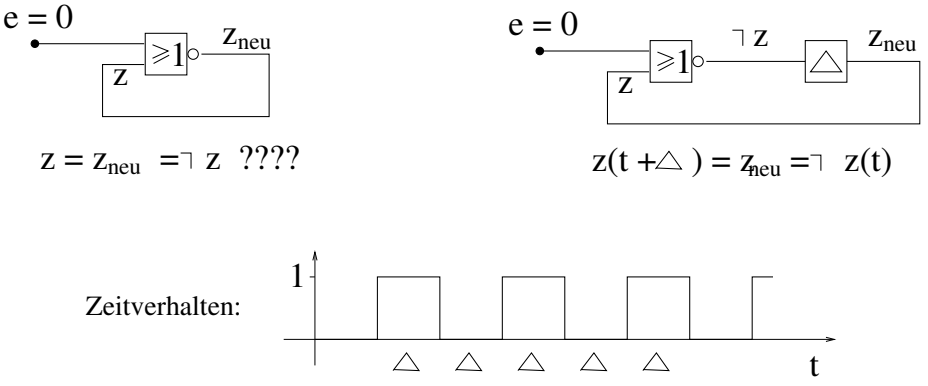
Schaltwerke

Schaltnetze und Schaltwerke

Schaltnetze dienen zur Beschreibung dessen, was innerhalb eines Prozessortakts abläuft. Die Taktzeit des Prozessors muss immer etwas größer sein als die Signallaufzeit des Schaltnetzes. Damit wird sichergestellt, dass am Ende des Takts an allen Ausgängen stabile Signale anliegen. Schaltnetze werden durch azyklische gerichtete Graphen beschrieben, d.h. darin sind keine gerichteten Kreise, im technischen Sinne also keine Rückkopplungen erlaubt.
Mit einem Schaltwerk wird das Verhalten eines Prozessors von einem Takt zum nächsten beschrieben. In Schaltwerken werden Rückkopplungen gezielt eingesetzt, um bestimmte Signale von einem Prozessortakt zum nächsten stabil zu halten. Bevor wir zu getakteten Schaltungen kommen, wollen wir uns allgemein mit den Effekten von Rückkopplungen beschäftigen.

NOR-Gatter mit Rückkopplung und asynchrone Flipflops

Die folgende Abbildung zeigt auf der linken Seite ein NOR-Gatter mit Rückkopplung. Setzt man für das Eingangssignal $e = 0$ voraus, dann ergibt sich ein scheinbar widersprüchliches Verhalten, denn das Sinal z_{neu} am Ausgang wäre die Negation des Eingangssignals z, aber andererseits müssen beide gleich sein. Eine andere Sicht ergibt sich, wenn man das Verhalten als einen zeitlichen Prozess betrachtet, bei dem die Schaltzeit des NOR-Gatters durch ein Verzögerungsglied Δ symbolisiert wird. Dann ergibt sich der Wert von z_{neu} erst nach einer zeitlichen Verzögerung als Negation von z. Idealisiert würde diese Schaltung also eine Schwingung erzeugen, die aber nicht von einem zentralen Zeittakt abhängt - man spricht deshalb von einem asynchronen Schaltwerk.

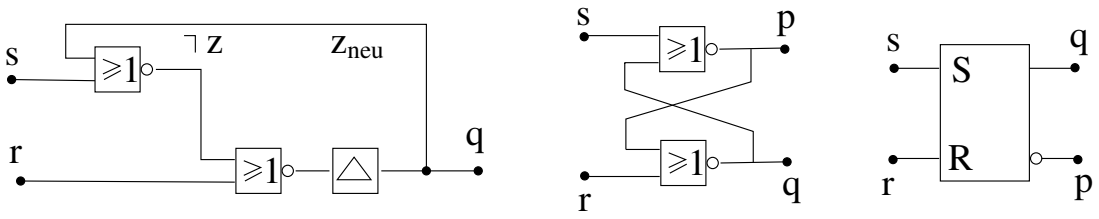


Man kann sich den oben beschriebenen Effekt zu Nutze machen, indem man die Identität als eine doppelte Negation realisiert und somit eine sich selbst stabilisierende Schaltung mit der folgenden Spezifikation aufbaut:

- Es gibt zwei Eingänge s (set) und r (reset) und einen Ausgang q
- Ein Impuls $s = 1$ setzt q (mit Verzögerung Δ) auf 1
- Ein Impuls $r = 1$ setzt q (mit Verzögerung Δ) zurück auf 0
- Die Situation $s = r = 1$ wird ausgeschlossen.

