— Informatik I —
Modul 4: Schaltwerke

Universität Zürich[UZH]

Modul 4: Schaltwerke

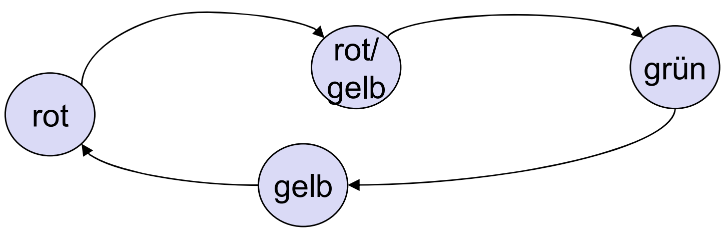
- Formale Grundlagen
  - Endliche Automaten
- Asynchrone Schaltwerke, Flip-Flops
- Synchrone Schaltwerke
- Spezielle Schaltwerke
  - Register, Zähler, Schieberegister

Schaltwerke

- Schaltnetze (kombinatorische Schaltungen):
  - Die Ausgabe hängt lediglich von den Werten der Eingangsvariablen zum gleichen Zeitpunkt ab.
  - Für diese Unterscheidung seien Laufzeitverzögerungen vernachlässigt!
- Schaltwerke (sequentielle Schaltungen):
  - Die Ausgabewerte hängen auch von Belegungen der Eingangsvariablen zu **vergangenen** Zeitpunkten ab.

Schaltwerke

- Man faßt alle Abhängigkeiten von Werten der Vergangenheit in einem sogenannten Zustand zusammen.
- Das Schaltwerk erzeugt damit seine Ausgabe in Abhängigkeit von den augenblicklichen Eingangsvariablen und seinem Zustand; diese Größen beeinflussen auch den nächsten Zustand des Schaltwerks.
- Man kann Schaltwerke als Implementierungen von deterministischen endlichen Automaten interpretieren.



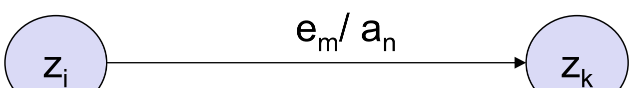
Zustände einer Ampel

Einführung in die Automatentheorie

- Ein 6-Tupel M = (E, A, Z, δ, ω, z_0) heißt Automat, wenn E, A und Z nichtleere Mengen sind
  - E ist die Menge der Eingangsbelegungen e,
  - A die Menge der Ausgangsbelegungen a und
  - Z die Menge der Zustände z.
- Überführungsfunktion $\delta: Z \times E \rightarrow Z$
  - δ ist eine auf der Menge $Z \times E$ definierte Funktion, deren Werte in Z liegen.
- Ausgabefunktion $\omega: Z \times E \rightarrow A$
  - ω eine auf der Menge $Z \times E$ definierte Funktion, deren Werte in A liegen.
- Grundzustand z_0

Beispiel

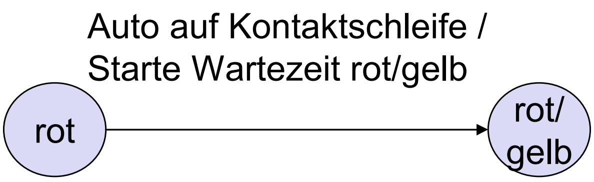
- Allgemein:



$\delta(z_i, e_m) = z_k$
$\omega(z_i, e_m) = a_n$

$e_m \in E$
$a_n \in A$
$z_i, z_k \in Z$

- Ampel (nicht vollständig spezifiziert)



E = {Auto auf Kontaktschleife, Fußgängertaste gedrückt, ...}

A = {Starte Wartezeit rot/gelb, Starte Summer, ...}

Z = {rot, rot/gelb, grün, gelb}

δ(rot, Auto auf Kontaktschleife) = rot/gelb
ω(rot, Auto auf Kontaktschleife) = Starte Wartezeit rot/gelb

Einführung in die Automatentheorie

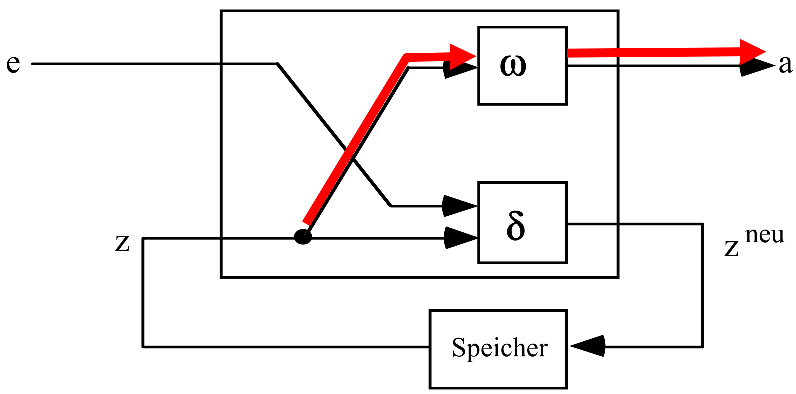
- Die Zustandsmenge Z ermöglicht die Speicherung von Wissen über Eingangsbelegungen der Vergangenheit.
- Die aktuelle Ausgabebelegung wird durch die Funktion ω, der neue Zustand durch die Funktion δ aus den aktuellen Eingangsbelegungen und dem alten Zustand erzeugt.

Mealy- und Moore-Automat (1)

- Hängt der Ausgabewert lediglich vom augenblicklichen Zustand ab, spricht man in diesem Falle von einem Moore-Automaten oder Moore-Schaltwerk.

$a = \omega(z)$ $\qquad z^{neu} = \delta(z,e)$

$a \in A$; $\qquad e \in E$; $\qquad z, z^{neu} \in Z$

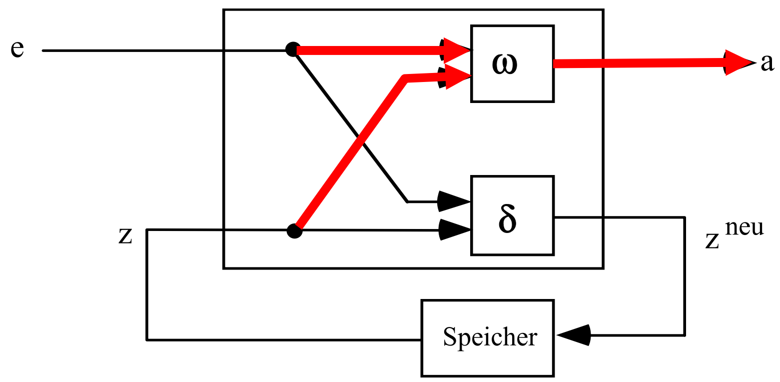


Mealy- und Moore-Automat (2)

- Geht auch die Eingabebelegung in die Berechnung des Ausgabewertes ein, erhält man in diesem Falle einen Mealy-Automaten oder Mealy-Schaltwerk.

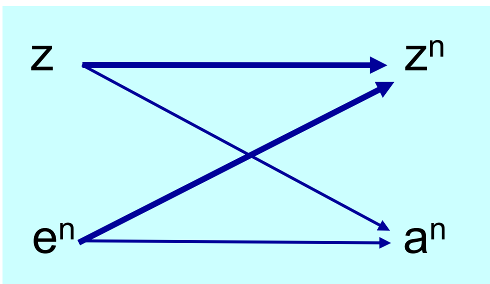
$a = \omega(z,e)$ $\qquad z^{neu} = \delta(z,e)$

$e \in E; \quad a \in A; \qquad z, z^{neu} \in Z$



Verhaltensunterschied dieser Automaten (1)

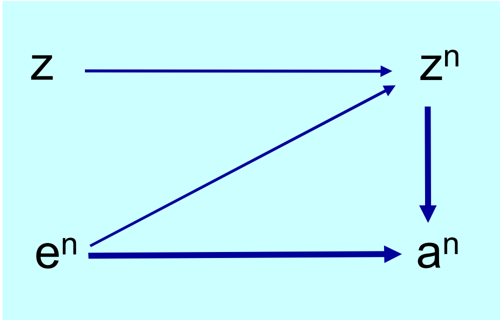
- Mealy-Automat:
  – Ausgabewerte können sich sofort mit der Änderung von Eingabevariablen ändern.
- Mealy-Automat 1. Art:
  – Es wird zunächst aus der neu anliegenden Eingabebelegung die Ausgabebelegung gebildet und dann in den Folgezustand gewechselt.
- Anwendung: synchrone Schaltwerke



Verhaltensunterschied dieser Automaten (2)

- Mealy-Automat 2. Art:
  - Es wird zunächst aus der neu anliegenden Eingabebelegung der Folgezustand und dann mit der noch anliegenden Eingabebelegung die Ausgabebelegung gebildet.

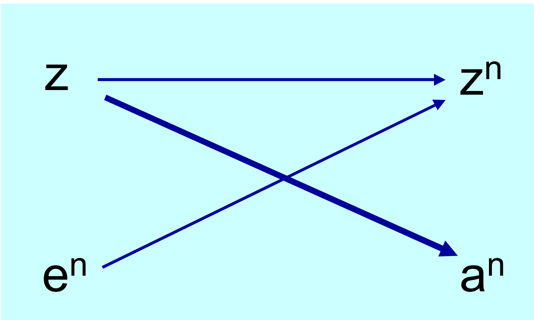
- Anwendung: asynchrone Schaltwerke



Verhaltensunterschied dieser Automaten (3)

- Moore-Automat:
  - Die Ausgabebelegung ist unabhängig von der Eingabebelegung, sie kann sich nur nach einem Zustandswechsel ändern.

- Anwendung: synchrone und asynchrone Schaltwerke



Darstellungsmöglichkeiten (1)

- Beim Entwurf eines Automaten liegt die Aufgabenstellung zunächst in einer informalen globalen Form vor z.B. durch:
  - Pflichtenheft:
    Beschreibt verbal, was die zu entwerfende Schaltung leisten soll.
  - Weitere Pläne:
    Ablaufpläne, Technologiebeschreibung, usw. zur Ergänzung der verbalen Beschreibung
- Für einen systematischen und ggf. rechnergestützten Entwurf ist ein Übergang zu einer formalisierten Beschreibung, die das Sollverhalten ausreichend spezifiziert, notwendig.

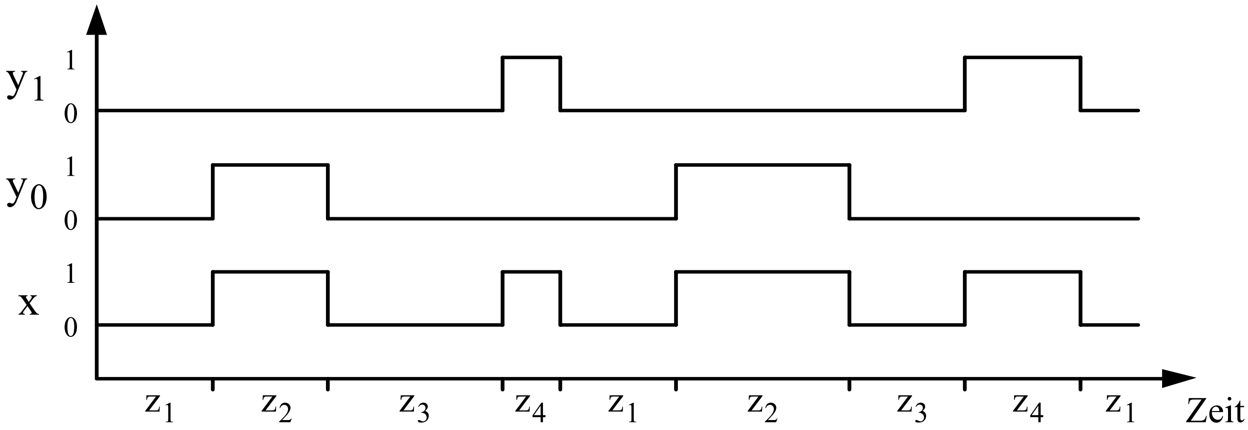
Darstellungsmöglichkeiten (2)

- Vier unterschiedliche Möglichkeiten zur formalisierten Beschreibung des Verhaltens eines Automaten sollen an einem Beispiel demonstriert werden:
- Beispiel:
  - Durch ein Schaltwerk soll eine einlaufende Impulsfolge am Eingang x derart verarbeitet werden, daß die Eingangsimpulse x abwechselnd an den beiden Ausgängen y_0 und y_1 erscheinen.
- Gesucht ist eine formalisierte Beschreibung dieses Problems.

