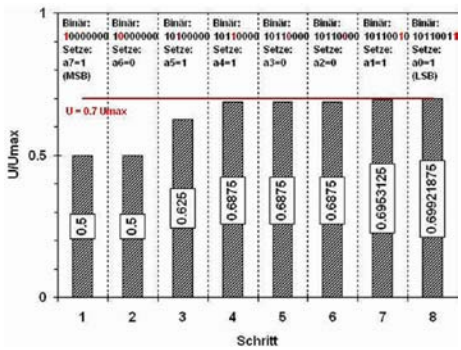


Abbildung 11.35: Sukzessive Approximation von $U = 0,7$ V.

Auf diese Weise kann man z.B. mit 8 bits eine Spannung von 0,7 V durch die Binärreihe 10110011 annähern. Hier wurde als maximale Spannung (=11111111) 10 V angenommen. Die Auflösung beträgt dann 1V/256 = 3,9 mV.

11.6.4 Delta-Sigma Wandler

Beim Delta-Sigma Wandler wird ein wesentlicher Teil der Arbeit in die digitale Nachbearbeitung verschoben. Das zeitabhängige Eingangssignal kommt

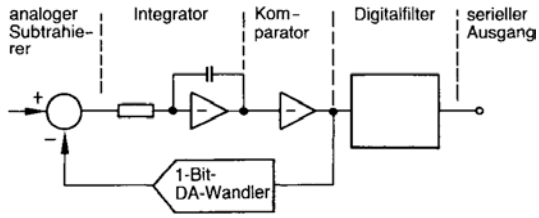


Abbildung 11.36: Delta-Sigma Wandler.

über den analogen Subtrahierer zum Integrator und verursacht an dessen Ausgang ein Signal, das der Komparator mit eins oder null bewertet. Der 1-Bit-Digital- Analog-Wandler erzeugt daraus eine positive oder negative Spannung, die über den Subtrahierer den Integrator wieder auf null zuriickzieht. Das nachgeschaltete Digitalfilter filtert die dadurch erzeugten binären Werte. Je nach digitaler Nachbearbeitung kann man daraus eine relativ langsame Sequenz von sehr hoch aufgelösten Werten (z.B. 24 bit) erhalten, oder einen sehr schnellen Strom von weniger hoch aufgelösten Werten.

11.7 Grenzen der Umsetzung

11.7.1 Quantisierungsfehler

Bei jedem Wandler findet man Abweichungen zwischen dem idealen und dem realen Ausgabewert. Diese Abweichungen umfassen vor allem Nichtlinearitäten und Quantisierungsrauschen.

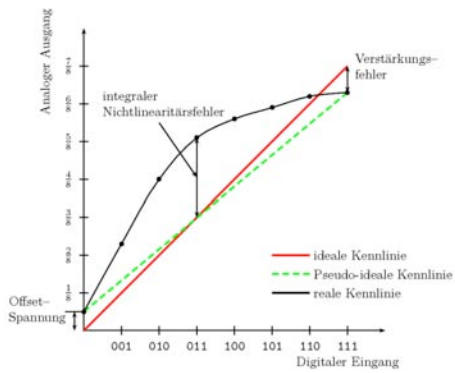


Abbildung 11.37: Kennlinie eines realen ADCs.

Abb. 11.37 zeigt einige der wichtigsten Fehlerquellen eines DACs: während die idealen Werte auf der

roten Kennlinie liegen sollten, findet man z.B. Offsetspannungen (d.h. eine endliche Ausgangsspannung für den Eingangswert 000), Verstärkungsfehler (d.h. eine falsche Steigung der Kennlinie), sowie Nichtlinearitätsfehler, d.h. die reale Kennlinie ist keine Gerade.

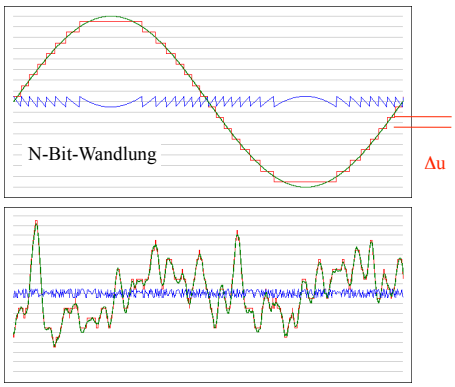


Abbildung 11.38: Quantisierungsfehler eines ADCs.

Da eine endliche Anzahl Bits ein reelles Signal nicht exakt wiedergeben kann, entsteht dadurch ein Quantisierungsfehler. Der obere Teil von Abb. 11.38 zeigt den resultierenden Fehler für ein sinusförmiges Signal, welches mit 4 Bit digitalisiert wird. Das digitalisierte Signal entspricht jeweils derjenigen ganzen Zahl, welche am nächsten beim reellen Wert liegt. Es entsteht somit ein Fehler von bis zu einem halben bit. Für nichtperiodische Signale gleicht der Fehler stärker einem Rauschen.