Die nachfolgende Abbildung zeigt auf der linken Seiten eine Schaltung zur Realisierung dieser Spezifikation. In der Mitte ist die gleiche Schaltung mit veränderter Platzierung der beiden NOR-Gatter und ohne das symbolische Verzögerungsglied abgebildet. Zusätzlich zum Ausgang q gibt es noch den Ausgang p, der die Negation von q liefert. Auf der rechten Seite wird ein neues Schaltsymbol für dieses Netz eingeführt. Man beachte, dass dabei die Anschlusspositionen der Ausgänge q und p vertauscht wurden.

Asynchrones RS–Flipflop



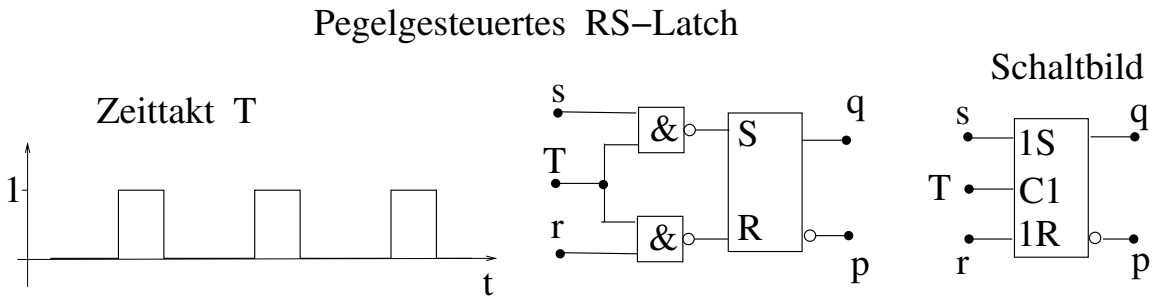
Impuls s=1 setzt Ausgang q auf 1
Impuls r=1 setzt Ausgang q auf 0

p = ¬ q

Schaltbild

Synchronisation

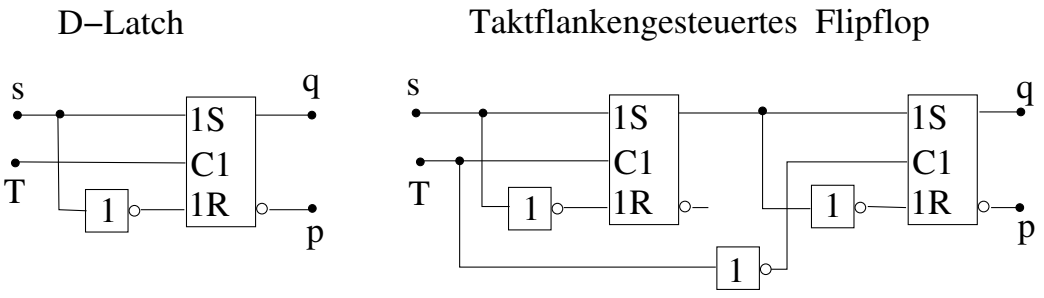
Das oben abgebidete Schaltwerk, ein sogenanntes RS-Flipflop, arbeitet wieder asynchron. Unser nächstes Ziel ist die Synchronisation eines solchen Flipflops mit einem zentralen Zeittakt T. Ein Gesamttakt besteht aus einen Abfragephase, in der T den Wert 1 hat, und einer Umschaltphase, in der T den Wert 0 hat. Ausgehend davon, dass in jeder Abfragephase alle Eingangswerte satbil sind, will man erreichen, dass sich die Ausgangswerte nur in der nachfolgenden Umschaltphase verändern, aber in der nächsten Abfragephase wieder stabil sind. Ein erster Schritt dazu ist die Konstruktion eines RS-Latches (Latch = Schnappschloss), mit dem verhindert wird, dass Signale, die in der Umschaltphase anfallen, eine Änderung des Ausgangssignals bewirken können. Dazu wird einfach das UND von s bzw. von r mit dem Taktsignal T gebildet (bei $T = 0$ keine Änderungen).



Diese Schaltung hat gemessen an unseren Anforderungen noch zwei Mängel:

1. Das Verhalten bei $s = t = 1$ ist unklar.
2. Das Ausgangssignal ist in der Umschaltphase stabil, es sollte aber in der Abfragephase stabil sein.

