13.13 Sequenzielle Schaltungen (abfolgeabhängige Schaltungen)

Bei den bisherigen **kombinatorischen** Schaltungen hingen die Ausgänge nur von der **aktuellen** Eingangskombination ab.

Bei sequenzielle Schaltungen hängen die Ausgangszustände nicht nur von der aktuellen, sondern auch **von früheren Eingangskombinationen** ab. Es wirkt sich der zeitliche Ablauf (die Sequenz) bestimmter Zustände und Zustandsänderungen an den Eingängen aus.

Das einfachste ist das Flip-Flop. Ein Flip Flop ist eine Kippschaltung, die zwei Zustände annehmen kann. Bereits im Kapitel über Transistoren haben wir Kippstufen behandelt.

- Astabile (kein stabiler Zustand, automatisches Hin- und Herkippen)
- Monostabile (ein stabiler Zustand, automatisches Zurückkippen aus instabilem Zustand
- Bistabile (zwei stabile Zustände)

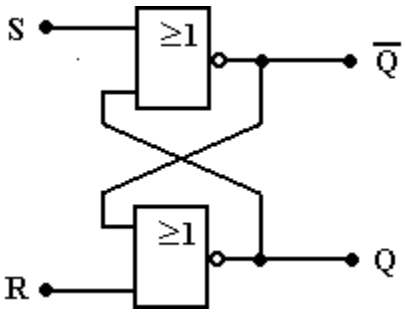
Flipflops in der Digitaltechnik sind im allgemeinen bistabil, haben also zwei **stabile** Ausgangszustände. Diese Ausgangszustände können nur durch bestimmte Änderungen der Eingänge geändert werden.

13.13.1 SR-Flipflop

Ein SR-Flipflop ist ein **S**et-**R**eset-Flip-Flop.

- Es hat die beiden Eingänge **S** und **R.** Mit diesen Eingängen kann man es in den einen Zustand **setzen** und aus diesem wieder zu**rücksetzen**.
- Es hat auch zwei Ausgänge **Q** und **Q "quer"**, die immer die Negation voneinander bilden.

Eine solche Schaltung lässt sich mit beispielsweise aus zwei NOR-Gattern verwirklichen.
Eine solche Schaltung heisst **NOR-Latch** (ein "Schnappschloss" aus NOR-Gattern).
Betrachten wir einmal die nebenstehende Schaltung und versuchen wir, eine Wahrheitstabelle aufzustellen.

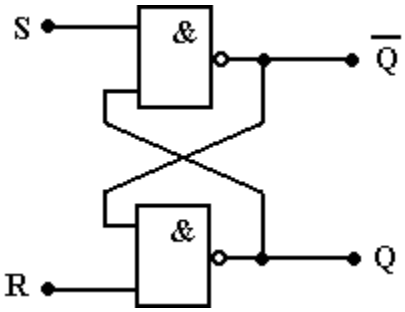


Bis jetzt haben wir das Verhalten einer "solchen" Schaltung problemlos mit einer Wahrheitstabelle angeben können. Hier werden wir aber feststellen, dass dies Probleme gibt: Man kann die Tabelle **nicht mehr eindeutig** mit Nullen und Einsen füllen, sondern muss Variablen (zum Beispiel X) einfügen, die einen **früheren Zustand** kennzeichnen.

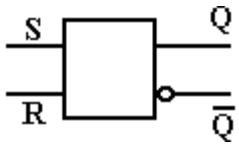
In der folgenden Wahrheitstabelle sind die vier möglichen Kombinationen der Eingänge S und R absichtlich nicht in der gewohnten "binären" Reihenfolge aufgeführt.

Fall	S	R	Q	Q quer	Kommentar
1	1	0	1	0	Setzen des Flipflops
2	0	1	0	1	Rücksetzen des Flipflops
3	0	0	X	NOT X	**Speicherfall**, das heisst, die Ausgänge **behalten den vorherigen** Zustand (1 0 oder 0 1)
4	1	1	0	0	"Verbotener" Fall, weil die beiden Ausgänge nun den gleichen Zustand haben, somit ist es keine Flipflopfunktion mehr. Dieser Fall ist auch nicht stabil und kann deshalb nicht gespeichert werden. Der Ausgangszustand 1 1 ist übrigens nicht möglich, da sich die beiden Gatter gegenseitig blockieren.

Die gleiche Schaltung lässt sich aus NANDs bauen.
Eine solche Schaltung heisst **NAND-Latch**. Auch hier wollen wir die Wahrheitstabelle betrachten.



Symbol des RS-Flipflops:



Fall	S	R	Q	Q quer	Kommentar
1	1	0	1	0	Setzen des Flipflops
2	0	1	0	1	Rücksetzen des Flipflops
3	1	1	X	NOT X	**Speicherfall**, das heisst, die Ausgänge **behalten den vorherigen** Zustand (1 0 oder 0 1)
4	0	0	1	1	"Verbotener" Fall, weil die beiden Ausgänge nun den gleichen Zustand haben, somit ist es keine Flipflopfunktion mehr. Dieser Fall ist auch nicht stabil und kann deshalb nicht gespeichert werden. Der Ausgangszustand 0 0 ist übrigens nicht möglich, da sich die beiden Gatter gegenseitig blockieren.

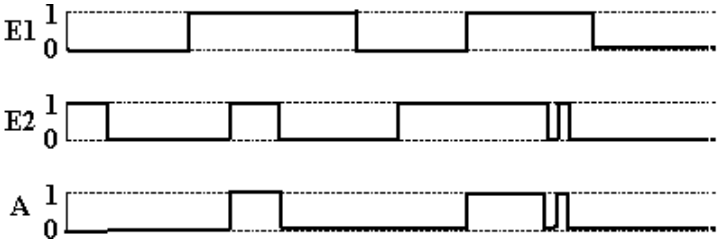
Wie wir nun gesehen haben, ist die Wahrheitstabelle nicht mehr eindeutig ausfüllbar, da die Eingangskombination im Fall 3 je nach "Vorgeschichte" zu unterschiedlichen Ergebnissen an den Ausgängen führen kann. Denn der Fall 3 kann entweder aus Fall 1 oder aus Fall 2 entstehen.
Damit wäre bewiesen, dass es sich hier nicht mehr um kombinatorische Schaltungen handelt.
Man sucht nun auch nach Wegen, dieses Verhalten der Schaltung auf andere Weise darzustellen. Solche Darstellungen sind die Impulsdiagramme:

13.13.2 Impulsdiagramme

Impulsdiagramme (oder Signaldiagramme) können das zeitliche Verhalten digitaler Signale optimal veranschaulichen:

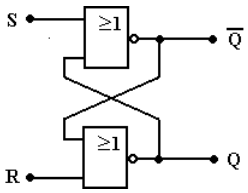
Man macht eine Grafik mit jedem Eingang und jedem Ausgang:
- Von links nach rechts läuft die Zeit
- Alle Signale liegen zeitlich genau übereinander
- Auf der Vertikalen wird für jedes Signal der Logikpegel aufgetragen, unten 0, oben 1.
- Mit den Eingangssignalen versucht man wie üblich jede mögliche Kombination durchzuspielen, und zwar sowohl bezüglich der Zustände wie auch der Änderungen.
- Die Ausgangssignale richten sich nach den Eingängen und den eingesetzten Funktionen
- Mit dieser Darstellung können nicht nur alle Zustände, sondern auch alle Zustands**änderungen** untersucht und veranschaulicht werden!

Selbstverständlich können Impulsdiagramme auch für simple kombinatorische Schaltungen angegeben werden. Beispiel: Das Impulsdiagramm eines UND-Gatters:



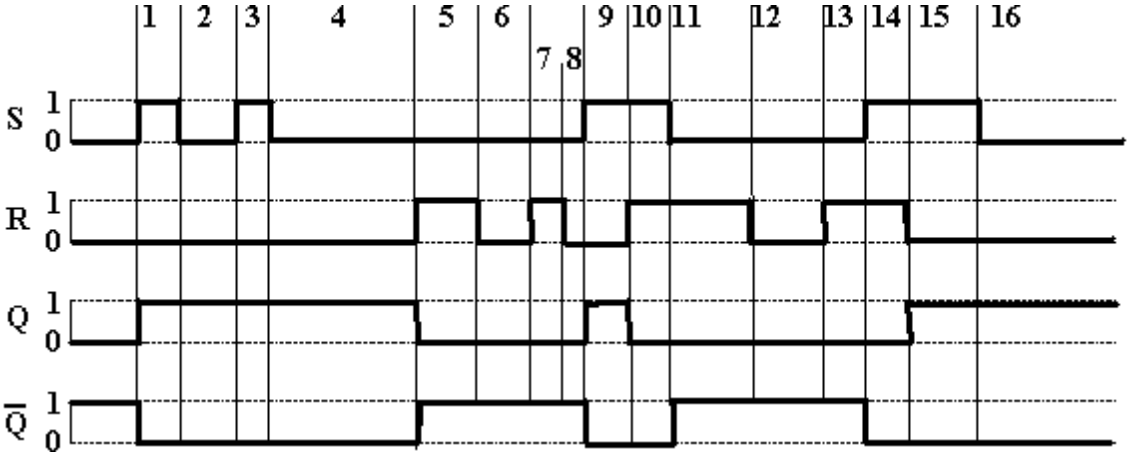
AND-Funktion: Das Ausgangssignal ist immer nur dort oben, wo beide Eingänge oben sind.

Für unser NOR-Latch ergibt sich das folgende Impulsdiagramm:



Wahrheitstabelle:

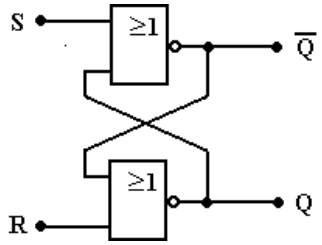
Fall	S	R	Q	Q quer	Kommentar
1	1	0	1	0	Setzen des Flipflops
2	0	1	0	1	Rücksetzen des Flipflops
3	0	0	X	NOT X	**Speicherfall**,
4	1	1	0	0	"Verbotener" Fall,



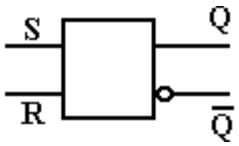
Zeitliche Phase	Kommentar
1	Set: Das Flipflop wird gesetzt, Q geht auf 1, Q quer auf 0
2	Speicherfall: die Ausgänge bleiben im vorherigen Zustand und ändern sich nicht
3	Ein erneutes Setzen ändert natürlich auch nichts
4	Speicherfall
5	Reset: Zurücksetzen des Flipflops, Q geht wieder auf 0
6	Speicherfall
7	Erneutes Zurücksetzen ändert natürlich nichts
8	Speicherfall
9	Setzen
10	Verbotener Fall: Reset-Signal kommt gleichzeitig mit noch bestehendem Setz-Befehl, Ausgänge werden beide gleich (0)
11	Setz-Signal verschwindet, Reset bleibt, Flipflop wird zurückgesetzt, Q auf 0
12	Speicherfall
13	Erneutes Reset ändern nichts
14	Verbotener Fall: Setz-Signal gleichzeitig mit noch bestehendem Reset, Ausgänge beide gleich (0)
15	Reset verschwindet, Set bleibt, Flipflop wird gesetzt, Q geht auf 1
16	Speicherfall

Das vorher besprochene SR-Flipflop ist ein **nicht taktgesteuertes** Flipflop. Das heisst, es ändert seinen Zustand gerade dann, wenn der entsprechende Setz- oder Rücksetzeingang aktiviert wird.
Von viel grösserer Bedeutung sind aber die **taktgesteuerten** oder **synchronen** Flipflops. Taktgesteuert bedeutet, dass das Flipflop erst auf den Taktbefehl hin auf die Zustände an den Eingängen reagiert. Der Taktbefehl ist im Prinzip ein weiterer Eingang. Dies hat den Vorteil, dass mehrere Schaltkreise gleichzeitig schalten können, also synchron arbeiten, im gleichen Takt. Man spricht dann von einer getakteten Schaltung. Den Begriff kennen Sie aus der PC-Technik, wo der Prozessor getaktet ist oder auch der Datenbus getaktet ist.