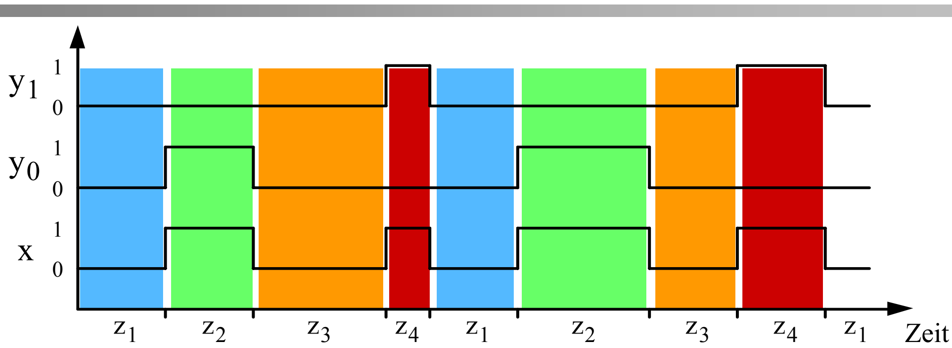
1. Zeitdiagramm

- Dient zur Veranschaulichung des Problems.

- Es wird eine beispielhafte Folge von Eingabebelegungen dargestellt.



1. Zeitdiagramm



z_1 = Ausgabe von y_0y_1 = 00 und warten auf x = 1 zur Ausgabe an Ausgang y_0

z_2 = Ausgabe von y_0y_1 = 10 und warten auf x = 0

z_3 = Ausgabe von y_0y_1 = 00 und warten auf x = 1 zur Ausgabe an Ausgang y_1

z_4 = Ausgabe von y_0y_1 = 01 und warten auf x =0

2. Ablauftabelle

Sind die Zustände bekannt, so kann die Überführungsfunktion $z_{k+1} = \delta(z_k,x)$ und die Ausgabefunktion $(y_0,y_1) = \omega(z_k,x)$ in einer Ablauftabelle dargestellt werden.

z_k	x	z_{k+1}	$y_0\ y_1$
z_1	0	z_1	0 0
z_1	1	z_2	1 0
z_2	1	z_2	1 0
z_2	0	z_3	0 0
z_3	0	z_3	0 0
z_3	1	z_4	0 1
z_4	1	z_4	0 1
z_4	0	z_1	0 0
Start-zustand z_k	Eingangs-belegung	Folge-zustand z_{k+1}	Ausgangs-belegungen

3. Automatentabelle

- Andere Darstellungsform desselben Sachverhalts:
  - In senkrechter Richtung: die Zustände.
  - In waagerechter Richtung: die Eingangsbelegungen.
  - In der Matrix: die Folgezustände.

z_k	z_{k+1} $x=0$	z_{k+1} $x=1$	$y_0\ y_1$
z_1	z_1	z_2	0 0
z_2	z_3	z_2	1 0
z_3	z_3	z_4	0 0
z_4	z_1	z_4	0 1

Moore-Automat

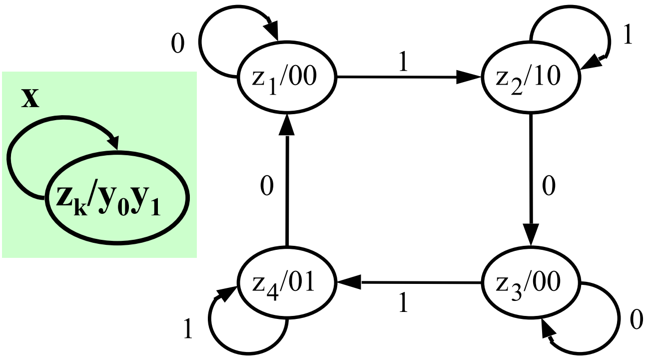
z_k	$z_{k+1}/y_0\ y_1$ $x=0$	$z_{k+1}/y_0\ y_1$ $x=1$
z_1	$z_1/00$	$z_2/10$
z_2	$z_3/00$	$z_2/10$
z_3	$z_3/00$	$z_4/01$
z_4	$z_1/00$	$z_4/01$

Mealy-Automat

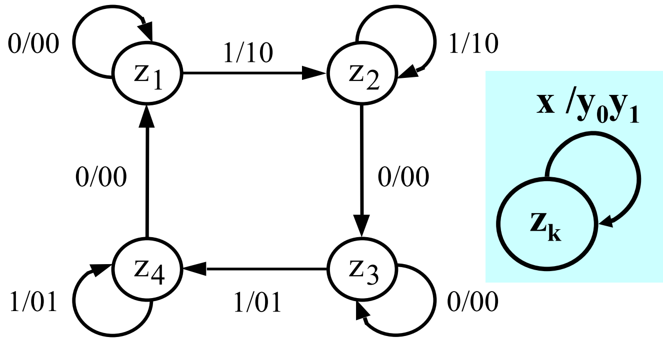
4. Automatengraph

- Die Überführungs- und Ausgabefunktionen werden in einem gerichteten Graphen AG = (Z,K) dargestellt, wobei Z die Menge der Zustände z und K die Menge der Übergänge k zwischen den Zuständen ist.

- Die Eingabebelegungen werden an die Kanten der zugehörigen Zustandsübergänge geschrieben.



Moore-Automat



Mealy-Automat

Realisierung von Automaten

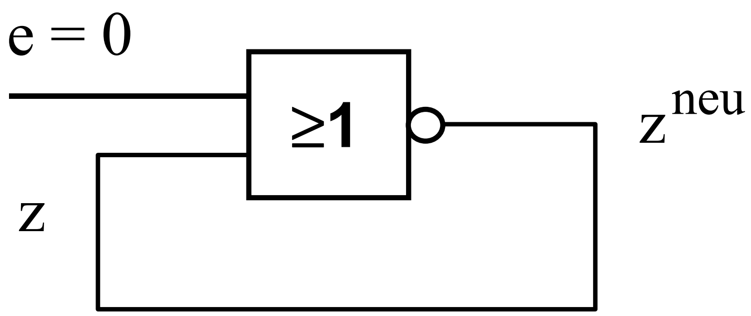
- Zur Speicherung vergangener Informationen ist ein Zustandsspeicher erforderlich.

- Einfachste Form dieses Zustandsspeichers:

  – Rückkopplung

- Durch die Rückkopplung lassen sich die in den Eingangsvariablen nicht mehr repräsentierten Informationen wieder am Eingang zur Verfügung stellen.

Beispiel: Rückgekoppeltes NOR-Gatter



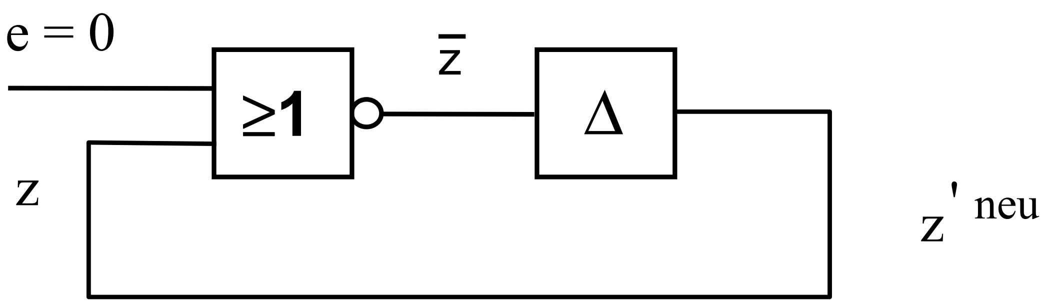
Als ideales Gatter betrachtet, ist die Schaltung unzulässig, denn es müßte gleichzeitig gelten:

$z^{neu} = \bar{z}$ und

$z = z^{neu}$

Rückgekoppeltes NOR-Gatter

In der Realität hat jede Schaltung hat eine Verzögerungszeit größer 0 (Totzeitmodell).



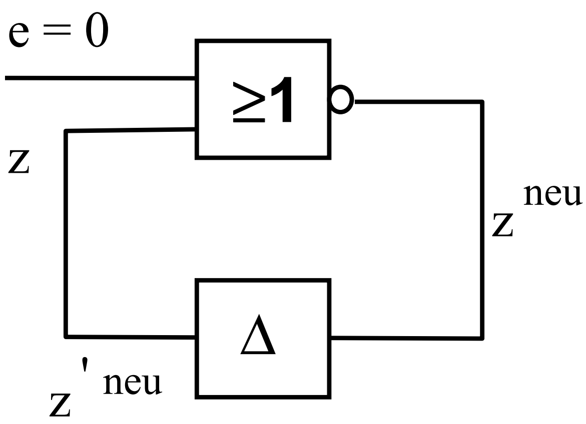
Mit dieser Verzögerung erhält man:

$$z'^{neu}(t+\Delta) = \overline{z(t)}$$

$$z(t+\Delta) = z'^{neu}(t+\Delta)$$

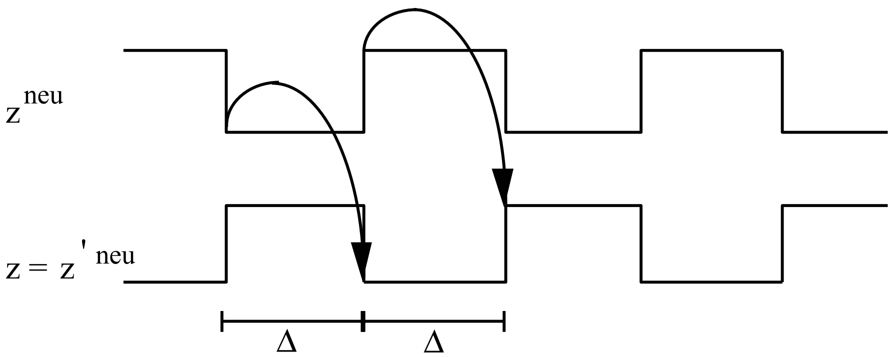
Rückgekoppeltes NOR-Gatter

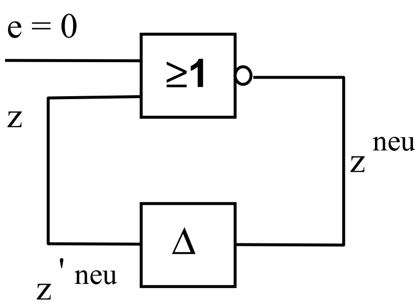
Zeichnet man die Schaltung etwas anders, so sieht man, daß das **Δ-Verzögerungsglied** dem **Speicher** entspricht.