Das erste Problem kann schnell behoben werden: Man verzichtet auf den Eingang r und lenkt statt dessen die Negation von s auf den zweiten Eingang. Diese Schaltung nennt man ein D-Latch, wobei das D für delay steht. Die Bedeutung von s hat sich jetzt im Vergleich zu RS-Flipflops etwas verändert: Mit $s = 1$ wird der Ausgang q auf 1 gesetzt und mit $s = 0$ erfolgt die Rücksetzung auf 0.
Zur Lösung des zweiten Problems schaltet man zwei solche Bauteile hintereinander und steuert das zweite mit der Negation des Taktsignals an. Das zweite D-Latch erhält also seinen Eingabewert in der Umschaltphase und hält in der folgenden Abfragephase seinen Ausgabewert stabil. Diese Schaltung nennt man ein taktflankengesteuertes Flipflop (auch unter der Bezeichnung Master-Slave-Flipflop bekannt). Sie realisiert eine taktgesteuertes Verzögerungsglied (für ein Bit).



Modellierung von Schaltwerken mit endlichen Automaten

Jedes Schaltnetz mit k Einängen und l Ausgängen realisiert eine Boolesche Funktion $f : \mathbb{B}^k \longrightarrow \mathbb{B}^l$. Betrachtet man ein Schaltwerk, bei dem Rückkopplungen ausschließlich durch taktgesteuerte Verzögerungsglieder erfolgen, dann kann man

(gedanklich) die Verzögerungsglieder auch zu einer Einheit außerhalb der restlichen Schaltung zusammenfassen, wobei dieser Rest dann rückkopplungsfrei - also ein Schaltnetz - ist. Wenn das Schaltwerk n Eingänge, m Ausgänge und k Verzögerungsglieder hat, dann realisiert das Schaltnetz eine Boolesche Funktion

$$f : \mathbb{B}^{n+k} \longrightarrow \mathbb{B}^{m+k},$$

denn das vollständige Verhalten von einem Takt zum nächsten wird durch n Eingangswerte und die k aktuellen Ausgangswerte der Rückkopplungen bestimmt und während des Taktes werden m Ausgabewerte berechnet und k neue Werte in die Rückkopplungen eingespeist. Von Außen sind die Werte in den Verzögerungsgliedern nicht sichtbar, d.h. man kann nur beobachten, welche Folge von Ausgabetupeln aus $\mathbb{B}^m$ bei einer bestimmten Folge von Eingabetupeln aus $\mathbb{B}^n$ generiert wird. Die Wertetupel der k Verzögerungsglieder kann man als einen inneren Zustand des Systems ansehen.

Somit kann ein Schaltwerk als ein dynamisches System über einer Zustandsmenge Q angesehen werden, das durch eine Funktion $\delta : \mathrm{In} \times Q \longrightarrow \mathrm{Out} \times Q$ beschrieben wird, mit $Q = \mathbb{B}^k$, $\mathrm{In} = \mathbb{B}^n$ und $\mathrm{Out} = \mathbb{B}^m$. Ein solches System wird auch als Mealy-Automat bezeichnet.

Definition: Ein Mealy-Automat ist ein 5-Tupel $(Q, \mathrm{In}, \mathrm{Out}, \delta, q_0)$, wobei

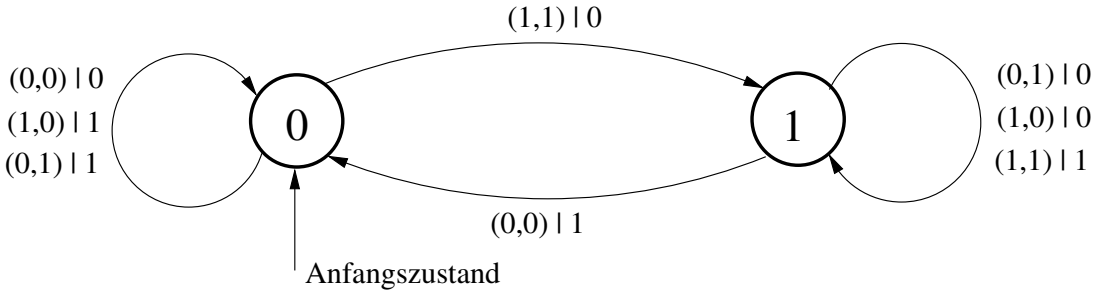
- Q ist eine endliche Zustandsmenge
- In und Out sind zwei endliche Mengen von Ein- und Ausgabesymbolen
- $\delta : \mathrm{In} \times Q \longrightarrow \mathrm{Out} \times Q$ ist eine Funktion, die als Zustandsüberführungsfunktion bezeichnet wird.
- $q_0 \in Q$ ist der Anfangszustand.

