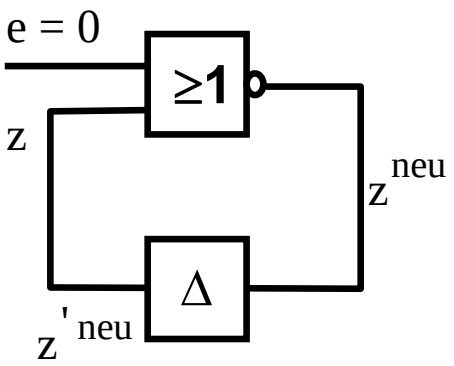
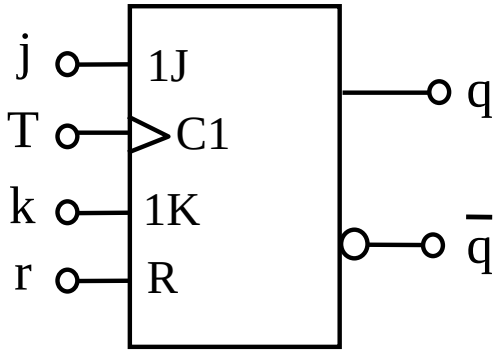
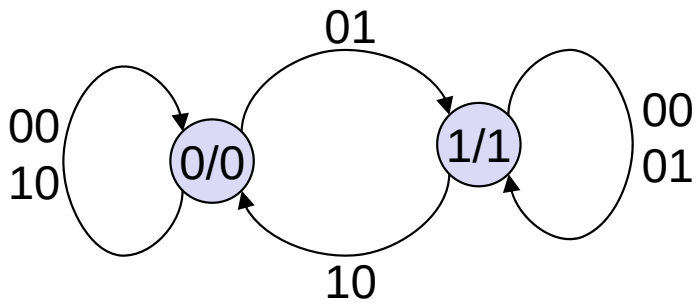
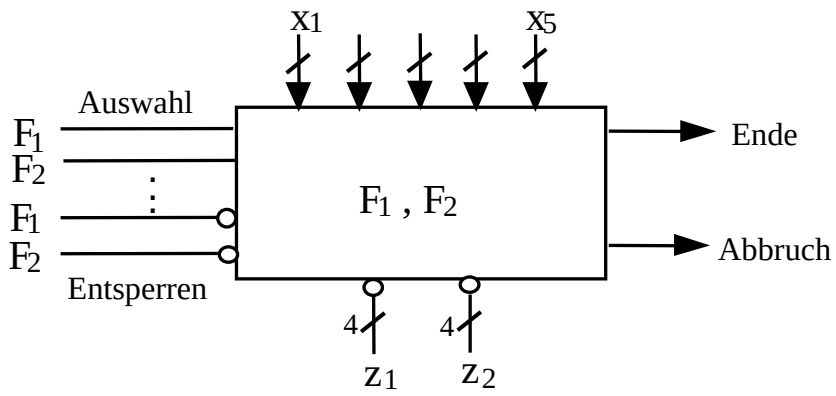


Kapitel 12: Schaltwerke





- Formale Grundlagen
  - Endliche Automaten
- Asynchrone Schaltwerke, Flip-Flops
- Synchrone Schaltwerke
- Register-Transfer-Ebene
- 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!

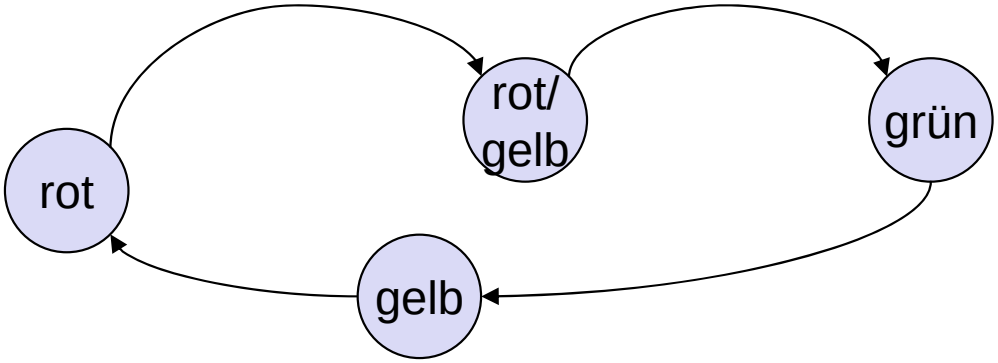
- Schaltwerke (sequentielle Schaltungen):
  - Die Ausgabewerte hängen auch von Belegungen der Eingangsvariablen zu vergangenen Zeitpunkten ab.