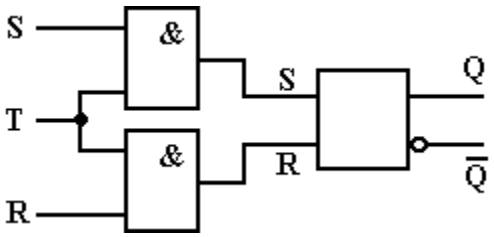
13.13.3 Takt**zustands**gesteuerte Flipflops (FFs)



Die Grundschaltung eines SR-Flipflops kann bekanntlich z.B. aus zwei NORs aufgebaut werden. Man verwendet als Abkürzung das Symbol rechts.

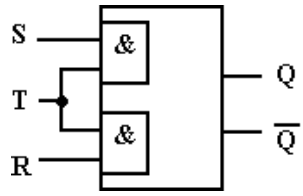


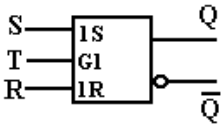
Aus diesem **nicht**taktgesteuerten FF kann man durch weitere Elemente ein **taktgesteuertes** FF machen:
Die beiden Eingänge (**S**etz und **R**ücksetz) werden je mit dem Taktsignal **T** UND-verknüpft.
Der Flipflop-Block reagiert also auf anstehende Zustände bei R und S erst dann, wenn auch T aktiv (1) ist.



Das heisst, in allen Fällen, wo **T = 0** ist, haben wir zusätzlich den **Speicherfall**, **un**abhänig von S und R. Die Ausgänge bleiben bei T = 0 also unverändert. Der "verbotene" Zustand (alle Eingänge = 1) kann hier immer noch auftreten.
Dieses Flipflop nennt man **takt*zustands*gesteuert**, weil es auf den 1-**Zustand** des Takteingangs reagiert.

Als Schaltbilder werden je nach Norm folgende Symbole verwendet:





Das rechte Symbol ist mit der so genannten Abhängigkeitsnotation bezeichnet.

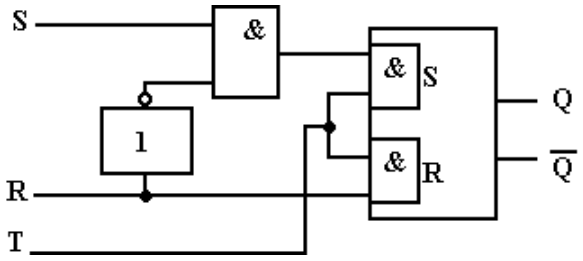
- Buchstaben im Symbol bezeichnen **Funktionen,** an Leitungen bezeichnen sie **Signale**.
- **G** bedeutet UND-Abhängigkeit auf die andern Eingänge mit gleicher Ziffer.
- Gleiche Ziffern deuten die gegenseitige Abhängigkeit der Eingänge an.
- Ein **steuernder** Eingang hat die Ziffer **nach** dem Buchstaben
- Ein gesteuerter Eingang hat die Ziffer **vor** dem Buchstaben

In diesem Fall heisst das:
Der Eingang Nr. 1 (T) wirkt UND-verknüpft auf die beiden Eingänge R und S.

Vermeidung der verbotenen Fälle:

➢ SR-FF mit dominierendem R-Eingang:

Durch geeignete Beschaltung der Eingänge kann man die verbotenen Eingangskombinationen vermeiden bzw. einen definierten Zustand des Flipflops herbeiführen:

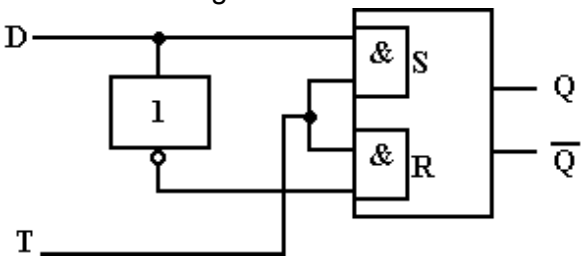


Dies ist ein FF mit dominierenden **R**eset-Eingang.
Das heisst, wenn R gleichzeitig mit S aktiv ist, so dominiert R, das heisst, das FF wird bei T = 1 zurückgesetzt.
Ein verbotener Fall kann somit nicht eintreten. Die zugehörige Wahrheitstabelle lautet dann:

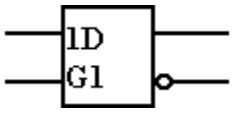
Fall	S	R	Q	Q quer	Kommentar
1	1	0	1	0	Setzen des Flipflops
2	0	1	0	1	Rücksetzen des Flipflops
3	0	0	X	NOT X	**Speicherfall**, X = vorheriger Zustand des Ausgangs
4	1	1	**0**	**1**	Der "verbotene Fall" tritt hier nicht ein, er führt eindeutig zum Rücksetzen des FFs.

➢ D-Flipflop:

Eine weitere Möglichkeit ist das direkte Invertieren von S- und R-Eingang.



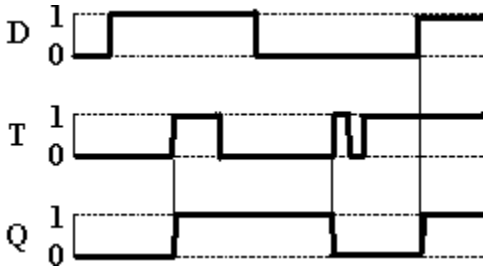
Das kann abgekürzt werden mit dem Symbol:



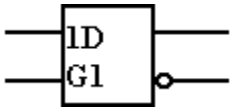
Dadurch entsteht nur noch **ein** (Daten-)Eingang, der mit D abgekürzt wird.
Das Flipflop heisst **D**-Flipflop (D von Delay = Verzögerung): Es leitet das **D**atensignal **D** verzögert an den Ausgang weiter, nämlich erst auf das Taktsignal hin.

Oder anders gesagt:
- Bei T = 1 ist das Flipflop **transparent**, jeder Eingangzustand erscheint direkt am Ausgang
- Bei T = 0 bleibt der letzte Zustand erhalten

Impulsdiagramm eines takt**zustands**gesteuerten D-FFs:



Symbol:



Diese Flipflops sind ebenfalls sehr verbreitet und als fertige Bausteine im Handel erhältlich, so dass keine externen Gatter dazugeschaltet werden müssen.

13.13.4 Takt**flanken**gesteuerte Flipflops

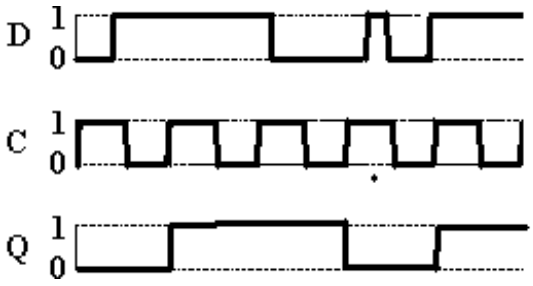
Wesentlich häufiger ist es erforderlich, dass ein Flipflop nur auf die **Aenderung** des Taktsignals, auf eine sogenannte **Taktflanke**, reagiert und nicht allein auf dessen Zustand. Dieses Verhalten kann durch einen geeigneten inneren Aufbau erreicht werden.
Der Zeitpunkt des Schaltens ist so genauer voraussagbar, da eine **Taktflanke** einen bestimmten Zeit**punkt** darstellt, während ein **Taktzustand** während einer längeren Zeit**dauer** herrscht.

Mögliche Flanken eines Taktsignals:	**verwendete Symbole:**
• Eine Aenderung von 0 nach 1 ist eine positive Flanke	⇑
• Eine Aenderung von 1 nach 0 ist eine negativeFlanke	⇓

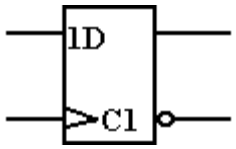
Uebrigens: Ein Taktsignal ist meistens ein **periodisches** Rechtecksignal.

Funktionsweise eines **positiv ein**flankengesteuerten D-FFs:
- Genau im **Moment** der positiven **Flanke** übernimmt der Ausgang Q den Zustand des Eingangs D. Unmittelbar darauf herrscht der Speicherfall bis zur nächsten pos. Flanke

Impulsdiagramm eines **positiv ein**flankengesteuerten D-FFs:

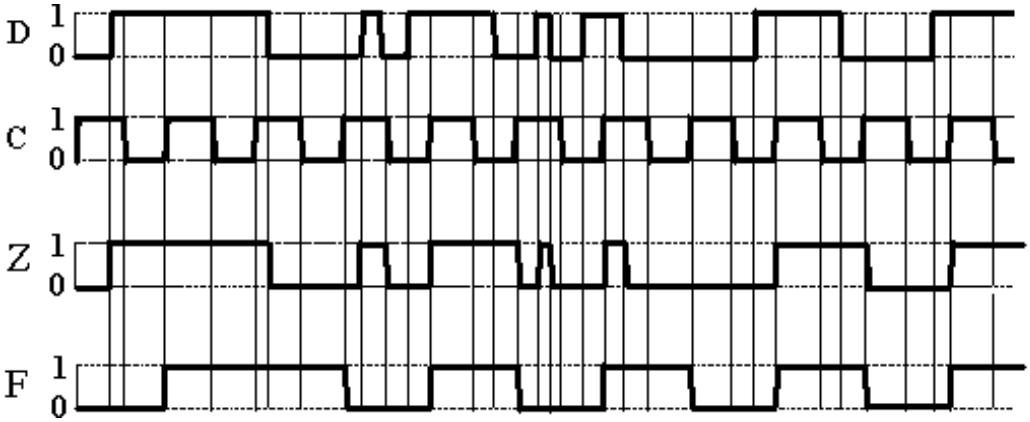


Symbol:
Der Steuereingang erhält einen Pfeil, der auf die Flankensteuerung hinweist.
Das Steuersignal wird dabei meist Clock-Signal genannt (clock = Uhr = Takt), abgekürzt CLK.
Die Funktion heisst dann **C.**



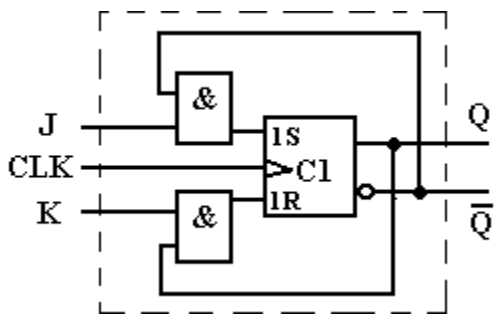
Merkmale des takt**flanken**gesteuerten im Unterschied zum takt**zustands**gesteuerten FF:
- Taktflankengesteuerte FFs können nicht transparent sein (Eingang wird nie dauernd an den Ausgang weitergeleitet).
- Eine Aenderung am Ausgang kann nur in den durch das Taktsignal definierten Zeitpunkten erfolgen (z.B. zur Zeit der positiven Flanke).
- Jeder Ausgangszustand dauert **mindestens einen Taktzyklus** lang oder ein Vielfaches davon.