Ist $a_1, a_2, a_3, \ldots, a_n$ eine Folge von Eingabesymbolen, dann wird duch den Automaten eine eindeutige Folge von Ausgabesymbolen $c_1, c_2, c_3, \ldots, c_n$ berechnet:

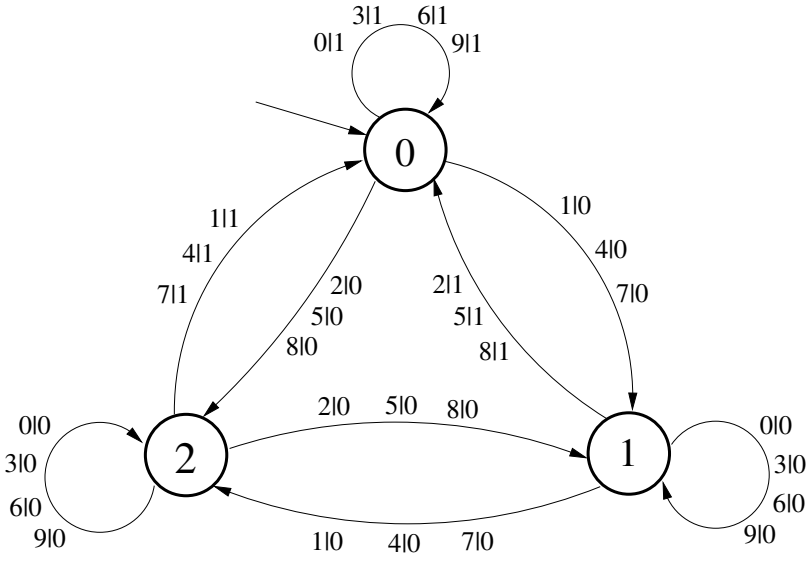
c_1 ergibt sich aus:	$\delta(a_1, q_0) = (c_1, q_1)$
c_2 ergibt sich aus:	$\delta(a_2, q_1) = (c_2, q_2)$
allgemein ergibt sich c_k aus:	$\delta(a_k, q_{k-1}) = (c_k, q_k)$

Zur anschaulichen Beschreibung von Mealy-Automaten verwendet man sogenannte Zustandsdiagramme. Dabei werden die Zustände als Knoten gezeichnet und die Zustandsüberführungsfunktion wird durch gerichtete Kanten mit speziellen Markierungen beschrieben.
Dazu betrachtet man für jedes Paar $(a, q) \in \mathrm{In} \times Q$ den Funktionswert $(c, q') = \delta(a, q)$ und zeichnet einen Pfeil von q nach q' der mit dem Label $a|c$ markiert wird.

Beispiel 1: Binärer Serienaddierer mit $\mathrm{In} = \mathbb{B}^2$ und $\mathrm{Out} = \mathbb{B}$.
Die Eingabe ist eine Folge von Bitpaaren $(a_0, b_0), (a_1, b_1), \ldots, (a_n, b_n)$, die zwei Zahlen $k = \sum_{i=0}^{n} a_i \cdot 2^i$ und $l = \sum_{i=0}^{n} b_i \cdot 2^i$ darstellen. Ziel ist die Berechnung der Binärdarstellung der Summe $k + l$.
Offensichtlich reichen zwei innere Zustände, nämlich 0 wenn bei der letzten Addition kein Übertrag entstanden ist und 1 wenn ein Übertrag entstanden ist.



Beispiel 2: Teilbarkeit von Dezimalzahlen durch 3.
Gegeben wird eine Dezimalzahl als Ziffernfolge (d.h. In= $\{0, 1, 2, \ldots, 9\}$) und man soll ausgeben, ob die bisher eingegebene Zahl durch 3 teilbar ist (1 für Ja und 0 für Nein). Man verwendet die Tatsache, dass eine Zahl genau dann durch 3 teilbar ist, wenn sich die Quersummen der Zahl durch 3 teilen lässt. Für den Automaten benötigt man nur drei Zustände, nämlich die Reste beim Teilen durch 3.



Beispiel 3: Teilstringerkennung in Binärstrings.
Der Automat soll erkennen, ob die letzen drei gelesenen Zeichen die Form 001 hatten. Wir konstruieren einen Automaten mit 3 Zuständen a, b, c, wobei der Zustand a symbolisiert, dass das letzte gelesene Zeichen eine 1 war oder wir ganz am Anfang stehen. Zustand b steht dafür, dass die aktuelle Eingabe auf 10 endet (oder nur aus dem Zeichen 0 besteht) und Zustand c symbolisiert, dass die aktuelle Eingabe auf 00 endet. Nur wenn der Automat im Zustand c ist und dann eine 1 liest, kann er eine 1 (also Ja) ausgeben, muss dafür aber wieder zurück in Zustand a.