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

Freie Universität Berlin

- Ein 6-Tupel $M = (E, A, Z, \delta, \omega, z_0)$ 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.

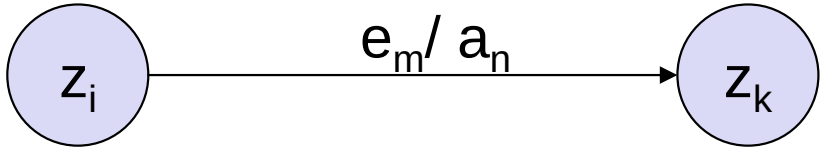
- Ausgabefunktion $\omega: Z \times E \rightarrow A$

  ω eine auf der Menge $Z \times E$ definierte Funktion, deren Werte in A liegen.

- Grundzustand z_0

Beispiel

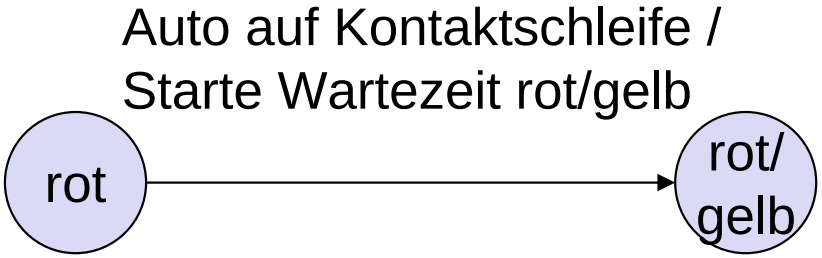
- Allgemein



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

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

- 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

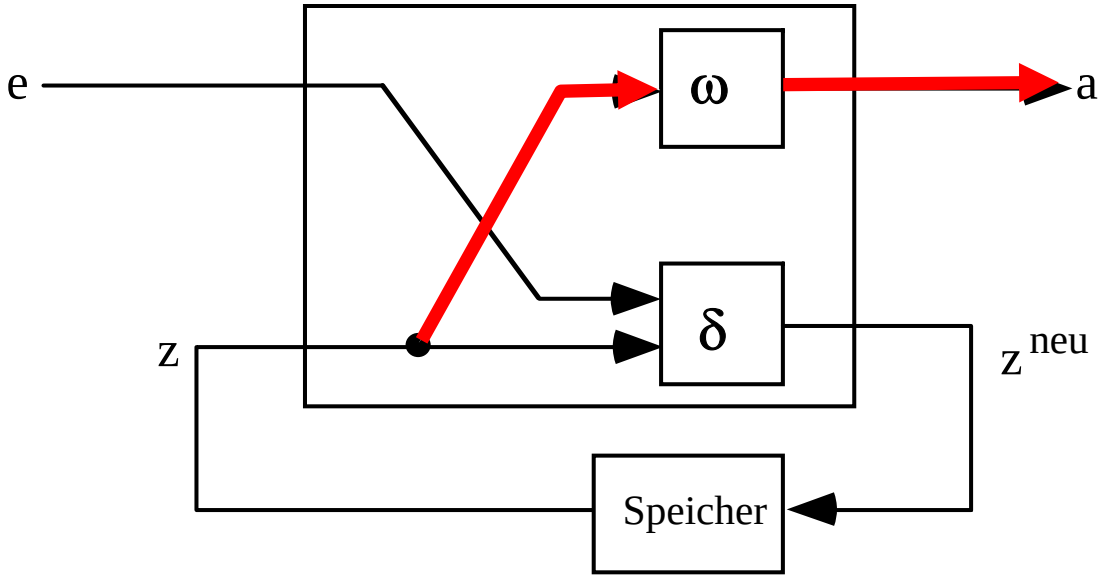
- Die Zustandsmenge Z ermöglicht die Speicherung von Wissen über Eingangsbelegungen der Vergangenheit.