Hier noch im Ueberblick die Gegenüberstellung eines takt**zustand**sgesteuerten (Ausgang **Z**) und eines takt**flanken**gesteuerten D-FFs (Ausgang **F**) bei gleichen Eingangssignalen D und C:

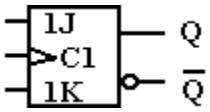


13.13.5 Einflankengesteuertes JK-Flipflop

Unter 13.13.3 haben wir aus einem SR-Flipflop ein taktgesteuertes FF gemacht, und dann noch durch zwei mögliche "Umbauten" den "verbotenen Fall" vermieden (D-Flipflop).
Es gibt weitere Möglichkeiten, den verbotenen Fall zu behandeln, nämlich durch eine definierte Rückführung der Ausgänge auf die Eingänge. Dies war bisher bei Kombinatorischen Schaltungen verboten, weil sich daraus Schaltungsschäden oder undefinierte Funktionen ergeben können. Es ist jedoch hier erlaubt, weil es separate Eingänge sind, auf die rückgeführt wird, und weil man Speicherfälle ja bewusst will.
Der Takteingang wird wiederum mit "Clock" bezeichnet:



So ist ein JK-FF aufgebaut. Die Eingänge werden J und K genannt, diese Buchstaben haben aber keine spezielle Bedeutung.
Das vereinfachte Symbol für das gestrichelt Umrandete ist:

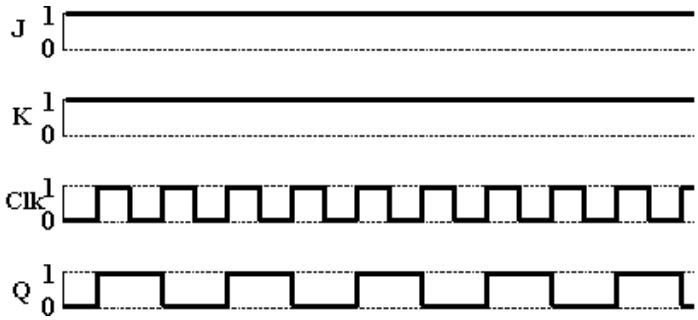


Es lässt sich sogar eine Wahrheitstabelle aufstellen, wobei die Eingänge nicht nur Zustände enthalten, sondern eben auch Flanken.

Die Wahrheitstabelle (3 Eingänge, 8 Fälle):
(Der Index n bezeichnet die Nummer einer Taktflanke, der Index n+1 bezeichnet also den Zustand nach der nächstfolgenden Taktflanke)

Fall	**Clk**	**J**	**K**	Q_{n+1}	$\overline{Q_{n+1}}$	**Bedeutung**
1	0	0	0	Q_n	$\overline{Q_n}$	Speicherfall
2	0	0	1	Q_n	$\overline{Q_n}$	Speicherfall
3	0	1	0	Q_n	$\overline{Q_n}$	Speicherfall
4	0	1	1	Q_n	$\overline{Q_n}$	Speicherfall
5	⇑	0	0	Q_n	$\overline{Q_n}$	Speicherfall
6	⇑	0	1	**0**	**1**	Rücksetzen des Flipflops
7	⇑	1	0	**1**	**0**	Setzen des Flipflopgs
8	⇑	1	1	$\overline{Q_n}$	Q_n	Toggle (Kippschalterfunktion) *

* Neu und interessant ist dabei der Fall 8: Die Wahrheitstabelle besagt, dass der nächstfolgende Zustand genau das Umgekehrte des vorherigen Zustands ist. Mit jeder positiven Clock-Flanke wechselt das Flipflop seinen Zustand, englisch mit "toggle" (kippen) bezeichnet. Wie sieht das Impulsdiagramm für den Fall 8 (J=K=1) aus?

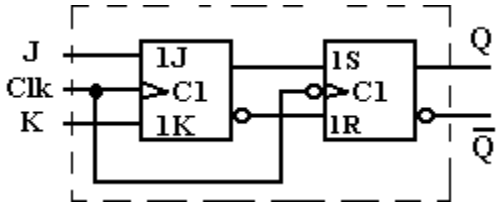
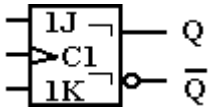


Feststellung: Q hat genau die doppelte Periodendauer oder halbe Frequenz wie Clk!
Anwendungen:

- Impulsschalter (Ein- und Ausschalten eines Verbrauchers durch Knopfdruck auf Impulstaste)
- **Frequenzteiler**: Ein Signal am Clock-Eingang kann exakt **durch 2 geteilt** werden. Dies ist die Grundlage für **digitale Zähler!** (Siehe später)

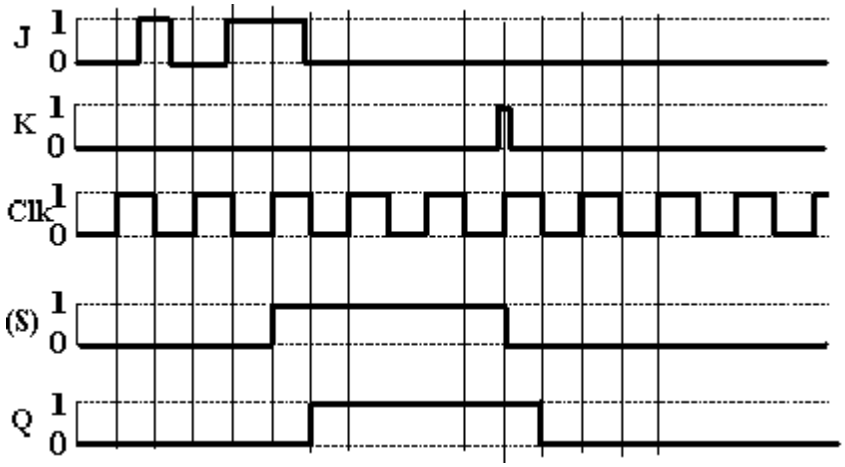
13.13.6 Zweiflankengesteuertes JK-Flipflop (Master-Slave-Flipflop)

Aus zwei JK-Flipflops oder einfacher aus einem JK-FF und einem SR-FF kann man ein sogenanntes Master-Slave-Flipflop machen. Mit einem solchen FF kann die Reaktion des Ausgangs um einen Taktschritt verzögert werden:

Aufbau: Das erste ist der Master und reagiert auf die positive, das zweite ist der Slave und reagiert auf die negative Flanke (deshalb das Ringlein am Takteingang).  	Das vereinfachte Symbol ist fast gleich wie das des normalen JK-FFs (zusätzliche Haken vor den Ausgängen, welche dass Zurückhalten der Information andeuten sollen):  Die Wahrheitstabelle ist die gleiche wie beim normalen JK-FF.

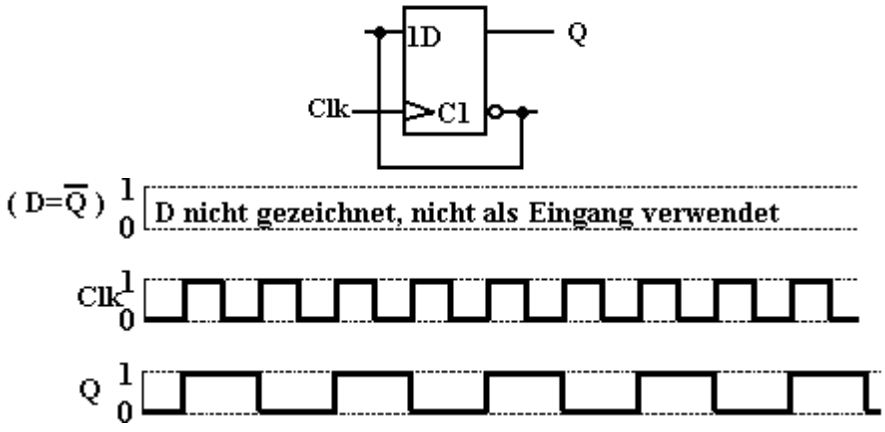
Dieses Prinzip des Zwischenspeicherns von Information wird bei **Schieberegistern** angewendet, welche zum digitalen Verzögern, Rechnen, Zählen und Datenverarbeiten benutzt werden (siehe später).

Ein mögliches Impulsdiagramm (S ist das Signal am 1S-Eingang des Slave-FFs und wird hier nur als Zwischenresultat angegeben)



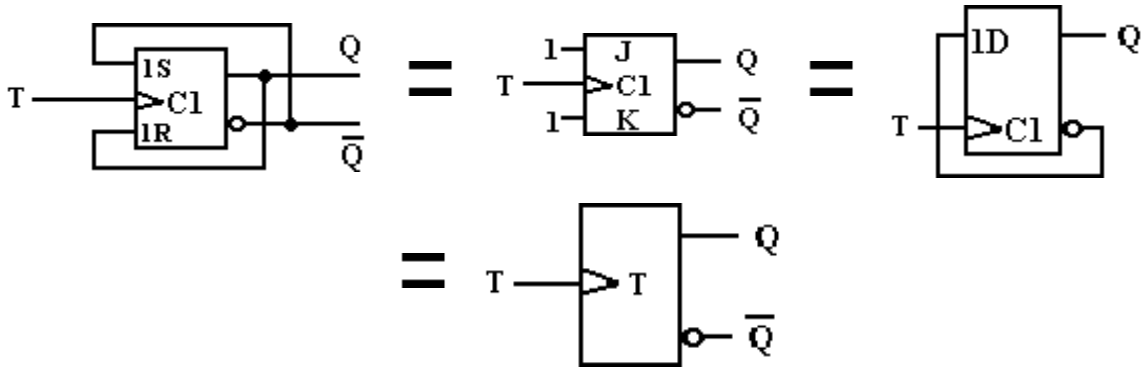
Uebung:

Wie sieht das Impulsdiagramm eines flankengesteuerten D-FFs aus, wenn man den invertierten Ausgang $\overline{Q_n}$ auf den D-Eingang zurückführt?