Rückgekoppeltes NOR-Gatter

Das Zeitverhalten dieser Schaltung im Zeitdiagramm:





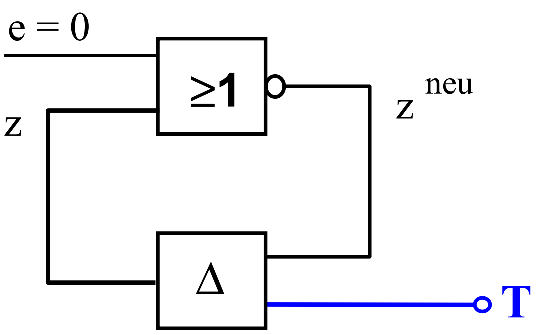
Das Verhalten der Schaltung ist stark von den Verzögerungszeiten abhängig.

Rückgekoppeltes NOR-Gatter

- Wie kann man ein von den Verzögerungszeiten unabhängiges Verhalten erreichen?

- Der Speicher wird so aufgebaut, daß z seinen alten Wert so lange beibehält, bis z explizit durch ein externes Signal T auf z^{neu} gesetzt wird.

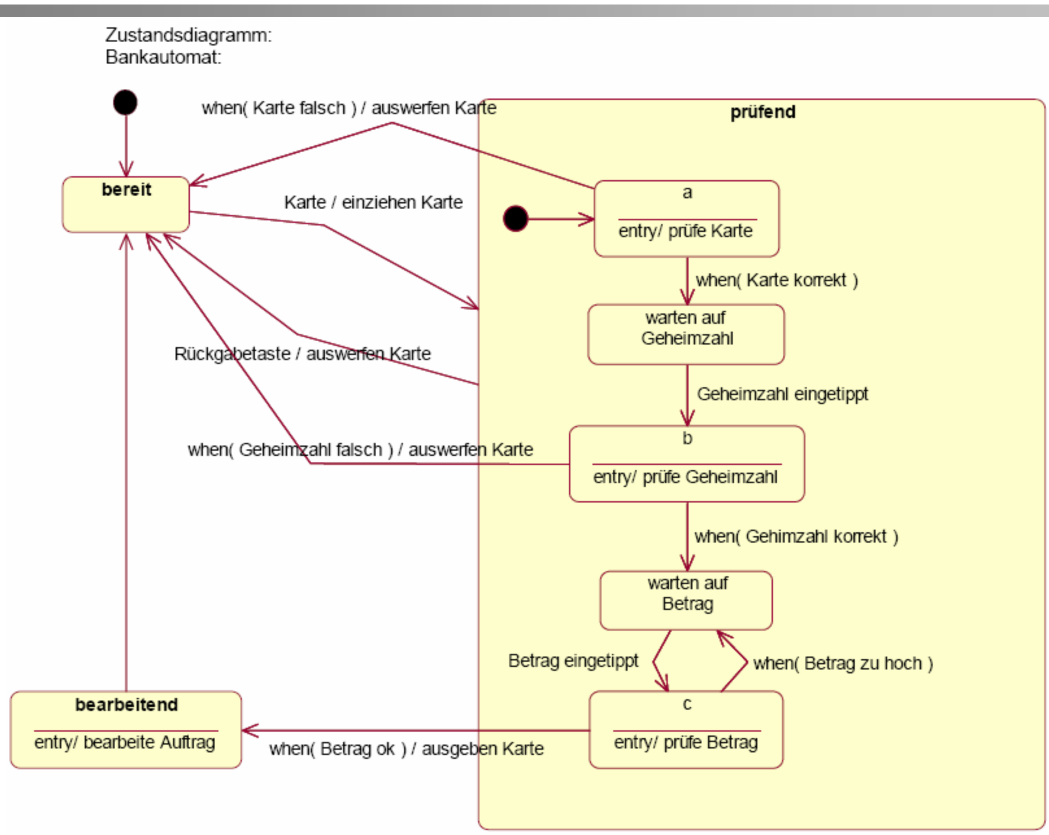
Beispiel:



z behält seinen Wert solange, bis T von 0 auf 1 wechselt.

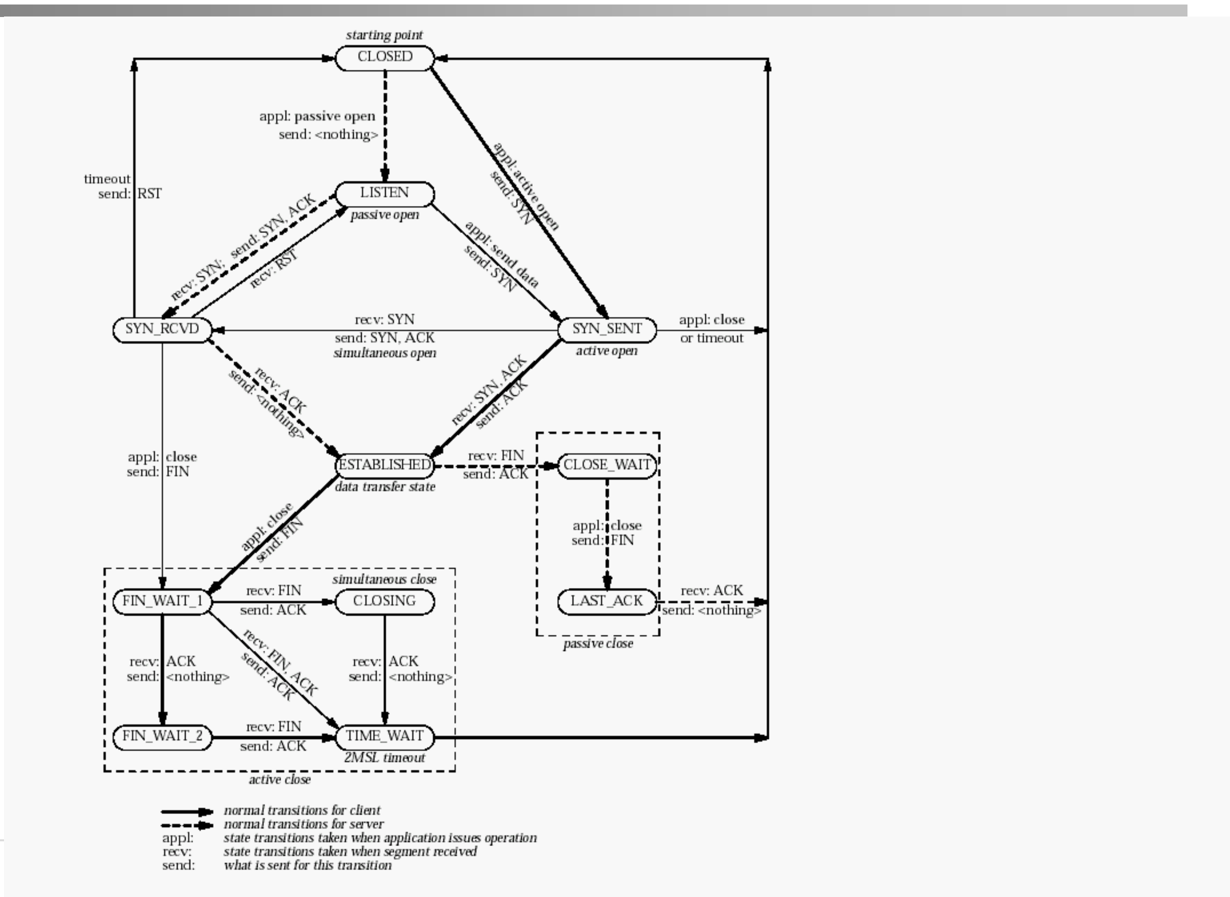
Dann wird z auf z^{neu} gesetzt und behält diesen Wert bis zum nächsten 01-Wechsel von T.

Automatengraph (Bancomat)



Automatengraph (Zustandsübergangsdiagramm)

TCP: Transmission Control Protocol



Modul 4: Schaltwerke

- Formale Grundlagen
  - Endliche Automaten
- Asynchrone Schaltwerke, Flip-Flops
- Synchrone Schaltwerke
- Spezielle Schaltwerke
  - Register, Zähler, Schieberegister

Definitionen

- Werden alle Zustandsspeicher von einem oder mehreren zentralen Synchronisationssignal(en) T (Takt) gesteuert, so spricht man von einem synchronen Schaltwerk.
- Anderenfalls spricht man von einem asynchronen Schaltwerk.
- Die Synchronisation über einen Takt kann flankengesteuert und pegelgesteuert sein

Pegelsteuerung

- Der Speicher ist während einer Takthälfte transparent, während der anderen speichert er.
- Die Eingänge wirken sich nur auf den Zustand aus, wenn der Takt z.B. den Wert 1 hat. Ist der Takt 0, wird der Zustand gespeichert.
- Nachteil: Die Eingangssignale können sich während der aktiven (transparenten) Taktperiode mehrfach ändern.
- Einfachste Realisierung: Konjunktive Verknüpfung jeder Eingangsvariablen mit dem Takt.
- Pegelgesteuerte Zustandsspeicher werden auch Latches genannt.

Flankensteuerung

- Nur während der positiven (0 → 1) oder der negativen (1 → 0) Taktflanke werden die Eingabewerte in den Speicher übernommen.

- Vorteil:
  Eingänge müssen nur für eine sehr kurze Zeitspanne gültig sein (und nicht über eine ganze Takthälfte wie bei der Pegelsteuerung).
  ⇒ die Auswertezeitpunkte sind exakter definiert

Schaltsymbol für einen flankengesteuerten Takteingang

Synchrone Schaltwerke vs. Asynchrone Schaltwerke (1)

- Synchrone Schaltwerke:
  - Mittlere und größere Schaltwerke werden fast immer als synchrone Schaltwerke entworfen.

- Vorteil:
  - Leichter zu analysieren und zu entwerfen als asynchrone Schaltwerke.

- Grund:
  - Synchrone Schaltwerke sind unabhängig von (teilweise fertigungsabhängigen) Verzögerungszeiten.

Synchrone Schaltwerke vs. Asynchrone Schaltwerke (2)

- Wird die Dauer des Taktes nur größer als die maximale Verzögerungszeit im Schaltnetz gewählt:

$\Rightarrow$ Die Ausgänge der Schaltnetze δ und ω haben sich stabilisiert, bevor sie sich auf z^{neu} auswirken.

$\Rightarrow$ Zur Analyse und Synthese eines synchronen Schaltwerks muß man lediglich die Schaltnetze δ und ω betrachten.

- Die Schaltung kann an den Stellen aufgetrennt werden, an denen die Speicherelemente sitzen.

Synchrone Schaltwerke vs. Asynchrone Schaltwerke (3)

- Asynchrone Schaltwerke:

- Der Entwurf asynchroner Schaltwerke ist aus zwei Gründen von Bedeutung:

  – Die in synchronen Schaltwerken benutzten Speicherbausteine sind selbst kleine asynchrone Schaltwerke.

  – Immer schneller werdende Bausteine zwingen zu teilweise asynchronen Entwurfstechniken.