- 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

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

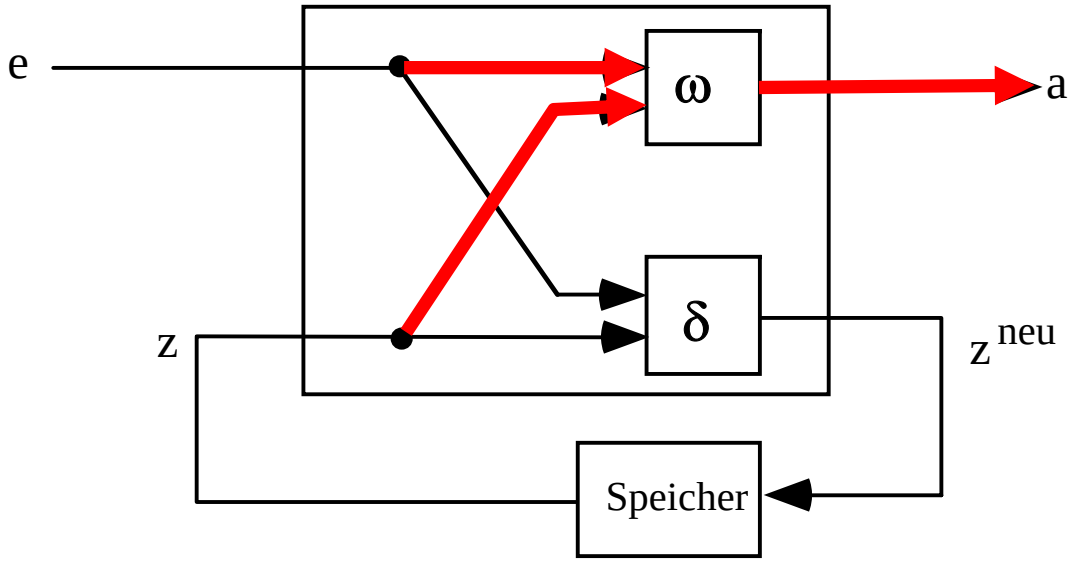
- $a = \omega(z)$ $\quad z^{neu} = \delta(z,e)$
- $a \in A;$ $\quad e \in E;$ $\quad z, z^{neu}$ Z



Mealy- und Moore-Automat

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

- $a = \omega(z,e)$ $\quad z^{neu} = \delta(z,e)$
- $e \in E;$ $\quad a \in A;$ $\quad z, z^{neu} \in Z$

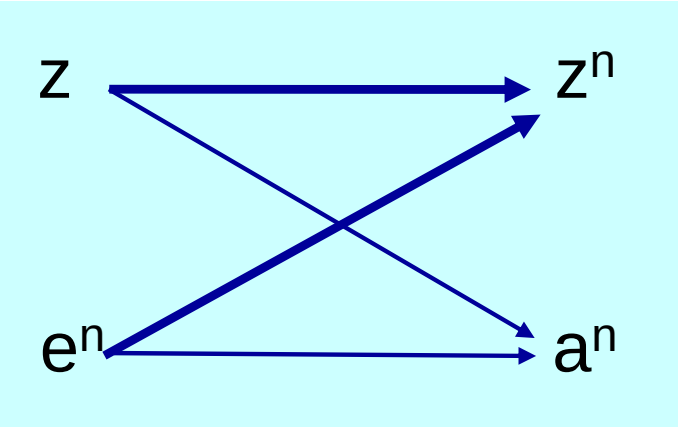


Unterschied im Verhalten der beiden Automaten

- 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.