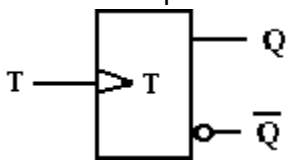
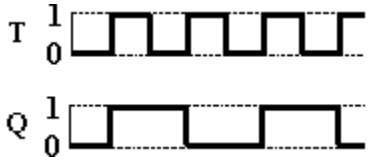
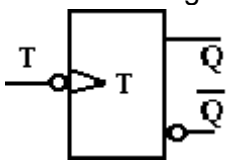
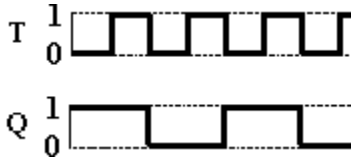


13.13.7 Zählerschaltungen (asynchrone Zähler)

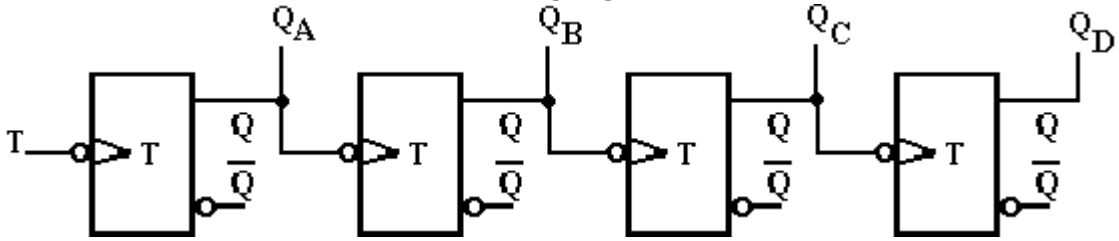
Aus 13.13.5 und 13.13.6 können wir auf mehrere Arten ein **T**oggle-Flipflop gemacht.
Man sagt ihm auch **T**rigger-Flipflop (trigger=auslösen) oder kurz **T-Flipflop.**



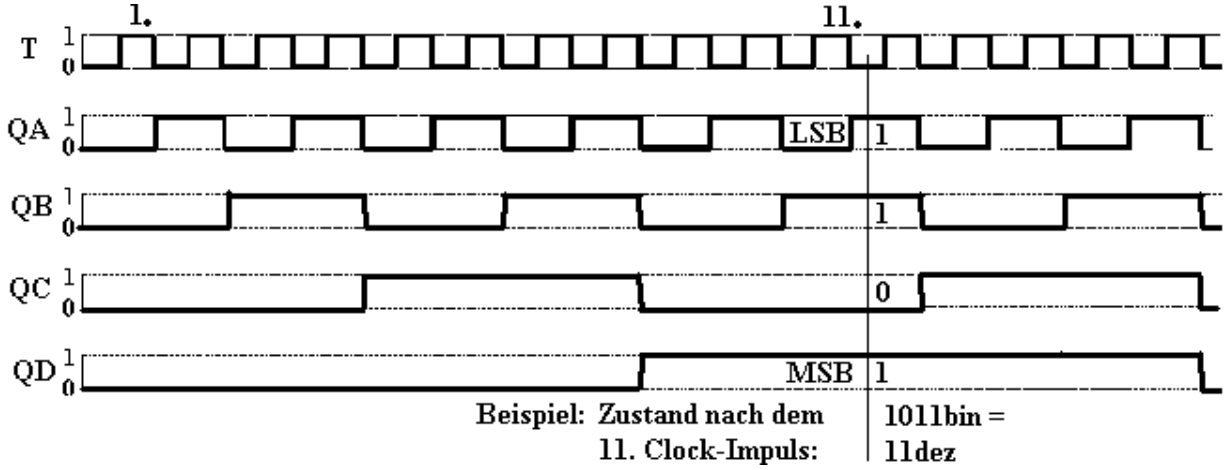
Mögliches Verhalten:

Schaltet bei positiver Flanke 	
Schaltet bei negativer Flanke 	

Einen Binärzähler erhält man durch simples Hintereinanderschalten von T-Flipflops.
Wenn man einen **Vorwärtszähler** will, muss man dazu die **negativ getriggerten** Flipflops nehmen und die **nichtinvertierten Q-Ausgänge** verwenden:



Mit 4 T-Flipflops erhält man einen vierstufigen Zähler und damit einen Vier-Bit-Zähler.
Er kann über 4 Stufen, also 2^4=16 Schritte zählen, nämlich von 0-15 (0000 - 1111).

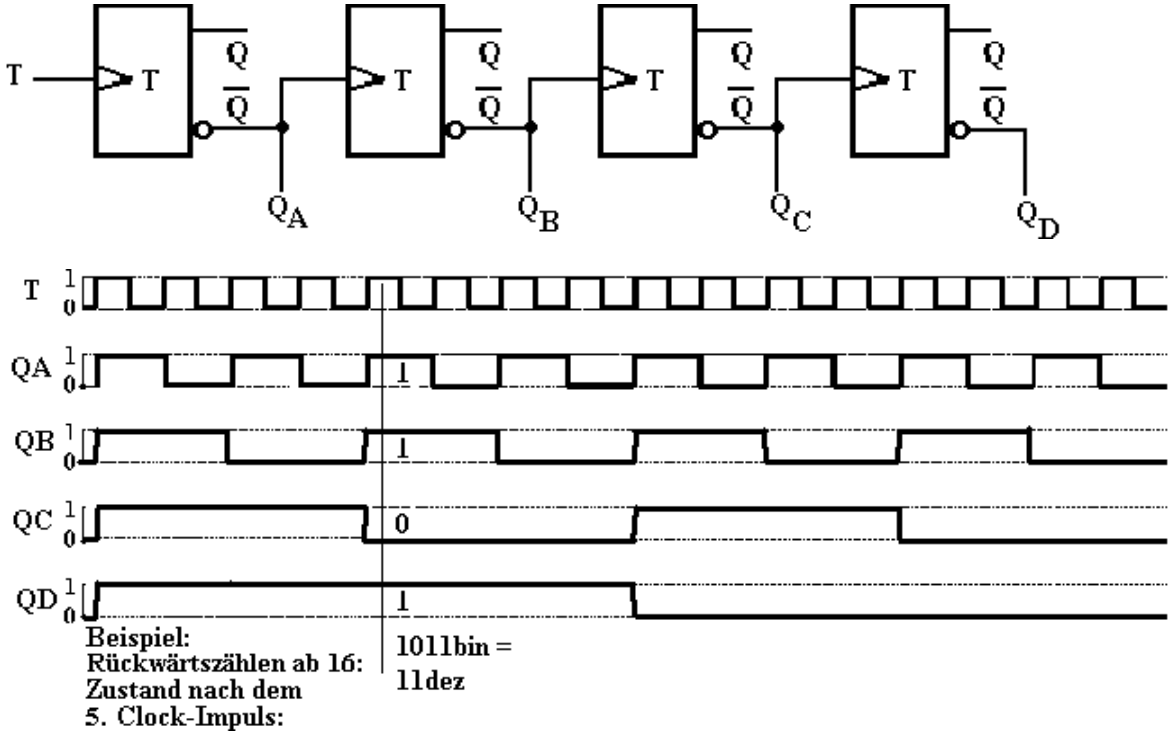


Anwendungsbeispiel: Auf den ersten Takteingang könnte zum Beispiel eine Lichtschranke wirken. Immer, wenn ein Objekt in die Lichtschranke tritt, entsteht eine negative Flanke, der

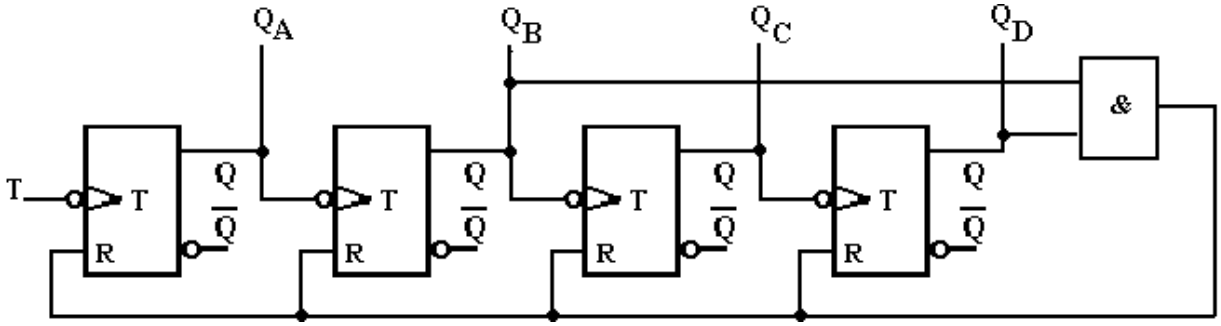
Zähler zählt eins höher. Die vier Ausgänge könnte man auf einen 4-Bit-zu-7-Segment-Decoder leiten, der das binäre Zählergebnis als Dezimal bzw. Hex-Ziffer anzeigen könnte.

Weitere Zählerformen:

- **Rückwärtszähler**
  (Hier muss man nun **positiv** getriggerte FFs und die **invertierten** Ausgänge nehmen)



- **BCD-Zähler (zählt von 0 bis 9 für Dezimalziffern)**
  Man muss einen 10er-Zähler so bauen, dass er auf 0 zurückgesetzt wird, wenn er auf den Dezimalwert 10 schalten möchte. Dazu muss man T-Flipflops mit einem **R**eset-Eingang haben. Die Reset-Eingänge müssen alle dann gleichzeitig aktiviert werden, wenn der Zähler auf den Binärzustand 1010 (=10 dezimal) wechseln will. Also nimmt man das Bit 1 (Wert=2) und das Bit 3 (Wert=8). Für diesen Vorwärtszähler muss man wieder **negativ** getriggerte FFs und die **nicht-invertierten** Ausgänge nehmen)



BCD-Zähler werden häufig verwendet, zum Beispiel in allen elektronischen Zählwerken oder auch in Digitaluhren zum Darstellen der Einerstelle von Stunden, Minuten oder Sekunden. Für die Zehnerstelle einer Sekundenanzeige muss dann aber ein 0-bis-5-Zähler realisiert werden.

- **Frequenzteiler:**

Alle Zähler sind gleichzeitig als Frequenzteiler verwendbar!
Mit einem Frequenzteiler kann man die Frequenz eines Eingangsssignals um beliebige Faktoren verkleinern.
Binärzähler funktionieren je nach Stufenzahl als **Teiler durch 2, 4, 8, 16, usw**. Durch **Zusatzschaltungen** analog wie beim BCD-Zähler lässt sich **jedes beliebige Teilverhältnis** herstellen!

All die vorbesprochenen Zähler sind so genannte **Asynchronzähler**. Das heisst, sie laufen **zeitlich nicht gleich**, sondern **nacheinander**, da jede Stufe erst die nächste ansteuert.