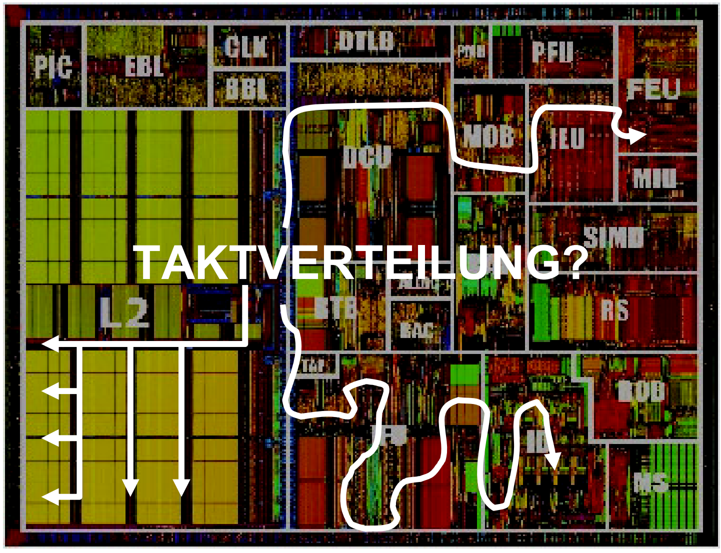
Warum ?

Synchrone Schaltwerke vs. Asynchrone Schaltwerke (4)

- Begründung:

  – Werden die Verzögerungszeiten der verwendeten Bausteine kleiner als die Signallaufzeiten auf der Schaltungsplatine/auf dem Chip (ca. 20-30 cm/ns)

  ⇒ dann ist der Takt ist nicht länger synchron, da er die einzelnen Bausteine je nach Entfernung zu für die Bausteine unterschiedlichen wahrnehmbaren Zeitpunkten erreicht!

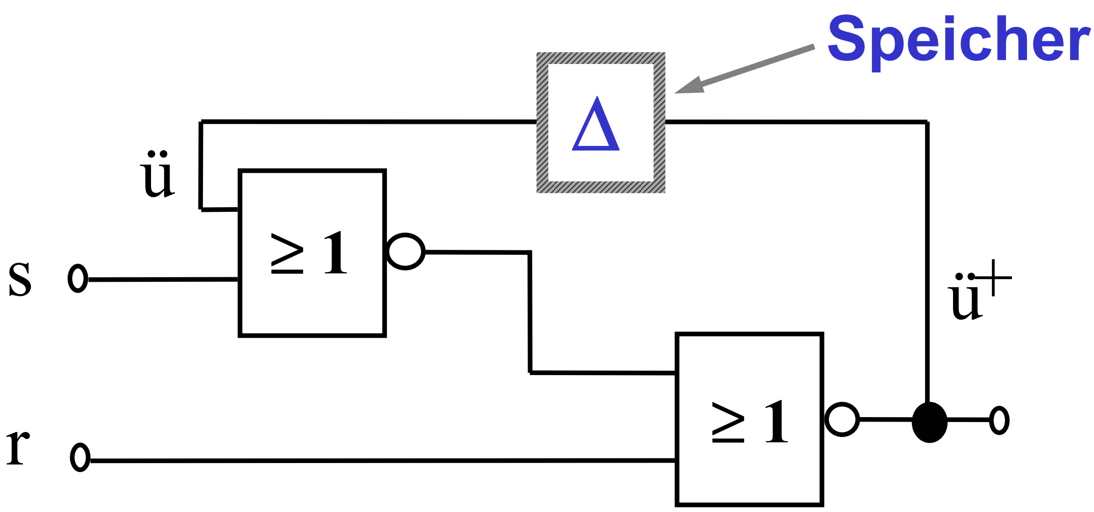


1 GHz Takt = 1 ns Taktdauer!

Beispiel: Asynchrones SW (Schaltbild des Speichers)

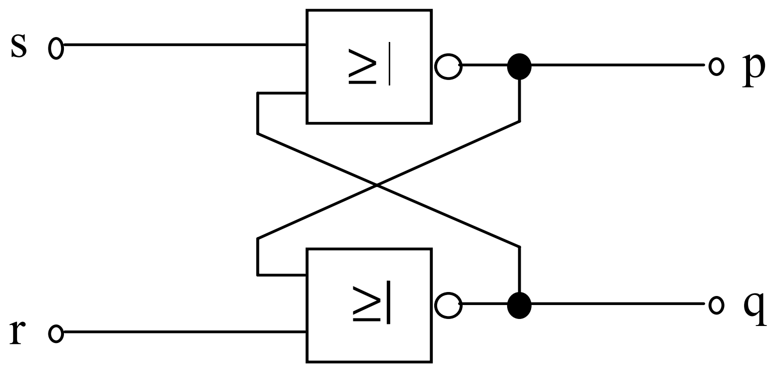
Übergangsgleichung: $\ddot{u}^+ = \overline{r \vee \overline{\ddot{u} \vee s}} = \bar{r}\,(\,\ddot{u} \vee s\,)$

Ausgangsgleichung: $q = \ddot{u}$



Beispiel: Asynchrones RS-Flipflop

- Dieser Speicher ist ein Standardelement.
  - Es wird als asynchrones RS-Flipflop (bistabile Kippstufe) bezeichnet.
  - Es wird üblicherweise nur etwas anders gezeichnet:



  - Der zusätzliche Ausgang p ist im allgemeinen komplementär zu q, solange r und s nicht gleichzeitig 1 sind: $\mathbf{p = \overline{q}}$
  - Nur für die beim Entwurf ausgesparte (und damit verbotene) Eingabebelegung (r, s) = (1, 1) ist p = q = 0

Probleme asynchroner Schaltwerke

- Asynchrone Schaltwerke arbeiten ohne einen zentralen Takt:
  - Sie reagieren sofort auf jede Änderung der Eingangs- und Zustandsvariablen.
  - Sie sind sehr störempfindlich
- Wettläufe von Zustandsvariablen:
  - Diese verursachen falsche Zustandsübergänge
  - Abhilfe: Wettlauffreie Zustandskodierung
- Hasardfehler in den Übergangs-Schaltnetzen:
  - Hierauf reagieren asynchrone Schaltwerke naturgemäß sehr empfindlich. Hasardfehler können ebenfalls falsche Zustandsübergänge und Oszillationen verursachen
  - Abhilfe: Entwurf hasardarmer Schaltnetze für die Übergangs- und Ausgabefunktionen.

Probleme asynchroner Schaltwerke

- Zur weiteren Verringerung des Störrisikos arbeiten asynchrone Schaltwerke darüber hinaus meist im sogenannten

normal fundamental mode.

- Hierbei darf sich maximal eine Eingangsvariable gleichzeitig ändern.
- Ein Eingabewechsel kann erst dann erfolgen, wenn alle internen Änderungen abgeklungen sind.

Wiederholung

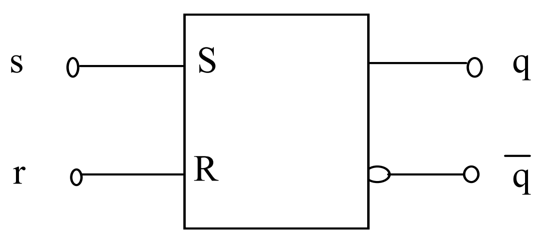
- Schaltnetze: Ausgabe hängt nur von Eingangssignalen ab (kombinatorische Schaltungen, combinational circuits)
- Schaltwerke: Ausgabe kann auch von internen Zustand abhängen (sequentielle Schaltungen, sequential circuits)
- Synchrones Schaltwerk: Zustandsspeicher ist taktgesteuert, andernfalls asynchron

Flipflops als Zustandsspeicher

- Die Probleme asynchroner Schaltwerke treten bei synchronen Schaltwerken nicht auf.
- Da alle Zustandsspeicher bei synchronen Schaltwerken durch einen zentralen Takt gesteuert werden, können sich alle Übergänge und die damit verbundenen Wettläufe stabilisieren, bevor der neue Zustand eingenommen wird.
- Synchrone Schaltwerke benötigen taktgesteuerte Zustandspeicher
- Hierfür werden Flipflops verwendet.
- Es existieren eine Reihe verschiedener Flipflop-Typen.

RS-Flipflop

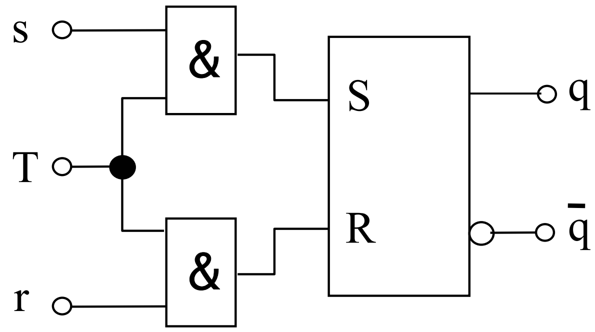
- Verhalten (RS-Flipflop):
  - Eingang s soll den Speicher setzen (s=1 ⇒ Ausgang q=1)
  - Eingang r soll den Speicher rücksetzen (r=1) ⇒ q=0)
  - Speichern: r und s beide 0 ⇒ q behält letzten Wert
  - Verboten: r und s gleichzeitig 1 ⇒ die Ausgänge p und q sind komplementär
  - Die Zustandsvariable q und ihre Negation $\overline{q}$ (= p) stehen am Ausgang zur Verfügung.
- Schaltsymbol des asynchronen RS-Flipflops:

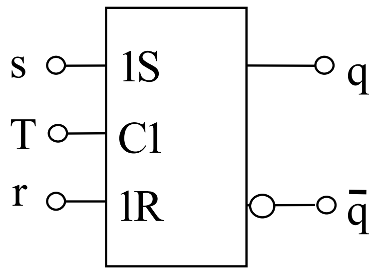


RS-Flipflop → pegelgesteuertes RS-Latch

- Um das RS-Flipflop in einem synchronen Schaltwerk verwenden zu können:
  - Taktsignal muß eingeführt werden, welches die Änderung der Zustandsvariablen in der inaktiven Taktphase verhindert.
  - Dies ist leicht zu erreichen, indem man die beiden Eingänge durch je ein UND-Gatter mit diesem Takt verknüpft:

- Wir erhalten das pegelgesteuerte RS-Latch

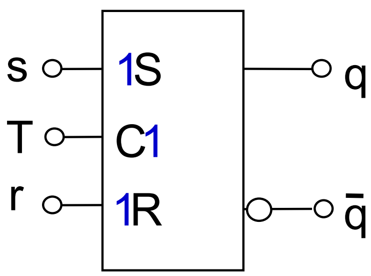




Anmerkung zur Notation

- Die Ziffer 1 bei den Eingängen (1S, 1R) bedeutet, daß sie in ihrer Wirksamkeit von dem ebenfalls mit 1 gekennzeichneten Takt C1 abhängen.

- Verursacht ein Eingang die Abhängigkeit, so folgt die Ziffer der Eingangsvariablen, anderenfalls geht sie voraus.



Ansteuertabelle (RS-Flipflop)

- Beim Entwurf synchroner Schaltwerke sind Zustand und gewünschter Folgezustand bekannt.
- Gesucht sind die Werte der Ansteuervariablen der Flipflops.
- Diese lassen sich leicht mit Hilfe der sog. Ansteuertabelle eines Flipflops bestimmen.
- Die Ansteuertabelle gibt den Zustandsübergang eines Flipflops unter den verschiedenen Eingabebelegungen wieder.
  - Sie läßt sich i.a. auf einfache Weise aus der Funktionstabelle der Ausgabe- und Übergangsfunktionen gewinnen.