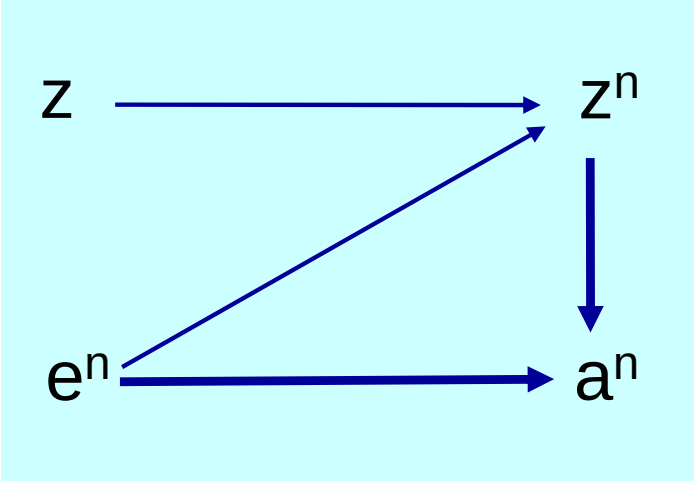
- Anwendung: synchrone Schaltwerke



Unterschied im Verhalten der beiden Automaten

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

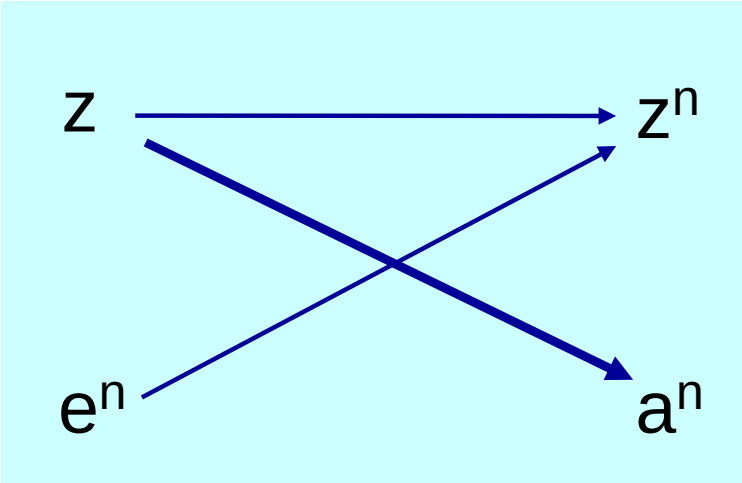
- Anwendung: asynchrone Schaltwerke



Unterschied im Verhalten der beiden Automaten

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

- Anwendung: synchrone und asynchrone Schaltwerke



Darstellungsmöglichkeiten

- 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

- 4 Möglichkeiten zur formalisierten Beschreibung des Verhaltens eines Automaten sollen an einem Beispiel demonstriert werden