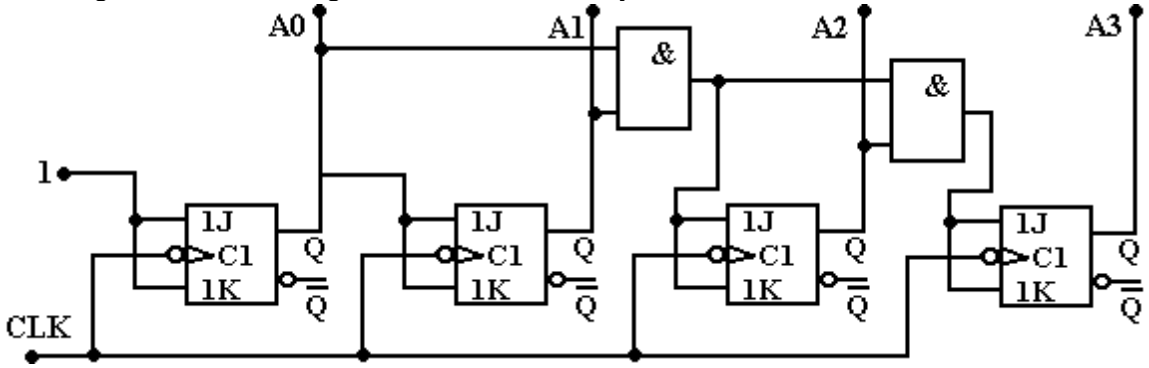
13.13.8 Synchronzähler

Für viele Fälle genügen die vorher besprochenen asynchronen Zähler, bei denen jede Zählstufe (jedes Zähl-Flipflop) das nächste steuert.
Bei sehr schnellen Zählvorgängen fallen die Signallaufzeiten und die Gatterreaktionszeiten ins Gewicht und können zu falschen Zwischenzählerständen führen.

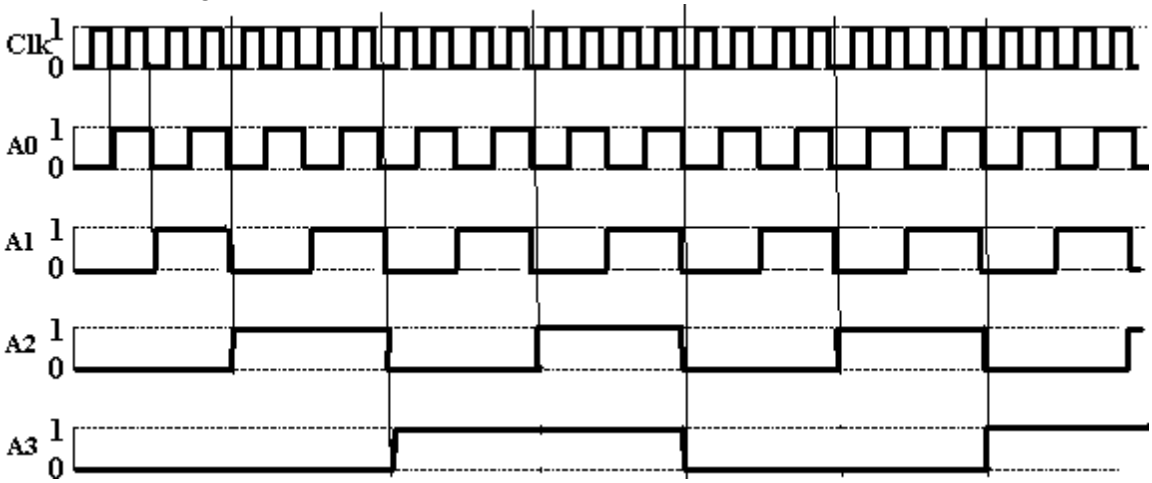
Deshalb bieten sich Synchronzähler an. Bei Synchronzählern werden alle Zählflipflops gleichzeitig (synchron) über ein Taktsignal geschaltet. Somit schaltet jedes beteiligte Bit eines mehrstufigen Zählers zur exakt gleichen Zeit.
Einen Synchron-Zähler erkennt man daran, dass alle Stufen über einen gemeinsamen Takteingang miteinander verbunden sind (ähnlich wie zuvor der gemeinsame Reset-Eingang).

Deshalb muss man hier ein Flipflop nehmen, das einen Takteingang hat und auch die Toggle-Funktion (Hin- und Herkippen) hat. Dies ist das **JK-Flipflop**.
Auch hier will man einen **Vorwärtszähler**, also muss man dazu die **negativ getriggerte** Flipflops nehmen und die **nichtinvertierten Q-Ausgänge** verwenden:

Das folgende Schema ergänzen wir zu einem synchronen Vorwärtszähler:



Beschreibung
(Das Impulsdiagramm des Vorwärtszählers liefert Hinweise zur Funktion 13.13.7)



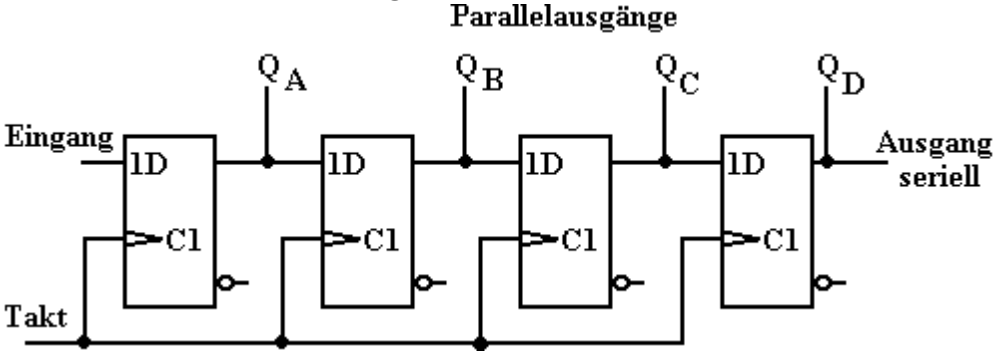
- Die nötige Kippfunktion der Flipflops ist nur aktiv, wenn J und K auf logisch 1 sind.
- J und K des ersten FFs werden somit auf 1 gelegt. Mit jeder negativen CLK-Flanke kippt es. Es liefert den Ausgang A0, das niederwertigste Bit.
- Das zweite FF darf gemäss Impulsdiagramm erst kippen, wenn der Ausgang Q des ersten FFs (A0) auf 1 ist.
  Also werden J und K des zweiten FFs mit dem Ausgang Q des ersten (A0) verbunden.
- Das dritte FF (A2) darf erst kippen, wenn A0 **und** A1 auf 1 sind, deshalb das UND-Gatter
- Das letzte FF (A3) darf erst reagieren, wenn A0 **und** A1 **und** A2 auf 1 sind, deshalb ein UND-Gatter mit drei Eingängen bzw. die zwei gestaffelten UND-Gatter.

13.13.9 Schieberegister

Der Begriff Register kann mit dem Begriff Speicher gleichgesetzt werden.

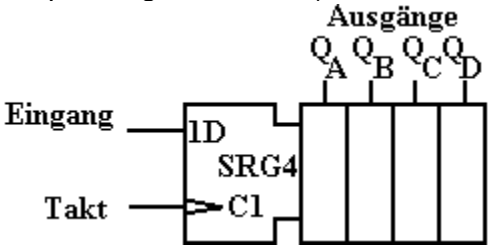
Ein Schieberegister (englisch: shift register) ist eine Schaltung, welche z.B. ein Byte taktgesteuert und bitweise aufnehmen, speichern und bitweise wieder abgeben kann.

Beispiel: Ein **4-bit-Schieberegister** aus D-Flipflops:

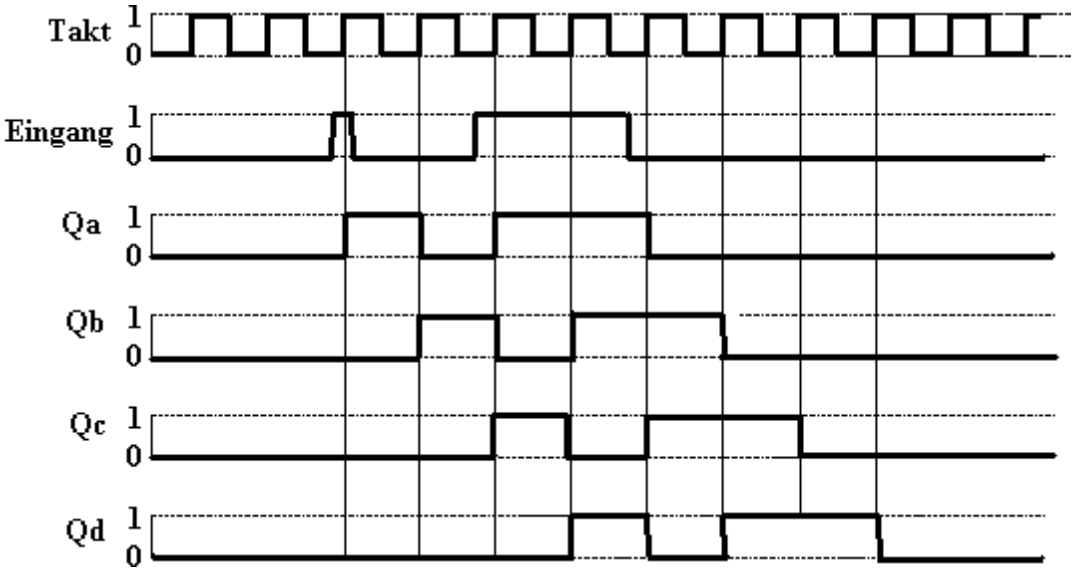


Funktion: Das Eingangssignal wird mit der ersten Taktflanke vom ersten D-FF übernommen und erscheint an dessen Ausgang QA. Mit der nächsten positiven Flanke wird es an den Ausgang QB weitergeleitet und so weiter (siehe Impulsdiagramm unten).

Das vereinfachte Symbol setzt sich aus dem so genannten Steuerkopf links (D-Flipflop-Charakteristik) und den Registerplätzen rechts mit den Ausgängen zusammen.



Schieberegister sind FIFO-Register. Das bedeutet **F**irst **I**n, **F**irst **O**ut. also: dasjenige Element, das zuerst hineingespeichert wurde, "fällt" auch als erstes wieder hinaus. Dies ist im Impulsdiagramm ersichtlich.



Im dargestellten Bild können die Daten am Ausgang seriell oder parallel abgenommen werden. Es gibt auch Schieberegister mit Parallel**ein**gabe, so der Ausgang dann seriell erfolgt. Daraus abgeleitet:

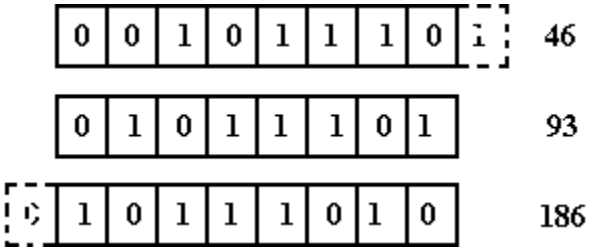
Wichtige Anwendung:
Seriell-parallel-Wandler und Parallel-seriell-Wandler.

Ebenso kann man Schieberegister im Prinzip mit jedem Flipflop-Typ aufbauen. Mit komplizierteren Verknüpfungen lassen sich auch Schieberegister mit umschaltbarer Schieberichtung bauen.
Ringregister sind Schieberegister, bei welchen der letzte serielle Ausgang wieder mit dem Eingang verbunden ist. Damit lässt sich z.B. eine Steuerung für ein Lauflicht mit "programmierbarem" Lichtmuster aufbauen.
Für Schieberegister gibt es weitere Anwendungen. Eine wichtige Bedeutung hat sie in der digitalen Arithmetik:
Für viele Prozessoren und Mikrocontroller gibt es die Befehle **Shift left** und **Shift right.**

- **Shift Left** bedeutet nach links schieben, ein binärer Wert wird **mit 2 multipliziert**
- **Shift Right** bedeutet nach rechts schieben, das Ergebnis ist die **ganzzahlige Division durch 2 (DIV 2)**

Beispiel: Die Dezimalzahl 93 liegt in einem **8-Bit**-Schieberegister gespeichert.