Ansteuertabelle (RS-Flipflop)

Ansteuertabelle des asynchronen RS-Flipflops:

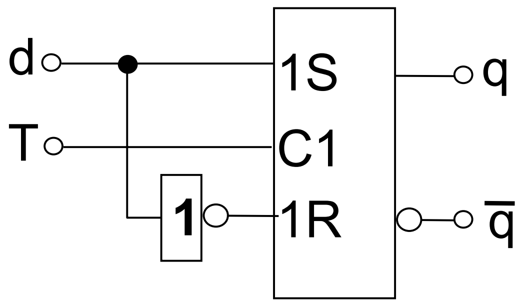
q^t	q^{t+1}	r	s	
0	0	-	0	Halten
0	1	0	1	Setzen
1	0	1	0	Rücksetzen
1	1	0	-	Halten

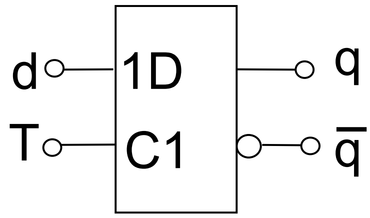
Voraussetzung:

Es dürfen keine unerlaubten Eingangsbelegungen auftreten.

D-Flipflop

- Bei einem RS-Flipflop ist stets die Nebenbedingung ($r \wedge s = 0$) zu beachten.
- Führt man eine Eingangsvariable d bejaht zum S-Eingang und negiert zum R-Eingang, ist diese Bedingung stets erfüllt.
- Damit erhält man ein sogenanntes D-Latch.





Verhalten des D-Flipflops

Verhalten:

- Der anliegende Eingabewert wird in allen Fällen als Flipflopzustand übernommen und einen Takt lang gespeichert.
- Das Eingangssignal wird um eine Taktperiode verzögert am Ausgang zur Verfügung gestellt.
- Daher der Name D-Latch von "to delay" = verzögern

Funktionstabelle:

d	q^t	q^{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

Ansteuertabelle des D-Latch

q^t	q^{t+1}	d
0	0	0
0	1	1
1	0	0
1	1	1

d	q^t	q^{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

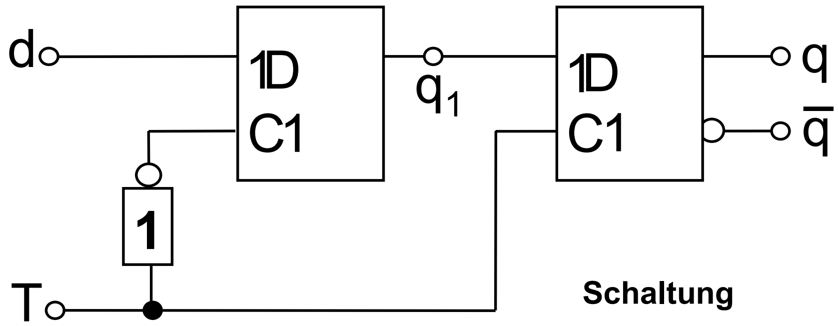
Leicht aus der Funktionstabelle durch Permutieren der Spalten gewinnbar.

Taktflankengesteuertes D-Flipflop

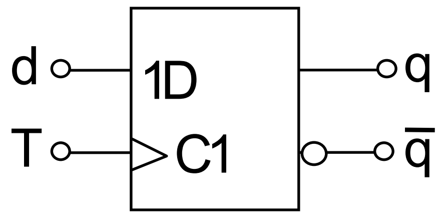
- Ein taktflankengesteuertes D-Flipflop erhält man durch die Zusammenschaltung zweier D-Latches, die mit komplementären Taktpegeln gesteuert werden.
- Das erste Latch wird Master-Latch, das zweite Slave-Latch genannt.
- Ein solches Flipflop wird auch als Master-Slave-Flipflop bezeichnet.

Funktionsweise

- Während T = 0 folgt das erste Latch den Änderungen des Eingangssignals d, während das zweite Latch verriegelt ist.
- Ändert sich T von 0 auf 1 (positive Taktflanke), wird das erste Flipflop verriegelt.
- Unabhängig von den nun auftretenden Änderungen von d bleibt der Ausgabewert q_1 gleich dem Wert von d, der beim 0-1-Wechsel des Taktes anlag.
- Dieser Wert wird in das zweite Latch übernommen und dort auch weiter gespeichert, wenn T wieder auf 0 zurückgeht.



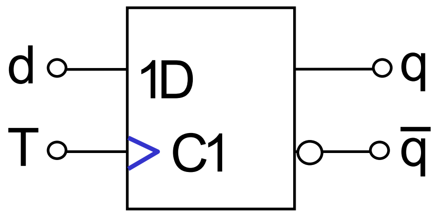
Schaltung



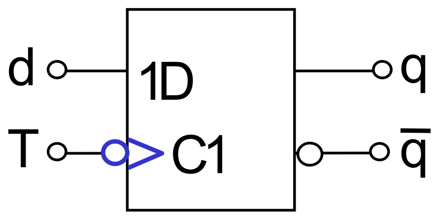
Symbol

Anmerkungen

- D-Flipflops sind die am einfachsten zu realisierenden flankengesteuerten Speicherelemente.
  - Sie sind wegen des geringen Flächenbedarfs die in integrierten Schaltungen am häufigsten verwendeten Speicherglieder.
- Im Schaltsymbol wird die Taktflankensteuerung durch ein Dreieck am Takteingang spezifiziert.



- Bei einer Steuerung mit der negativen Taktflanke wird ein Negationszeichen vor das Dreieck gesetzt.

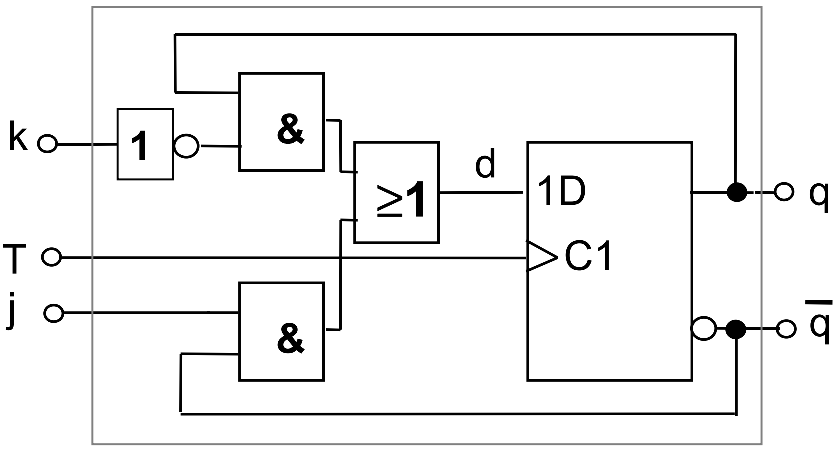


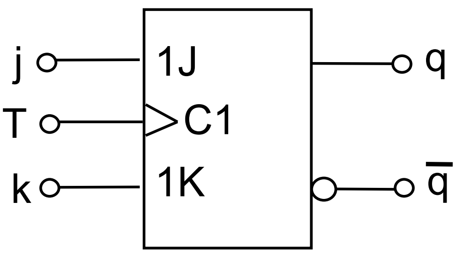
JK-Flipflop

- Beim RS-Flipflop war die Eingangsvariablen-Kombination $r = s = 1$ verboten
- Ziel: Ein Flipflop entwerfen, welches diese Kombination nutzt.
- Als vierte Funktion neben "speichern", "setzen" und "rück-setzen" soll bei Eingangskombination $r = s = 1$ der Flipflop-Inhalt komplementiert werden.
- Bezeichnung:
  - j: resultierender Setzeingang
  - k: resultierender Rücksetzeingang

  $\rightarrow$ JK-Flipflop
- Dieses Verhalten läßt sich durch Zusatzbeschaltung schon bekannter Flipflops erreichen.

Schaltbild des synchronen JK-Flipflops





$$d = q^t\, \bar{k} \vee \bar{q}^t\, j$$

Funktions-/Ansteuertabelle des JK-Flipflops

Verkürzte Funktionstabelle des JK-Flipflops:

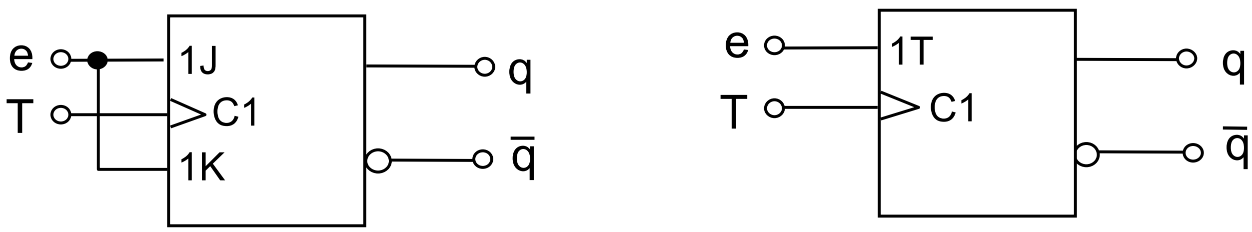
j	k	q^{t+1}	Funktion
0	0	q^t	speichern
0	1	0	rücksetzen
1	0	1	setzen
1	1	$\overline{q^t}$	wechseln

Aus obiger Tabelle läßt sich auch leicht die Ansteuertabelle des JK-Flipflops gewinnen:

q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

Das T-Flipflop

- Ein T-Flipflop ("to toggle", kippen) hat nur einen Eingang.
- Liegt an diesem Eingang eine "1", kippt das Flipflop mit jedem Taktimpuls in einen anderen Zustand, hat die Eingangsvariable den Wert "0", behält das Flipflop seinen alten Zustand bei.
- Durch geeignete Eingangsbeschaltung eines JK-Flipflops läßt sich leicht das Verhalten eines T-Flipflops erzeugen.



Synchrones T-Flipflop

T-Flipflop: Verkürzte Funktionstabelle

- Verkürzte Funktionstabelle des T-Flipflops

e	q^{t+1}	Funktion
0	q^t	speichern
1	$\overline{q^t}$	wechseln

- Ein synchrones Setzen oder Rücksetzen des T-Flipflops ist mit dem Eingang e nicht möglich.

- Um das Flipflop in einen definierten Grundzustand zu bringen, ist daher ein zusätzlicher Setz- oder Rücksetzeingang notwendig.

Zusammenfassung

- RS-Flipflop (asynchron): r=s=1 verboten
  → RS-Latch (getaktet, pegelgesteuert)

- D-Flipflop, D-Latch: r und s =0 immer beachtet

- Taktflankengesteuertes D-Flipflop durch Zusammenschaltung zweier D-Latches

- JK-Flipflop: r und s = 1 → Ausgang komplementieren

- T-Flipflop: Eingang 1 → Ausgang komplementieren, sonst speichern

Wichtige Hilfsmittel: Ansteuertabellen

RS-Flipflop

q^t	q^{t+1}	r	s
0	0	-	0
0	1	0	1
1	0	1	0
1	1	0	-

D-Flipflop

q^t	q^{t+1}	d
0	0	0
0	1	1
1	0	0
1	1	1

JK-Flipflop

q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

T-Flipflop

q^t	q^{t+1}	e
0	0	0
0	1	1
1	0	1
1	1	0

Modul 4: Schaltwerke

- Formale Grundlagen
  - Endliche Automaten
- Asynchrone Schaltwerke, Flip-Flops
- Synchrone Schaltwerke
- Spezielle Schaltwerke
  - Register, Zähler, Schieberegister

Entwurf synchroner Schaltwerke

- Die Vorgehensweise beim Entwurf synchroner Schaltwerke soll ebenfalls an einem Beispiel erläutert werden.