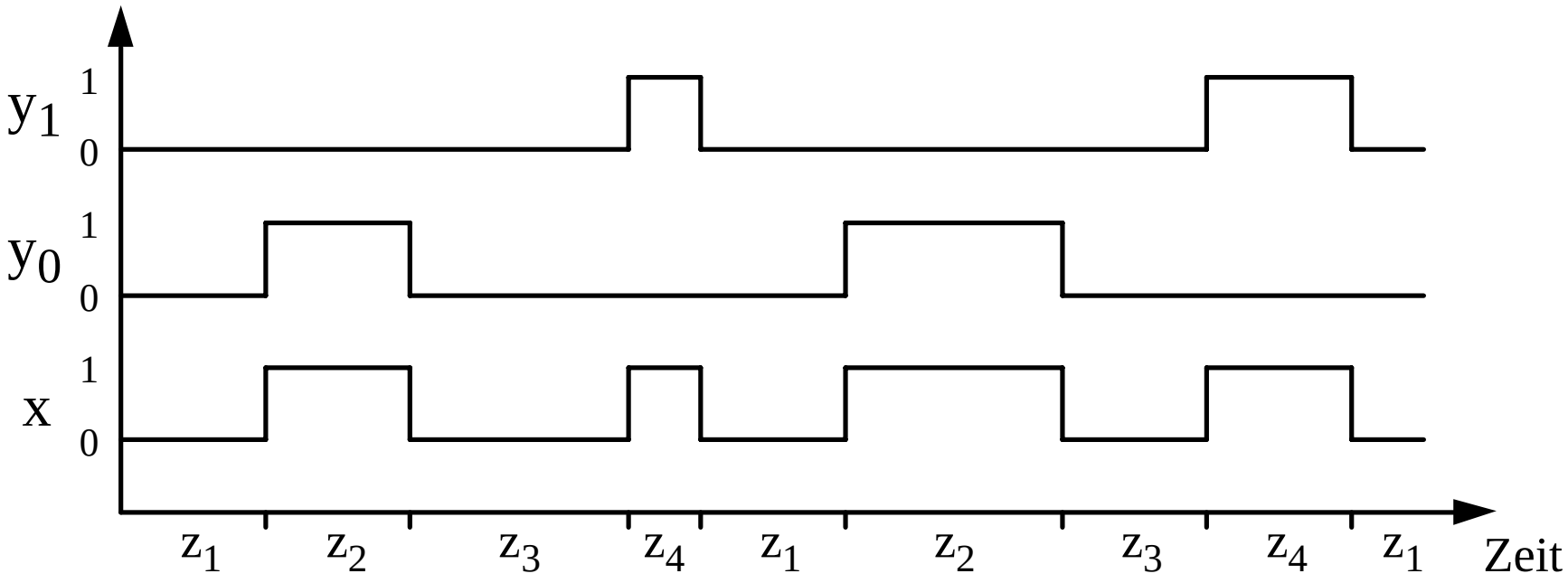
- Beispiel:

- 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.

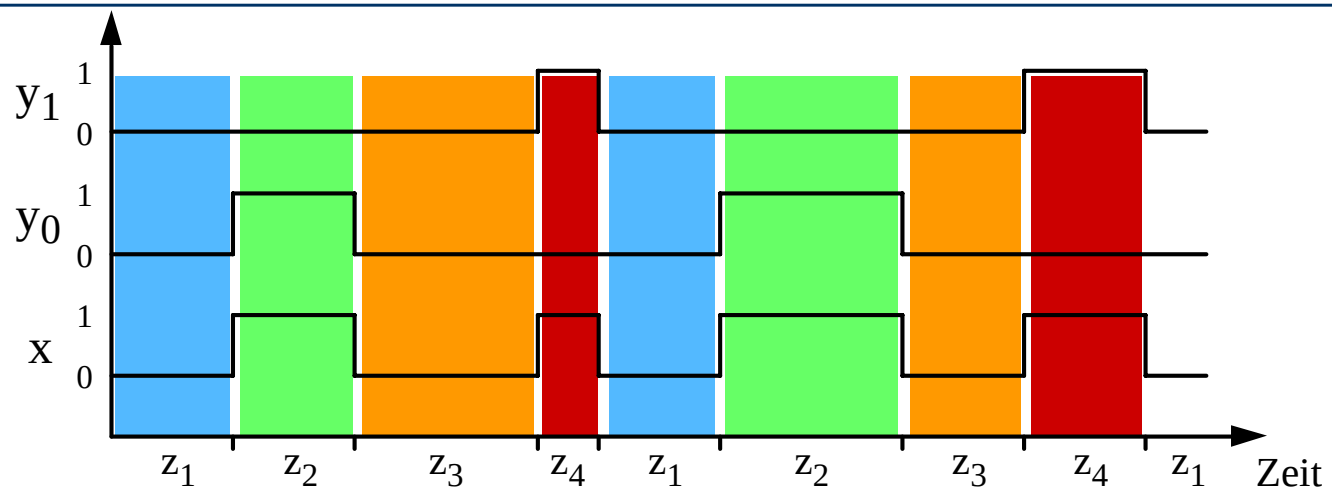
- Gesucht ist eine formalisierte Beschreibung dieses Problems.