- Durch Linksschieben entsteht das Doppelte (=186)
- Durch Rechtsschieben entsteht der ganzzahlige Anteil der Hälfte (46 statt 46,5)



Wenn man ein Register hat, das nicht zum Schieben vorgesehen ist, so handelt es sich um ein reines
Speicherregister.
Das Weiterschieben kann z.B. verhindert werden, indem man die beteiligten Flipflops mit den entsprechenden Eingängen (z.B. bei JK-Flipflops) in den Zustand **"Speicherfall"** versetzt.

13.14. Speicher und ihre Organisation

Allgemeines

Jedes Flipflop, das wir kennengelernt haben, ist als Speicherzelle für 1 Bit einsetzbar.

Solche Flipflops können im Prinzip auch aus zwei Transistoren gebaut werden (Siehe Kap. 11.6.3, Transistoren, bistabile Kippstufe).

Bekanntlich sind Datenspeicher für viele Bits ausgelegt.
Damit man möglichst wenige Leitungen benötigt, werden die Speicherzellen zu mindestens zweidimensionalen Speichermatrizen zusammengefasst. Darin gibt es Zeilen und Spalten. Beim Speichern und Lesen von Daten müssen diese Speicherzellen verwaltet werden. Man muss ihnen "sagen", wann sie "dran" sind, also muss man sie adressieren.

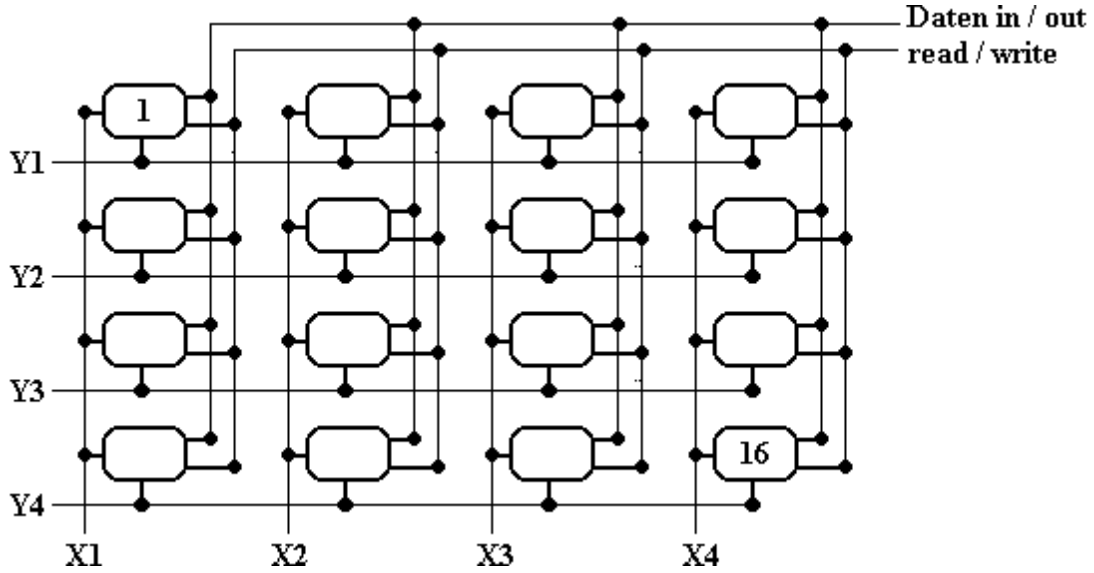
Für jedes Bit wären dann eigentlich vier Leitungen nötig:

- Eine Leitung als Dateneingang (speichern)
- Eine Leitung als Datenausgang (lesen)
- Eine Leitung zum Auswählen der Zeile der Speicherzelle
- Eine Leitung zum Auswählen des Spalte der Speicherzelle

Man kann nun die Daten**ein**gänge aller Speicherzellen sowie die Daten**aus**gänge aller Speicherzellen zusammenschalten.
Damit ein zu speicherndes Bit nur in **einer** Zelle abgelegt wird, muss die entsprechende Zelle angesprochen, adressiert werden. Das Analoge gilt beim Lesevorgang.

Bei idealer Organisation des Speichers sind für einen 16-Bit-Speicher vorerst 10 Leitungen erforderlich: (4 Zeilen + 4 Spalten + 1 Dateneingang + 1 Datenausgang).

Ein sogenannter **16x1-Speicher** kann also mit 10 Leitungen direkt verwaltet werden:

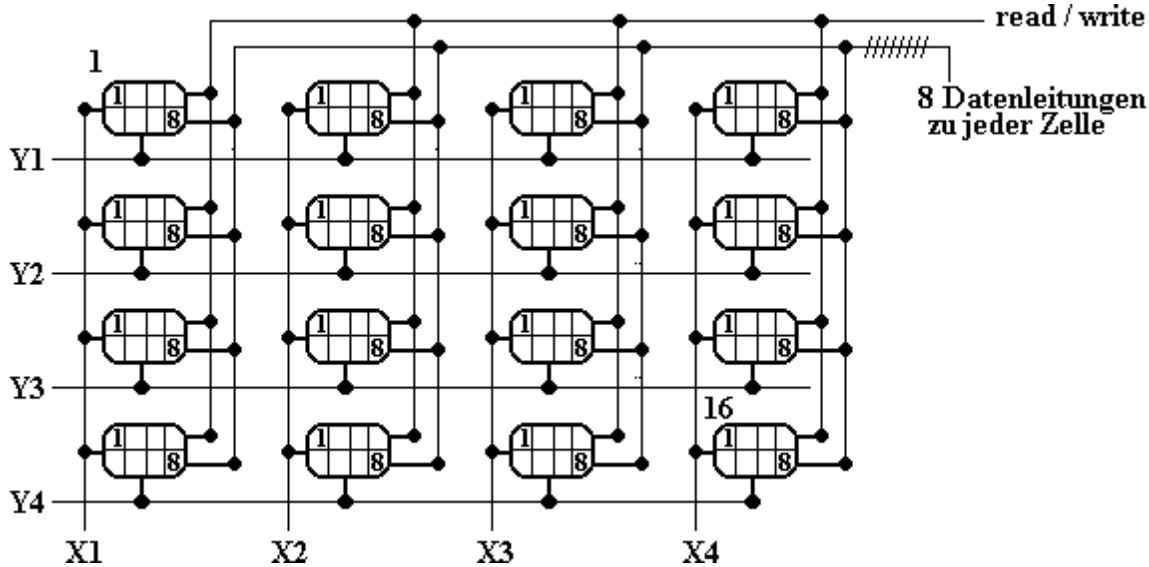


Kenngrössen und Funktion dieses Speichers:

- Gesamtkapazität: 16 Bit
- 1 Bit pro Speicherzelle
- Mit den Koordinatenleitungen X und Y wird die Speicherzelle adressiert.
- Mit der Leitung "Daten in / out" werden die Daten gespeichert oder ausgelesen.
- Mit der Leitung "read/write" wird festgelegt, ob es sich um einen Lese- oder Schreibzugriff handelt.

Nun speichern die meisten Speicher aber nicht einzelne Bits, sondern zumindest 8-Bit-Worte, also Bytes. Man will also jeweils alle 8 Bits gleichzeitig schreiben oder lesen. Damit kann man Speicherzellen machen, die 8 Plätze aufweisen. Die Adressierung wird dabei nicht komplizierter, da immer noch die ganze Zelle angesprochen wird, lediglich die Datenleitungen werden natürlich verachtfacht:

Ein **16x8-Bit-Speicher** kann mit 17 Leitungen direkt verwaltet werden:



Kenngrössen dieses Speichers:

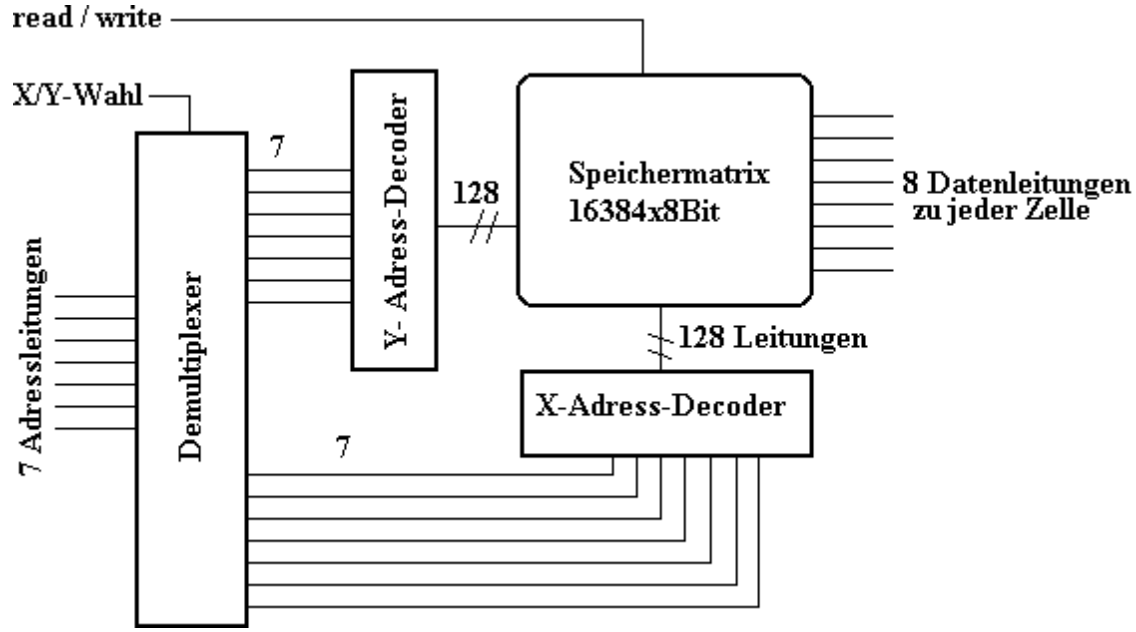
- Gesamtkapazität: 16x8 Bit = 16 Bytes
- 8 Bits pro Speicherzelle

Die 8 Koordinatenleitungen können sogar noch aus drei Leitungen demultiplext werden (2^3=8). Diese Vereinfachung ist vor allem bei höheren Speichern sinnvoll, damit die Speicherbausteine möglichst wenige Anschlüsse haben und damit auch klein werden.
In den obigen beiden Beispielen reduziert sich dadurch der Verdrahtungsaufwand auf 5 Leitungen für den 16x1-Bit-Speicher und auf 12 für den 16x8-Bit-Speicher.