- Beispiel: Serienaddierer
  - Zwei beliebig lange Dualzahlen sollen stellenweise addiert werden.
  - Die Addition beginnt mit der Stelle niedrigster Wertigkeit.
  - In jeder nachfolgenden Stelle muß der Übertrag der vorhergehenden Stelle berücksichtigt werden.
  - Die Zahlen werden bitweise eingegeben, pro Taktschritt eine Stelle.
  - Die Ausgabe soll ebenfalls bitweise erfolgen, wobei die Ausgabefolge zu jedem Zeitpunkt die Summe der bisherigen Eingabefolgen (ohne Übertrag) darstellt.

$$\begin{array}{r} 1001101001 \\ +\ \ \underline{1010011010} \\ 10100000011 \end{array}$$

617 + 666 = 1283
- später mehr
zur Rechnerarithmetik!

Grundsätzliche Vorgehensweise

Ausgangsbasis: verbale Aufgabenstellung

1. Zusammenstellung der Ein- und Ausgabevariablen
2. Festlegung der Zustände
3. Entwerfen des Automatengraphen
4. Aufstellen einer Automatentafel
5. Wahl der Zustandskodierung
6. Erzeugung der kodierten Ablauftabelle
7. Erweiterung der Ablauftabelle um Flipflops
8. Minimierung der Ausgangs- und Ansteuernetze der Flipflops
9. Schaltwerk zeichnen

Aufstellen des Automatengraphen

- Beispiel des Serienaddierers:
  Zwei Eingabevariablen-Folgen, eine Zahl x und eine Zahl y, sowie eine Folge von Ausgabewerten, die Summe s.

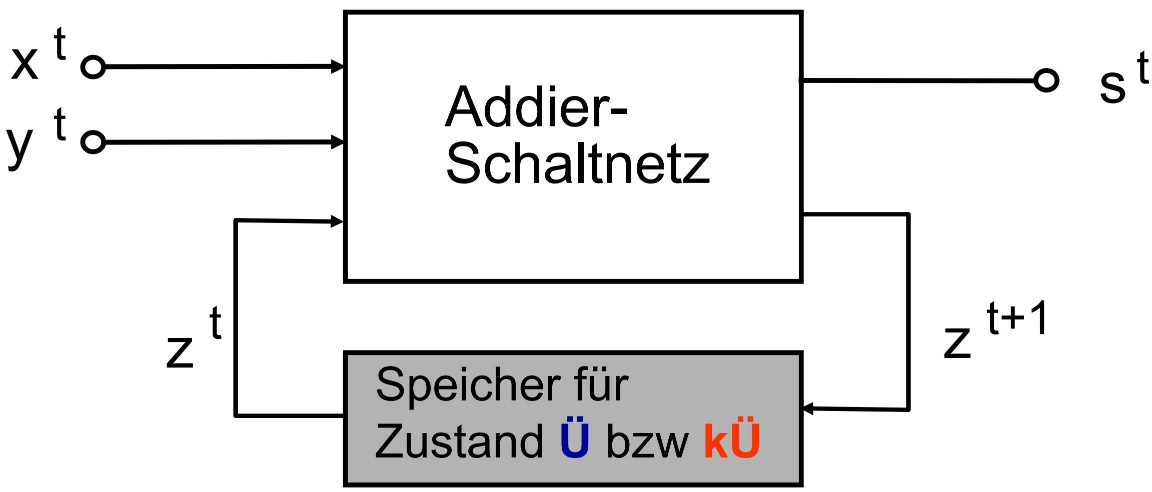
- In einem gegebenen Takt muß von der Vorgeschichte des Schaltwerks lediglich der Übertrag aus dem vorhergehenden Takt bekannt sein.

- Demnach reichen zwei Zustände aus:
  - Zustand Ü: Wird in dem Fall eingenommen, daß ein Übertrag aus der vorhergehenden Stelle zu berücksichtigen ist
  - Zustand kÜ: Repräsentiert den anderen Fall (kein Übertrag)

Blockschaltbild des Serienaddierers

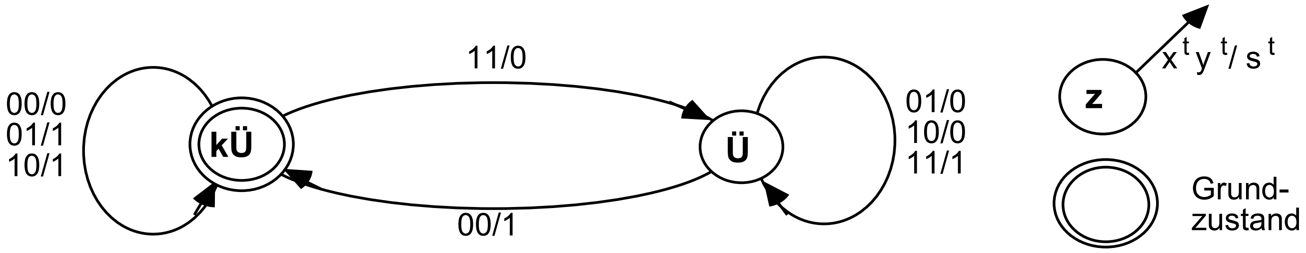
Eingabe: x und y Ausgabe: s

Zustandsspeicher für Ü bzw. kÜ



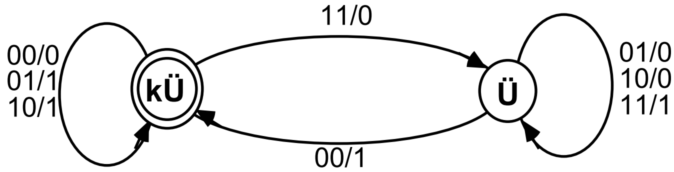
Automatengraph

- Es ist ein Mealy-Schaltwerk nötig, da die Ausgabe von den aktuellen Werten der Eingabevariablen abhängen soll.
- Im Automatengraphen werden deshalb die Ausgabebelegungen an die Kanten geschrieben.
- Automatengraph des Serienaddierers:



Aufstellen der Automatentafel

Aus dem Automatengraphen läßt sich die Automatentafel ableiten.



Automatentafel des Serienaddierers:

z	z^+ / s, x y = 00	01	10	11
kÜ	kÜ /0	kÜ /1	kÜ /1	Ü /0
Ü	kÜ /1	Ü /0	Ü /0	Ü /1

Bei synchronen Schaltwerken werden stabile Zustände nicht gesondert markiert, da angenommen wird, daß alle Zustände bis zum nächsten Taktzyklus stabil sind.

Wahl der Zustandskodierung

- Bei asynchronen Schaltwerken:
  - Zustandskodierung sehr kritisch
  - Wahl einer geeigneten Zustandskodierung ist für das Funktionieren des Schaltwerks entscheidend
- Bei synchronen Schaltwerken:
  - Zustandskodierung unkritisch
  - Ein synchrones Schaltwerk funktioniert grundsätzlich mit jeder eindeutigen Zustandskodierung
- Eine gute Zustandskodierung kann bei synchronen Schaltwerken jedoch den Schaltungsaufwand reduzieren
  - Zustandskodierung minimaler Länge bei k Zuständen:

$$\lceil \mathrm{ld}\ k \rceil = \text{minimale Anzahl der Flipflops}$$

Zustandskodierung beim Serienaddierer

- Zustandskodierung trivial, da nur zwei Zustände Ü und kÜ

➔ Nur eine Zustandsvariable ü

z	ü
kÜ	0
Ü	1

- Erzeugen der Ausgabe- und Übergangs-Schaltnetze:
  - Einsetzen der Zustandskodierung in die Automatentafel

    ➔ Kodierte Ablauftabelle des Schaltwerks

Kodierte Ablauftabelle

z	ü
kÜ	0
Ü	1

Zustandskodierung

z	z^+ / s, x y = 00	01	10	11
kÜ	kÜ/0	kÜ/1	kÜ/1	Ü/0
Ü	kÜ/1	Ü/0	Ü/0	Ü/1

Automatentafel

	$ü^t$	x^t	y^t	$ü^{t+1}$	s^t
kÜ	0	0	0	0	0
	0	0	1	0	1
	0	1	0	0	1
	0	1	1	1	0
Ü	1	0	0	0	1
	1	0	1	1	0
	1	1	0	1	0
	1	1	1	1	1

Kodierte Ablauftabelle

Anmerkungen

- Aus der kodierten Ablauftabelle lassen sich bereits die Funktionsausdrücke der Ausgabeschaltnetze ableiten
- Zur Erzeugung der Schaltnetze der Überführungsfunktion muß zuvor noch der verwendete Flipflop-Typ festgelegt werden
- Jeder Flipflop-Typ muß anders angesteuert werden
- Die Wahl eines Flipflop-Typs beeinflußt die Größe der Überführungsschaltnetze.
- Auch die Güte einer Zustandskodierung kann nur im Hinblick auf einen bestimmten Flipflop-Typ beantwortet werden.

Wahl des Flipflop-Typs

- Der Serienaddierer soll mit JK-Flipflop realisiert werden.
- Ansteuertabelle des JK-Flipflops:

q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

Ansteuerung des Flipflops

- Kodierte Ablauftabelle wird um die Ansteuereingänge des Flipflops erweitet:

q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

$ü^t$	x^t	y^t	$ü^{t+1}$	s^t	j^t	k^t
0	0	0	0	0	0	-
0	0	1	0	1	0	-
0	1	0	0	1	0	-
0	1	1	1	0	1	-
1	0	0	0	1	-	1
1	0	1	1	0	-	0
1	1	0	1	0	-	0
1	1	1	1	1	-	0

Minimierte Ausgangs- und Ansteuernetze

j^t

— y —

0	0	X	X	
0	1	X	X	x

— ü —

k^t

— y —

X	X	0	1	
X	X	0	0	x

— ü —

s^t

— y —

0	1	0	1	
1	0	1	0	x

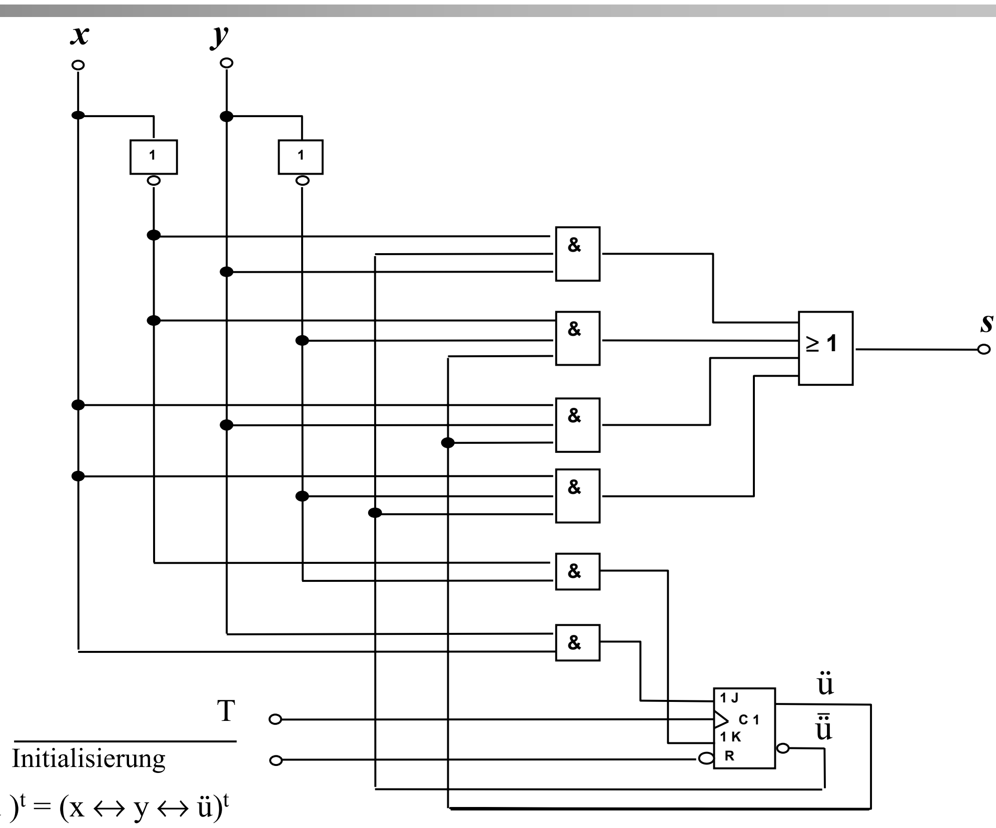
— ü —

$$j^t = (x\ y)^t$$

$$k^t = (\bar{x}\ \bar{y})^t$$

$$s^t = (x\ \bar{y}\ \bar{ü} \vee \bar{x}\ y\ \bar{ü} \vee x\ y\ ü \vee \bar{x}\ \bar{y}\ ü)^t = (x \leftrightarrow y \leftrightarrow ü)^t$$