11.7.2 Abtastrate

Das Abtasten muss mit genügend hoher Rate erfolgen: Signale, deren Frequenz höher ist als die halbe Abtastfrequenz, werden "zurückgefaltet". Dies wird als "Nyquist-Shannon-Theorem" bezeichnet. Wird ein monochromatisches Signal einmal pro Periode abgetastet, so erscheint es konstant: siehe Abb. 11.39 oben. Ist die Abtastrate kleiner als die halbe Frequenz, so wird die gemessene Frequenz kleiner als die wirkliche Frequenz:

$$f_{\text{rekon}} = |f_{\text{Signal}} - f_{\text{Abtast}}|.$$

Im Beispiel von Abb. 11.39 betragen in der oberen Hälfte die beiden Frequenzen $f_{sig} = f_{dig} = 1\,\text{s}^{-1}$ und

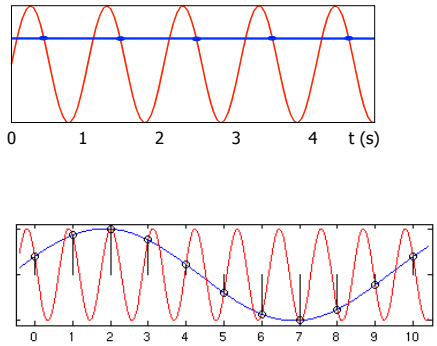


Abbildung 11.39: Oben: Abtasten eines sinusförmigen Signals einmal pro Periode; unten: $f_{dig} = 0{,}9\ \text{s}^{-1}$, $f_{sig} = 1{,}0\ \text{s}^{-1}$.

damit $f_{\text{rekon}} = 0$. Im unteren Beispiel ist $f_{sig} = 0{,}9\ s^{-1}$, $f_{dig} = 1{,}0\ s^{-1}$ und damit $f_{\text{rekon}} = 0{,}1\ \text{s}^{-1}$. Dieses 'Zurückfalten' des Signals zu einer kleineren Frequenz wird als 'Aliasing' bezeichnet.

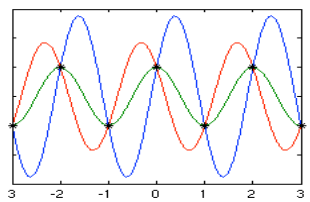


Abbildung 11.40: Kritische Frequenz für digitale Abtastung.

Ein eindeutiges Ergebnis erhält man, falls die Abtastrate f_{Abtast} größer ist als die doppelte Signalfrequenz, $f_{Abtast} > 2f_{Signal}$. Wie in Abb. 11.40 gezeigt, ergeben bei $f_{Abtast} > 2f_{Signal}$ noch mehrere analoge Signale die gleichen digital gemessenen Werte.

Das Abtasttheorem wurde zuerst von Harry Nyquist (1889-1976) postuliert und von Claude Shannon (1916-2001) bewiesen. Es sollte nicht nur in Bezug auf das Signal, sondern auch auf das Rauschen beachtet werden: Breitbandiges Rauschen wird bei Verletzung des Abtasttheorems in das Signalband hinein gefaltet und verschlechtert so das Signal-zu-Rausch Verhältnis.

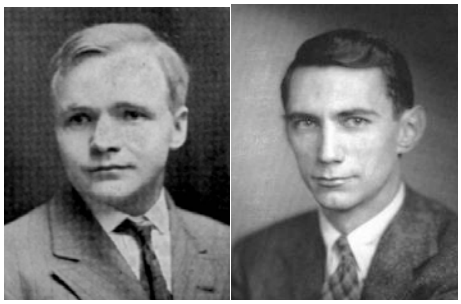


Abbildung 11.41: Harry Nyquist (1889 - 1976, links) und Claude E. Shannon (1916-2001, rechts).

11.7.3 Oversampling

In manchen Fällen lohnt es sich, die Abtastrate höher anzusetzen als den Wert, der sich aus dem Nyquist-Theorem ergibt. Dies ist u.a. dann nützlich, wenn die digitale Auflösung beschränkt ist. Dies gilt sowohl für die analog-zu-digital Konversion, wie auch beim umgekehrten Vorgang.

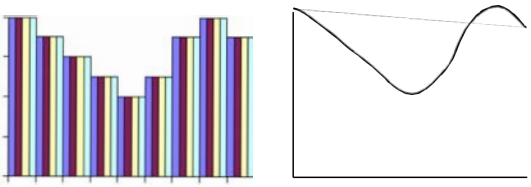


Abbildung 11.42: Digital-Analog Konversion mit vierfachem Oversampling. Links digitales Signal, rechts analoges, gefiltertes Signal.

Abb. 11.42 zeigt ein Beispiel: Eine Wellenform wird mit vierfachem Oversampling erzeugt. Ein Tiefpassfilter glättet, so dass die Unterschiede zwischen den Digitalisierungsstufen reduziert werden. Dadurch entsteht ein analoges Signale, welches glatter aussieht als ohne Oversampling.