Nun immer etwas aufwändiger: Der Aufbau eines **16Kx8-Speichers (=16KB)**:

Kenngrössen und Organisation eines solchen Speichers:

- Gesamtkapazität: 2^14x8 Bit = 16384 x 8Bit = 16KBytes
- 8 Bits pro Speicherzelle, also 8 Datenleitungen
- Zellenanordnung in 2^7=128 Zeilen x 2^7=128 Spalten, demultiplexierbar aus je 7 Adressleitungen
- Speicher verwaltbar mit 17 Leitungen



Nach diesem Schema lassen sich beliebig grosse Speicher aufbauen.
Mit aufwändiger Bautechnik lässt sich die Anzahl Anschlüsse tief halten.
So könnte man z.B. auch dreidimensionale Matrizen bauen, was die Anzahl Adressleitungen weiter reduziert.
Ferner könnte man die 9 Steuerleitungen links wiederum multiplexen zu 4 Leitungen, die Datenleitungen liessen sich auf 3 multiplexen. Es bräuchte aber ein paar Steuerleitungen mehr. Total ergäben sich vielleicht 10 Leitungen für die Verwaltung von 8x16384=131072 Bits.Allerdings müssen dann die Adressbefehle zeitlich gestaffelt eintreffen, was die Verarbeitungsgeschwindigkeit harabsetzt.

All dies gilt natürlich für alle Speicher mit freiem Schreib- und Lesezugriff.

Wie Sie wissen, gehören dazu primär die **RAM**s.

13.14.1 RAM

RAM heisst **R**andom **A**ccess **M**emory, das sind Schreib- und Lesespeicher mit freiem Zugriff, das heisst direktem Zugriff auf jede beliebige Speicherzelle.
RAMs gelten grundsätzlich als **flüchtige Speicher**, das heisst: Ohne Spannungsversorgung geht der Inhalt verloren.

Statische RAM

Bei statischen RAMs wird jedes Bit als Zustand eines Flipflops (bistabile Kippstufe) gespeichert.
Dabei werden normale Transistoren oder MOS-FET-Transistoren eingesetzt.
Ist die Information einmal gespeichert, bleibt sie ohne weitere Speicherzugriffe gespeichert, solange die Versorgungsspannung vorhanden ist.

Vorteile:

- Rasche Zugriffszeit.
- Für kleinere Systeme sind sie ausgezeichnet geeignet, da keine Refresh-Elektronik (wie bei dynamischen RAMs) nötig ist.
- Ausserdem lässt sich der Speicherinhalt mit Batterien oder Kondensatoren puffern, also ohne Netzspannung erhalten.

Nachteile:

- Relativ hoher Platzbedarf pro Speicherzelle auf dem Chip, deshalb für hochintegrierte Grossspeicher weniger geeignet.

Dynamische RAM

In dynamischen RAMs wird jedes Bit als Ladung eines Kondesators gespeichert. Da die sehr kleinen Kapazitäten einer raschen Selbstentladung unterliegen, müssen gespeicherte Einsen ständig wieder neu gespeichert werden, der Kondensator wird also nachgeladen.

Vorteile:

- Hohe Integrationsdichte, wenig Platzbedarf.
- Für grössere Systeme geeignet, wo der Aufwand des Refreshcontrollers keine Rolle spielt.

Nachteile:

- Speicher muss ständig aufgefrischt werden. Dies erfordert eine Refresh-Steuerung innerhalb oder ausserhalb des eigentlichen Speichers.
- Während des Refresh-Vorgangs ist der Speicher für Zugriffe z.B. durch die CPU gesperrt, dies bedeutet eine Reduktion der Zugriffsgeschwindigkeit.
- Pufferung mit Batterien ist kaum sinnvoll, da auch im StandBy-Betrieb ständig der Refreshcontroller laufen muss.

13.14.2 ROM

ROMs sind **R**ead **O**nly **M**emories. Sie können nur Daten ausgeben. Ausserdem sind es **nichtflüchtige Speicher.** Die Daten sind fix und unveränderlich in jeder Speicherzelle programmiert. Sie werden bereits bei der Herstellung der Halbleitermasken festgelegt und müssen praktisch auf ewige Zeiten auch ohne Spannungsversorgung erhalten bleiben. ROMs werden nur eingesetzt, wenn keine Programm- oder Datenänderungen in einem Mikrocomputersystem mehr zu erwarten sind. Beispielsweise könnten die Daten für einen Kalender in einem ROM zur Verfügung stehen.

Auch ein Buch oder eine CD-ROM sind ROMs.

13.14.3 PROM

PROMs sind **P**rogrammable **R**ead **O**nly **M**emories. Auch sie können nur Daten ausgeben. Der Hersteller des Chips liefert leere Speicherzellen, die alle 0 oder alle 1 enthalten. Die Daten sind einmalig durch den Anwender (z.B. ein PC-Produzent) einprogrammierbar. Dies geschieht mit definierten Spannungsstössen auf die entsprechend adressierten Speicherzellen, womit bestimmte Einsen zu Nullen werden oder Nullen zu Einsen werden. Das Programmieren geht sehr schnell, da die zu programmierenden Zellen zerstört werden. Dieser Vorgang ist aus diesem Grund irreversibel! Bei einem Programmierfehler muss das Bauteil entsorgt werden.
In PROMs können Programme z.B. für Steuerungen gespeichert werden, und jeder einzelne Baustein kann z.B. eine individuelle Serienummer enthalten. In PCs war lange Zeit das BIOS in einem PROM gespeichert. Häufig werden PROMs in Stecksockeln in die Hardware eingebaut, so dass sie später durch neuere Versionen ersetzt werden können.

Eine einmal beschreibbare CD-ROM ist auch ein PROM.

13.14.4 EPROM

EPROMs sind **E**rasable **P**rogrammable **R**ead **O**nly **M**emories. Auch sie sind im Prinzip dazu bestimmt, im Betrieb nur Daten auszugeben und sie auf ewige Zeiten auch ohne Spannungsversorgung zu halten. EPROMs sind "erst" seit 1975 auf dem Markt. Die Programmierung geht länger als beim PROM, da die Zellen nicht zerstört werden und dennoch die Information sicher und dauerhaft einprogrammiert werden muss. Dazu muss eine spezielle Halbleiterschicht (das floating gate) aktiviert werden. Das programmierte Bitmuster kann jedoch gelöscht und durch ein neues ersetzt werden.
Es kann aber nicht wie beim RAM durch einen adressierten Schreibzugriff gelöscht werden. Zum Löschen wird UV-Licht benötigt, das durch ein Quarzglasfenster im Chipgehäuse eindringen kann. Die Bestrahlung muss zur sicheren Löschung etliche Minuten dauern. Alle Speicherzellen werden gemeinsam gelöscht, eine Adressierung ist nicht nötig. EPROMs im Einsatz sind mit einem Kleber überklebt, der das Fenster abdeckt und kein Licht eindringen lässt.

13.14.5 EEPROM

EEPROMs sind **E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emories. Also elektrisch löschbare programmierbare ROMs. Wie der Name schon sagt, sind sie nicht durch Licht, sondern durch Strom löschbar. Dies hat den Vorteil, dass man bei einer Programmrevision nicht während langer Zeit die gesamte Information löschen muss, sondern gezielt nur die fehlerhaften Bits oder Bytes ersetzen kann. Ein EEPROM ist eigentlich schon fast wie ein RAM, und zwar ein **nichtflüchtiges RAM**, ein **NOVRAM** (**no**n **v**olatile **RAM**) , das seine Information auch ohne Spannungsversorgung halten kann. EEPROMS werden heute in allen Mikrocontrollersystemen als bei Bedarf veränderbare Programmspeicher verwendet.

Eine wiederbeschreibbare CD-RW ist eigentlich auch ein EEPROM, ausser dass es optisch und nicht elektrisch beschrieben und gelesen wird.

13.14.6 Flash-EPROM

Flash EPROMs sind ein Zwischending zwischen EPROM und EEPROM. Sie haben die einfachere Speicherstruktur der EPROMs, ihre Zellen sind auch hier nur alle gleichzeitig löschbar. Dafür aber elektrisch wie beim EEPROM und innerhalb von wenigen Millisekunden. Die meisten heutigen BIOS im PC sind Flash-EPROMs.

13.15 Analog-Digital-Wandler und Digital-Analog-Wandler

Bereits ganz am Anfang des Kapitels Digitaltechnik (13.1.) haben wir die Art und "Herstellung" von digitalen Signalen besprochen.

Bekanntlich besteht unsere Welt aus kontinuierlichen Grössen. Wenn wir diese analogen Werte mit digitalen Maschinen verarbeiten wollen, müssen sie digitalisiert werden. Ferner ist es nötig, dass digitale Werte oder Signale wieder in analoge gewandelt werden, weil die menschlichen und übrigen natürlichen "Schnittstellen", z.B. unsere Sinne, nach wie vor analog sind.
Deshalb brauchen wir massenhaft elektronische Schaltungen, welche analoge in digitale Signale wandeln können und umgekehrt. Diese heissen

- **Digital-Analog-Wandler (D-A-Converter, DAC)**
- **Analog-Digital-Wandler (A-D-Converter, ADC)**

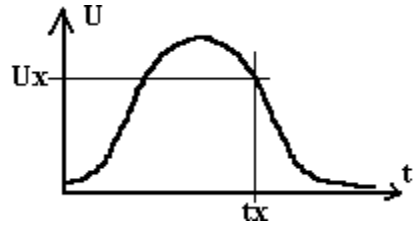
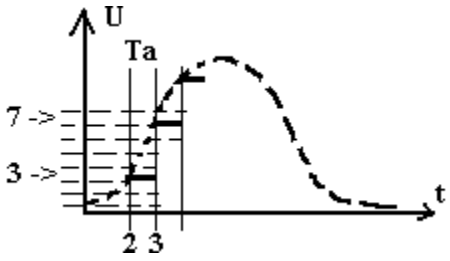
Einsatzgebiete von ADC und DAC:

Unterhaltungselektronik (Audio, Video), Uebertragungstechnik, Telekommunikation, Messtechnik, Steuer- und Regelungstechnik usw.

Beispiele:

Digital-Analog-Wandler (DA-Converter)	**Analog-Digital-Wandler (AD-Converter)**
Soundkarte Ausgang	Soundkarte Eingang
CD-, Minidisc- oder MP3-Player	Digitalmessgeräte
ISDN-Leitung zu Telefonhörer	Telefonmikrofon zu ISDN-Leitung

Zur Erinnerung:

	**Analoges** Signal: Das Signal ist **zeit- und wertekontinuierlich.**
	**Digitales** Signal: Nach **Abtastung und Quantisierung** ist das Signal **zeit- und wertediskret**, das heisst, es kommen nur definierte Werte vor.