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

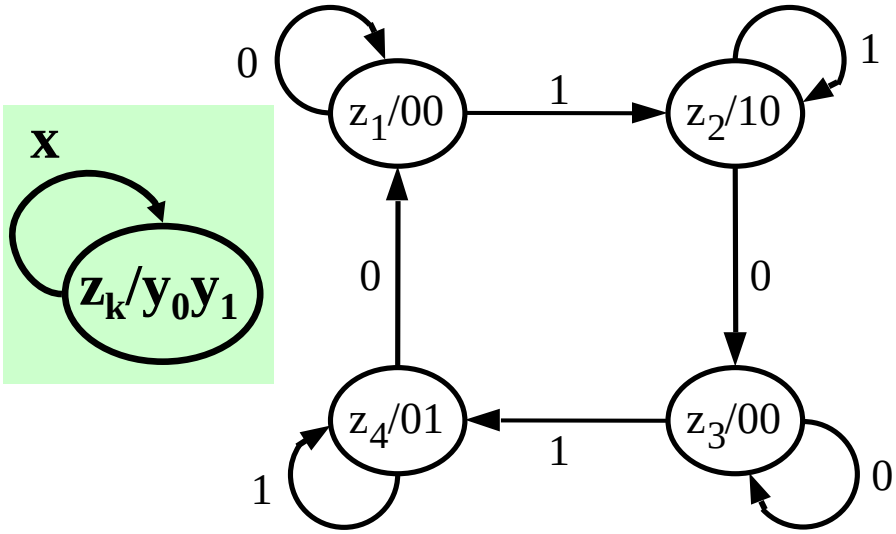
z_k	$z_{k+1}/\,y_0\,y_1$ x=0	$z_{k+1}/\,y_0\,y_1$ x=1
z_1	$z_1/00$	$z_2/10$
z_2	$z_3/00$	$z_2/10$
z_3	$z_3/00$	$z_4/01$
z_4	$z_1/00$	$z_4/01$