Realisierung des Serienaddierers



$$j^t = (x\ y)^t$$

$$k^t = (\bar{x}\ \bar{y})^t$$

$$s^t = (x\ \bar{y}\ \bar{ü} \vee \bar{x}\ y\ \bar{ü} \vee x\ y\ ü \vee \bar{x}\ \bar{y}\ ü)^t = (x \leftrightarrow y \leftrightarrow ü)^t$$

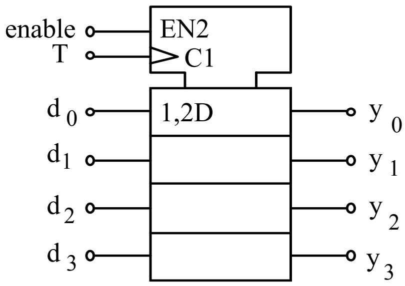
Modul 4: Schaltwerke

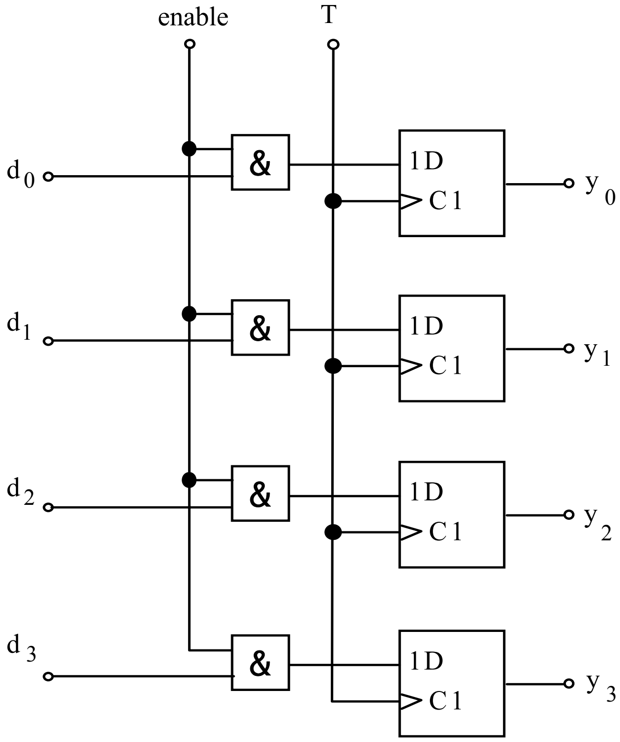
- Formale Grundlagen
  - Endliche Automaten
- Asynchrone Schaltwerke, Flip-Flops
- Synchrone Schaltwerke
- Spezielle Schaltwerke
  - Register, Zähler, Schieberegister

Register

- Lineare Anordnung von Flipflops zur Speicherung mehrerer Bits (Bitvektor).
- Die Flipflops werden mit einem gemeinsamen Takt angesteuert.
- Einfachstes Register:
  - Unverkoppelt nebeneinandergesetzte D-Flipflops.
- Im allgemeinen werden die Flipflops durch zusätzliche gemeinsame Steuersignale beeinflußt.

4-Bit-Register aus D-Flipflops mit Freigabesignal

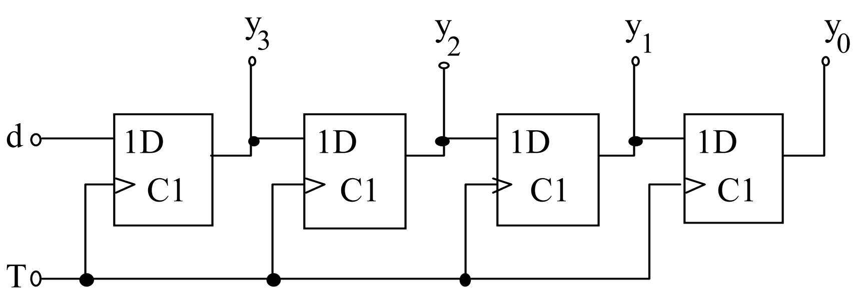


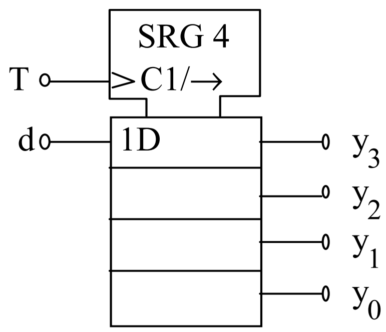


Nur wenn "enable" = 1 ist, werden die Daten übernommen.

Schieberegister (1)

- Kette von in Reihe geschalteten Registern oder D-Flipflops
- Der Ausgang eines Speicherelements ist jeweils mit dem Eingang des nächsten verbunden.

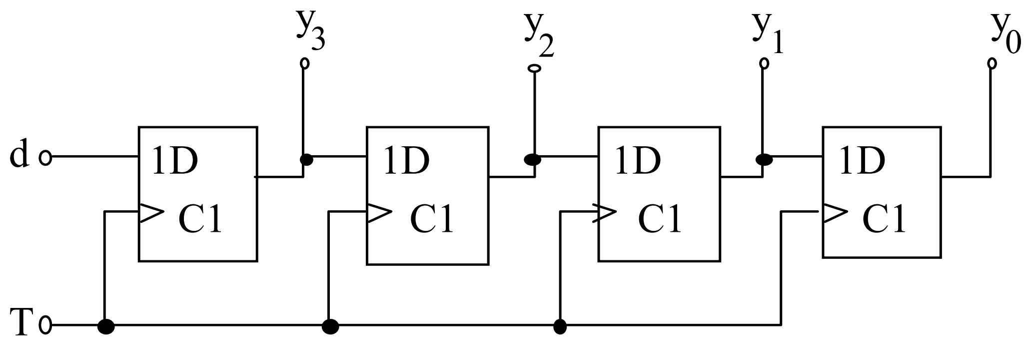




Schieberegister (2)

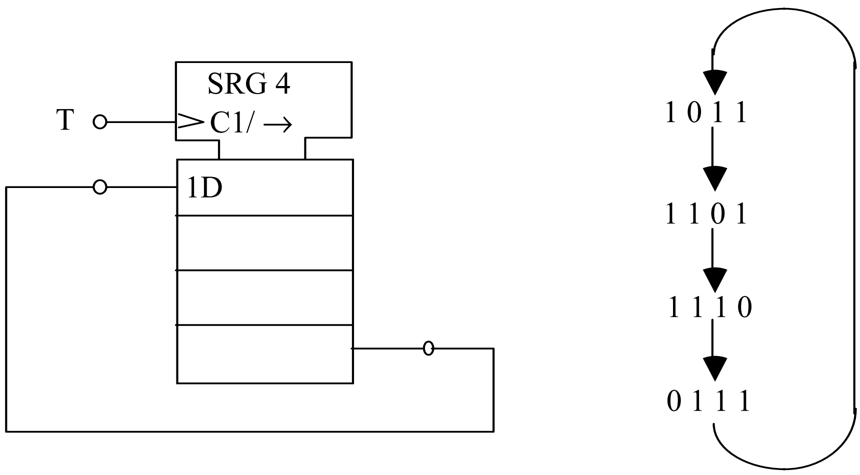
- Division/Multiplikation durch/mit 2

- Interpretiert man die Bitfolge $y_3\ y_2\ y_1\ y_0$ als Dualzahl, entspricht ein Rechtsschieben (mit d = 0) einer Division durch 2 (ohne Rest).
- Schiebt man die Bitfolge ein Bit nach links (mit 0 als neuem letztem Bit), erhält man eine Multiplikation mit 2.



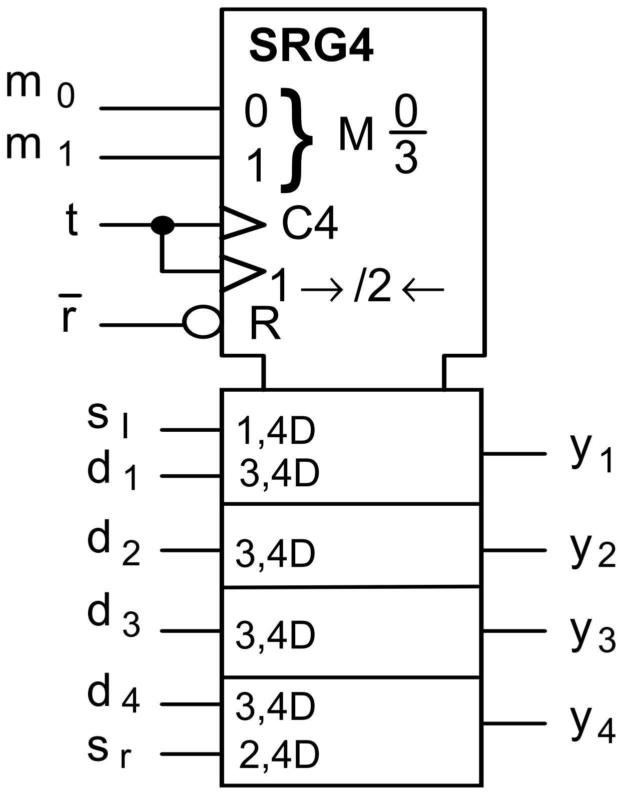
Schieberegister (3)

- Umlaufspeicher/Ringzähler
  Verbindet man den seriellen Ausgang eines Schieberegisters mit seinem seriellen Eingang, erhält man einen Umlaufspeicher (Ringzähler), der eine Bitfolge beliebig lange zwischenspeichern kann und dabei im Kreise schiebt.



Umlaufspeicher mit Beispiel für umlaufenden Speicherinhalt

Konkreter Baustein: 74LS194 (1)



Funktionsmodi

Daten, die an den 4 Paralleleingängen $\mathbf{d_1}$ bis $\mathbf{d_4}$ anliegen, werden parallel geladen.