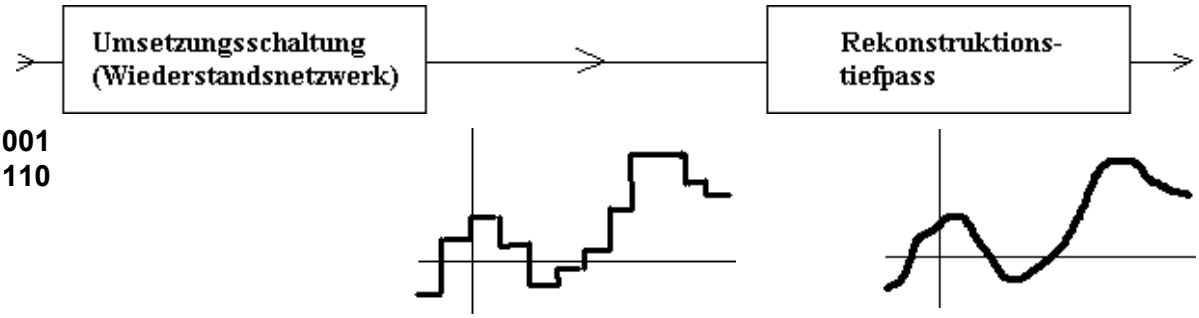
Die beiden wichtigen Kernbegriffe:

Abtastzeit bzw. Abtastfrequenz (sample rate)	Zeitliche Angabe, wie häufig ein analoger Wert in einen digitalen Wert umgewandelt wird. Das Abtasttheorem fordert, dass die Abtastfrequenz mind. doppelt so gross sein muss wie die abzutastende Frequenz. $f_a \geqq 2 \cdot f_{max}$
Auflösung (resolution)	Anzahl der möglichen digitalen Stufen, zu denen ein Analogwert zugeordnet werden kann. (bei der Quantisierung)

13.15.1 D-A-Wandler (Digital-Analog-Wandler)

Voraussetzung für eine D-A-Wandlung ist, dass das digitale Signal in einem bewertbaren Code vorliegt. Der Wandler hat die Aufgabe, die Bewertung der einzelnen Bits vorzunehmen.

Prinzip eines D-A-Wandlers:



binär codierte Signalwerte

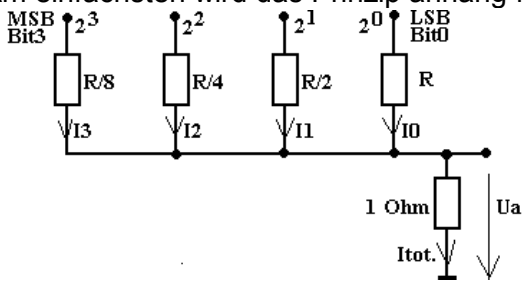
Treppenförmiges Signal

Analogsignal

Ein digitales Signal liegt bekanntlich wertediskret vor. Es gibt nur gewisse Signalwerte. Deshalb ist das erste Ergebnis der Wandlung ein treppenförmiges Signal

Mögliche Schaltungen zur D-A-Wandlung

Am einfachsten wird das Prinzip anhang folgender Schaltung klar:



Es ist ein 4-Bit-Wandler, der Leitwert der oberen Widerstände entspricht der Gewichtung des entsprechenden Bits.
Eine logische 0 entspreche 0 Volt, eine logische 1 entspreche 5 Volt.
Die Widerstände haben die Werte 625, 1250, 2500 und 5000 Ohm.
Gemäss der Knotenregel addieren sich die Ströme und fliessen durch den 1-Ohm-Widerstand. Dort erzeugen sie den Spannungsabfall, der wertmässig dem summierten Strom und damit dem digital anliegenden Wert entspricht.

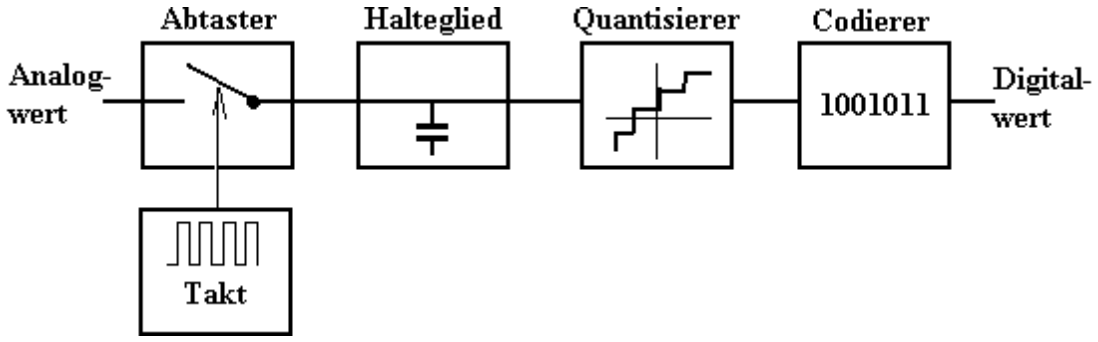
Dies zeigt sich in der Tabelle:

Zahl	Bit3	Bit2	Bit1	Bit0	I3	I2	I1	I0	Itot.	Ua

13.15.2 A-D-Wandler (Analog-Digital-Wandler)

Wie ganz am Anfang des Kapitels aufgeführt, umfasst die A-D-Wandlung 3 Schritte:
1. Abtastung 2. Quantisierung 3. Codierung

Prinzip eines A-D-Wandlers



Der Abtaster nimmt zu exakt bestimmten Zeitpunkten eine Probe (sample) des Analogsignals.

Das Halteglied ist ein analoger Zwischenspeicher, der das zuvor abgetastete Signal für die Dauer der Quantisierung speichern muss. Abtast-/Halte-Schaltungen sind unter dem englischen Begriff **sample/hold** bekannt.

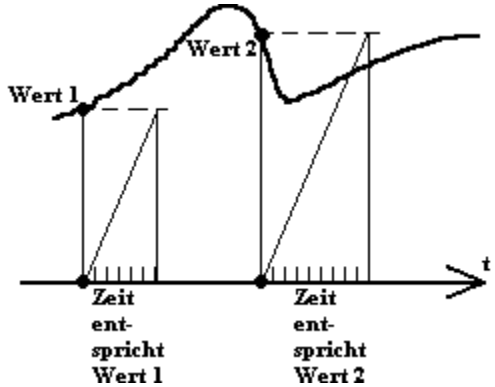
Der Quantisierungsvorgang braucht mehr oder weniger Zeit. Er kann erst nach Abschluss der Abtastung (**sample**) beginnen (in der **hold**-Phase). Für die Quantisierung sind verschiedenste Verfahren bekannt:

Mögliche Schaltungen zur A-D-Wandlung (Quantisierung):

Es gibt verschiedenste Verfahren zur Wandlung von Analog- in Digitalsignale, die alle Vor- und Nachteile haben. Kriterien sind zum Beispiel: Schnelligkeit, Linearität, Reproduzierbarkeit. Hier sind nur einige Verfahren kurz beschrieben.

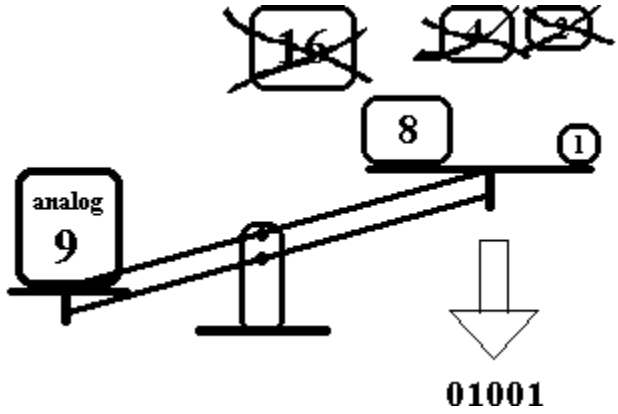
Zeitprinzip (Gebräuchlich sind Rampen- und Doppelrampenverfahren):

Das abzutastende Amplitudenwert des Analogsignals wird mit der Zeit verglichen, die eine linear ansteigende Vergleichsspannung braucht, bis sie den Wert des Analogsignals erreicht. Während dieser Zeit zählt ein digitaler Zähler die Taktimpulse. Diese Impulszahl ist dann der Digitalwert und der ist proportional zum Analogwert. Die Wandlung muss stets innerhalb eines Abtastzyklus' (von einem Sample zum andern) abgeschlossen werden können!
Die Logik benötigt zudem einen D-A-Wandler, der das digital erhaltene Signal wieder zurückwandelt, damit es mit dem gesampelten Signal verglichen werden kann.
Vorteil: einfache Schaltung, relativ schnell. Nachteil: Je nach Bauteiletoleranz nicht sehr genau.



Iterationsvverfahren (Wägeverfahren)

Der Digitalwert wird wie mit einer alten Waage ermittelt:
Auf der einen Seite "liegt" der Analogwert, eine Kontrolllogik "legt" auf die andere Seite **binäre Gewichtssteine** mit der Wertigkeit der Bits, beginnend mit dem Grössten.
Jene Steine oder Bits, die gesetzt werden können, ohne dass die Waage kippt, ergeben im Digitalwert eine 1.



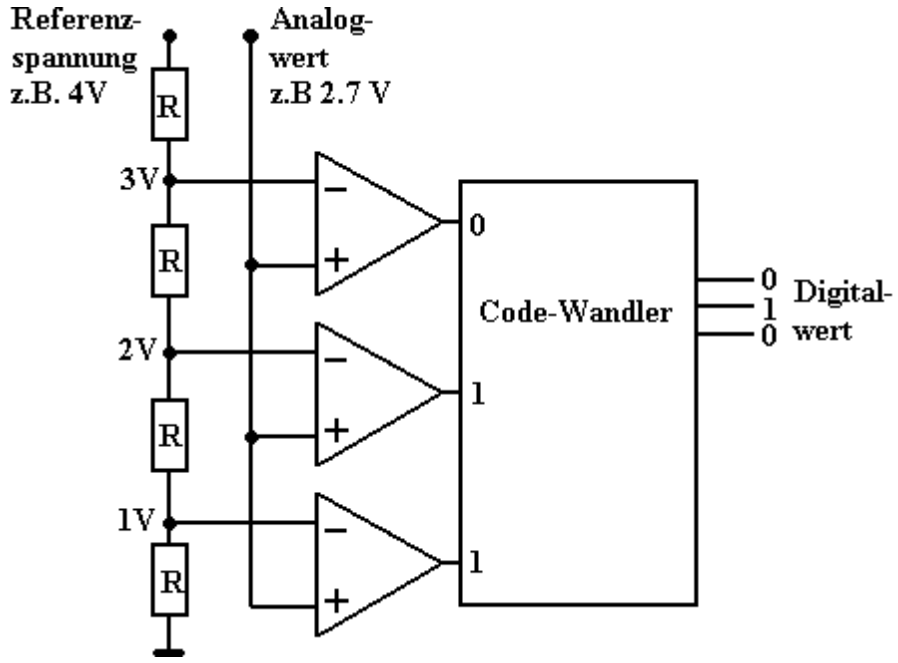
Zum Vergleich des ermittelten Ergebnisses mit dem Eingangssignal benötigt die Logik auch hier einen D-A-Wandler.

Direktumsetzung (mit Flash-Wandler)

Dies ist im Prinzip die direkteste und schnellste Art der A-D-Wandlung, aber mit hohem Schaltungsaufwand verbunden. Für jede gewünschte Stufe der Quantisierung benötigt man einen Komparator. Für eine 16 Bit Auflösung benötigt man also 65536 Komparatoren.

Vorteil: sehr schnell. Nachteil: Grosser Schaltungsaufwand.

Aufbau:



Ende Digitaltechnik