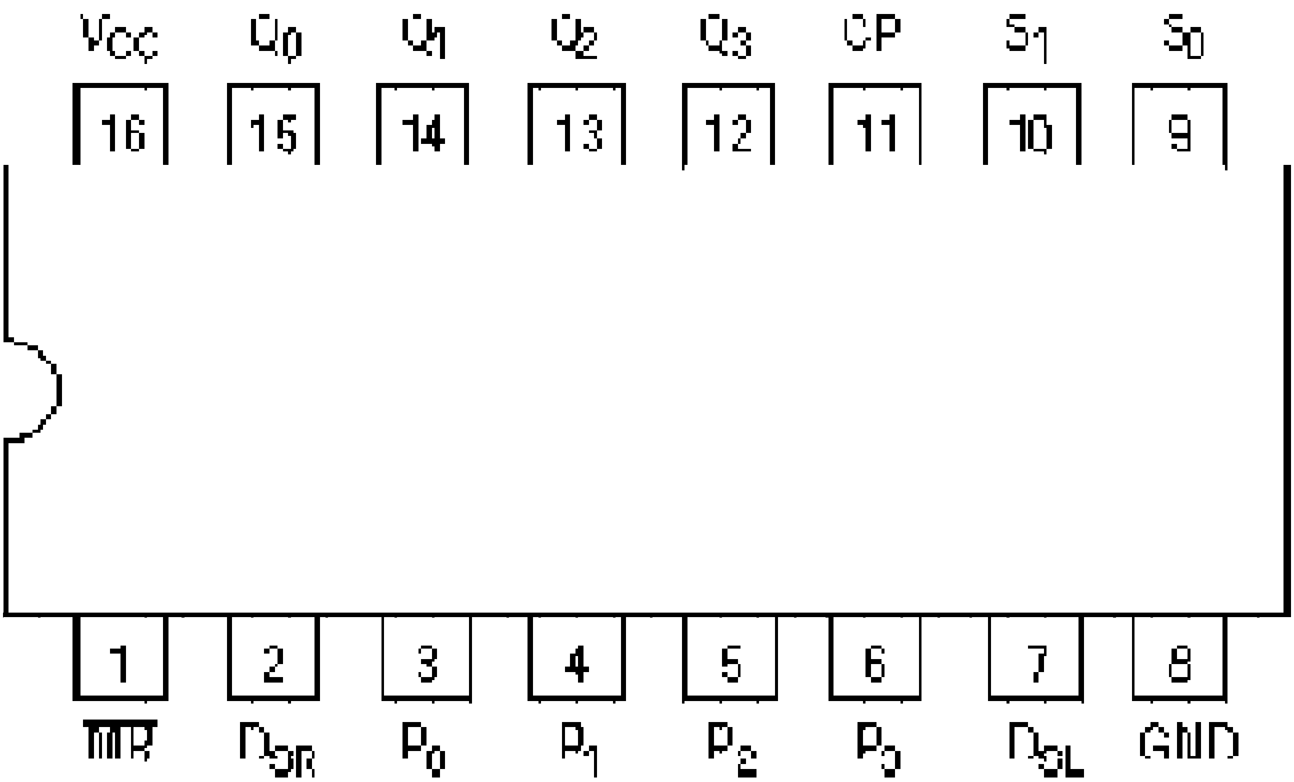
Daten am Eingang werden $\mathbf{s_l}$ seriell übernommen und die alten Registerinhalte nach unten weitergeschoben.

Daten am Eingang $\mathbf{s_r}$ werden seriell übernommen und alte Registerinhalte nach oben weitergeschoben.

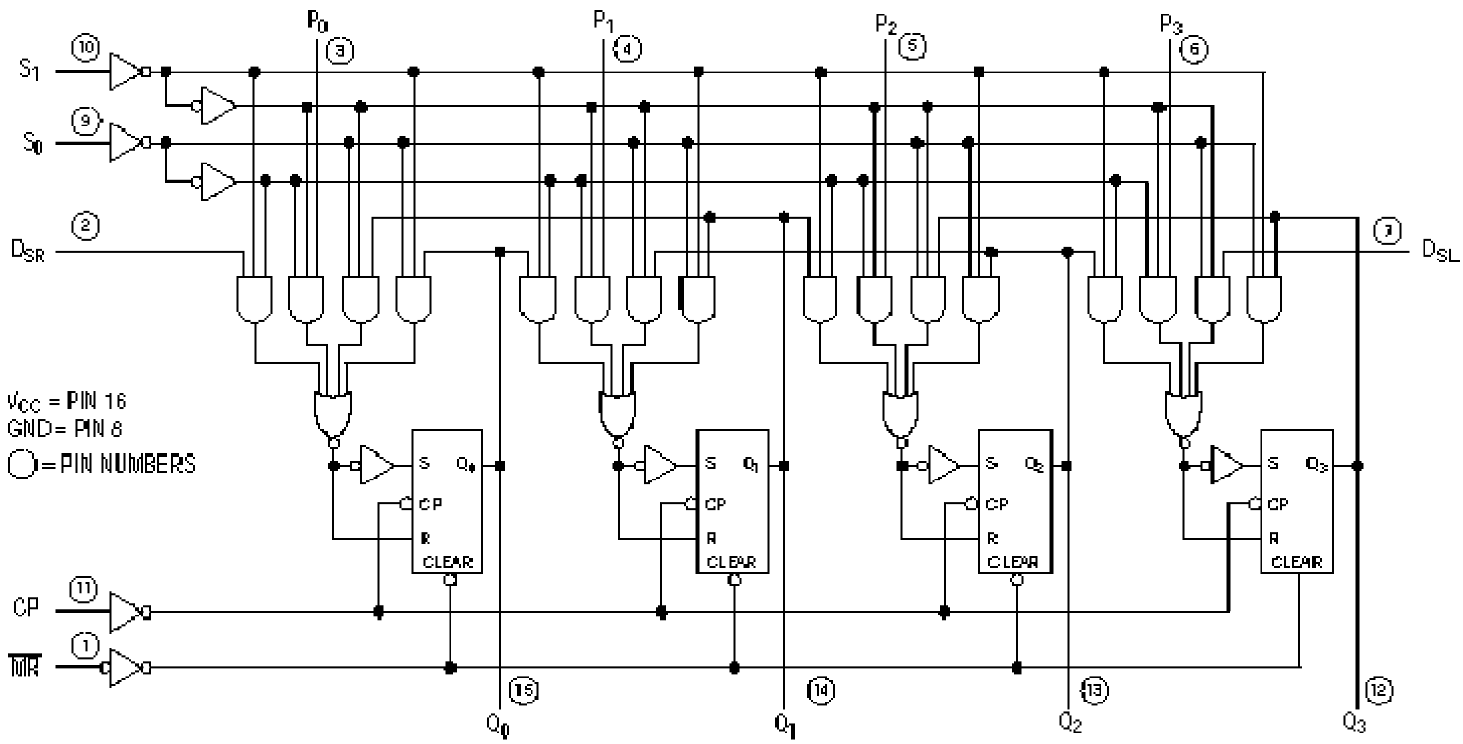
Die aktuellen Registerinhalte werden über mehrere Takte gespeichert.

Mit dem Rücksetz-Eingang $\overline{\mathbf{r}}$ kann man das Schieberegister auch asynchron rücksetzen.

Konkreter Baustein: 74LS194 (2)



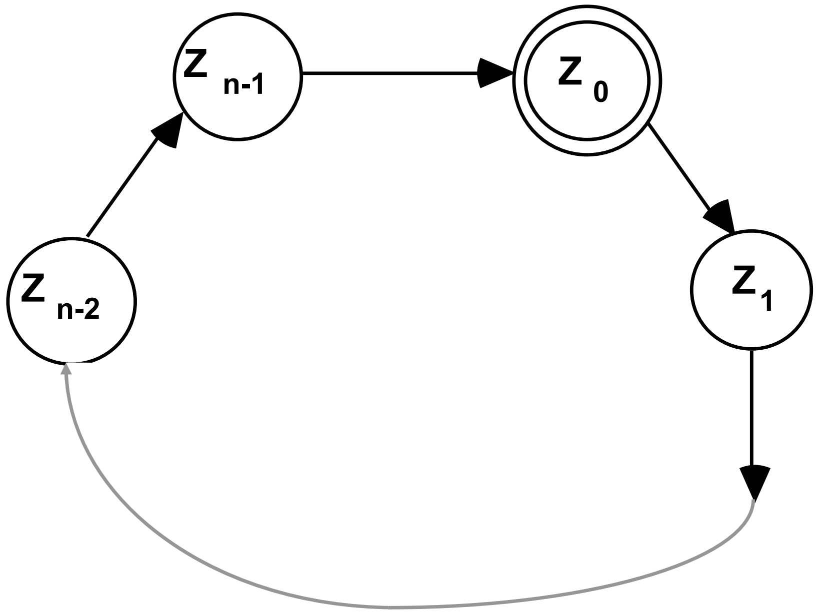
Konkreter Baustein: 74LS194 (3)



Zähler (1)

- Zähler erfüllen in digitalen Systemen mehrere Aufgaben:
- Man kann Impulse abzählen.
- Man kann aufeinanderfolgende Adressen eines Speichers adressieren (z.B. bei Programmzählern) oder aufeinanderfolgende Arbeitsschritte kontrollieren (bei Steuerwerken).
- Eine vorgegebene Impulsfolge läßt sich in der Frequenz reduzieren, der Zähler wirkt als Frequenzteiler. Dabei macht man sich die Tatsache zunutze, dass sich das Bit i einer Zahl $z_n \ldots z_i \ldots z_0$ nur 2^{-i} mal so oft ändert wie Bit 0, wenn diese Zahl fortlaufend inkrementiert wird.

Zähler (2)



Grundlegendes Übergangsdiagramm

Anmerkungen

- Durch Modifizierung der Grundstruktur:

➔ Setz- oder Rücksetzeingänge, Freigabeeingang oder eine Möglichkeit, vorwärts und rückwärts zu zählen.

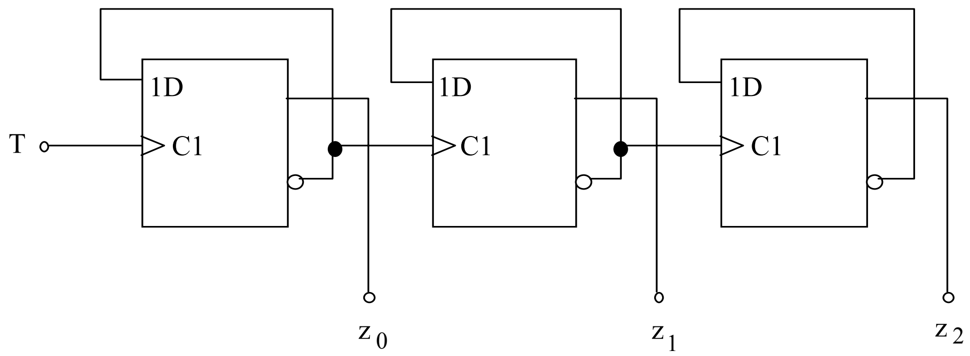
- Sind die Zustände dual kodiert (Zustand Z_i wird mit der Dualzahl i kodiert), liegt ein Dualzähler vor.

- Abhängig von der Länge n des Zählzyklus wird ein Zähler als Modulo-n-Zähler bezeichnet.

- Ist $n = 2^m$, so handelt es sich um einen m-stelligen Zähler.

Asynchrone Zähler

- Die Größe des Ansteuerschaltnetzes wächst mit zunehmender Bitanzahl stark an.
- Aus diesem Grund sind asynchrone Zähler (engl.: ripple counter) attraktiv.



Realisierung eines asynchronen 3-stelligen Dualzählers

Nachteile asynchroner Zähler

- Diese Schaltung ist langsamer, da jedes Flipflop erst dann reagiert, wenn das vorhergehende Flipflop von 1 nach 0 umgeschaltet hat.
- Außerdem ändern sich die Ausgänge der Schaltung nicht gleichzeitig.