Mealy-Automat

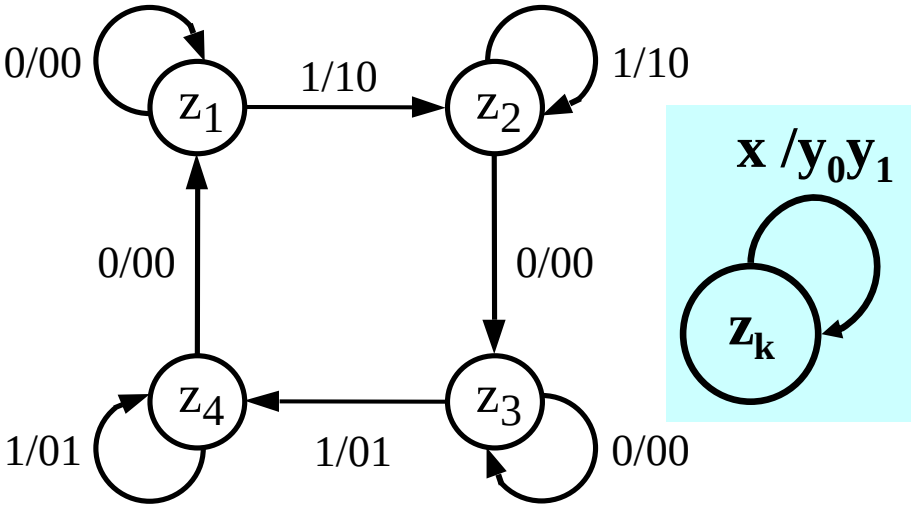
4. Automatengraph

- 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

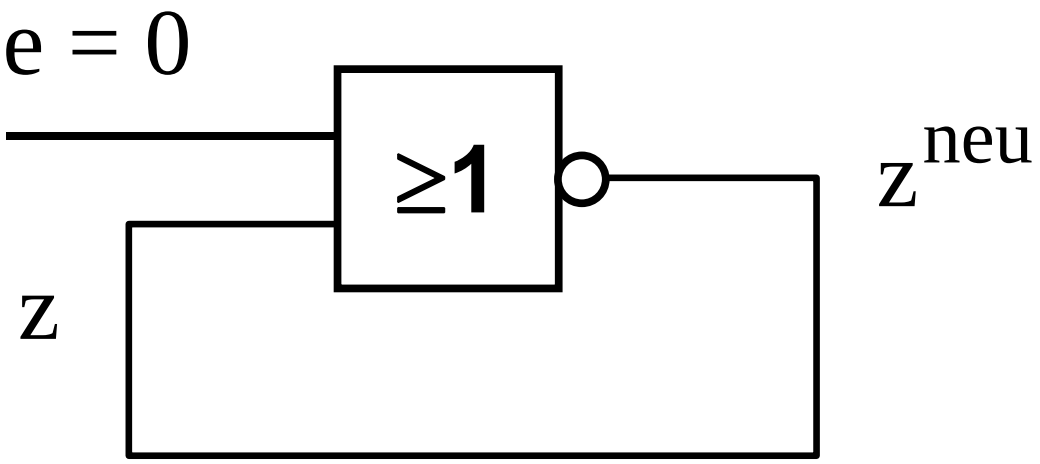
- Zur Speicherung vergangener Informationen ist ein Zustandsspeicher erforderlich.

- Einfachste Form dieses Zustandsspeichers:

Rückkopplung

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

Beispiel: Rückgekoppeltes NOR-Gatter



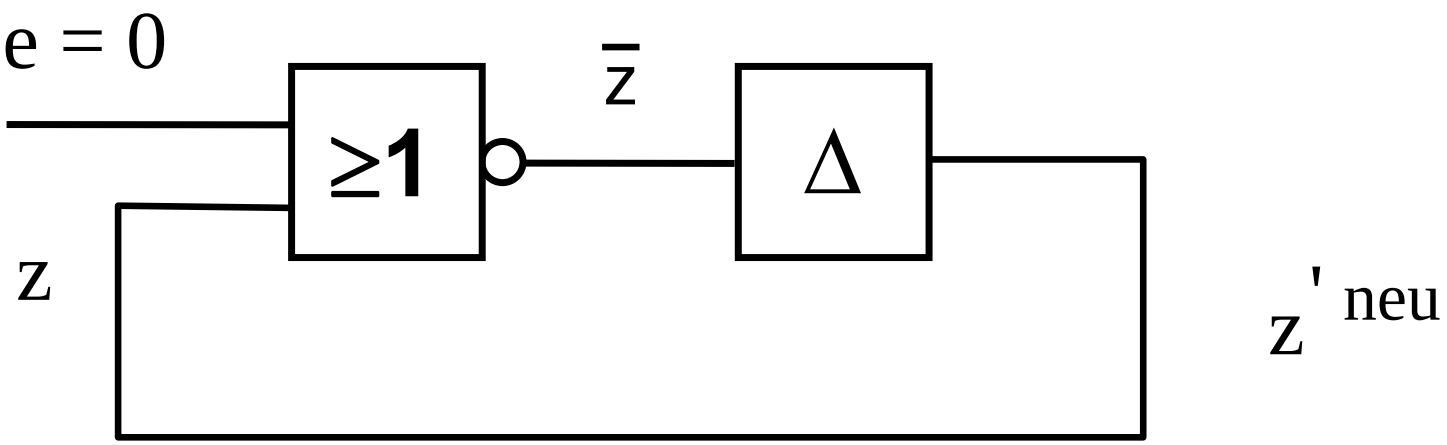
Als ideales Gatter betrachtet ist die Schaltung unzulässig, denn es müsste gleichzeitig gelten:

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

$$z = z^{neu}$$

Rückgekoppeltes NOR-Gatter

In der Realität hat jede Schaltung hat eine Verzögerungs-zeit größer 0 (Totzeitmodell).



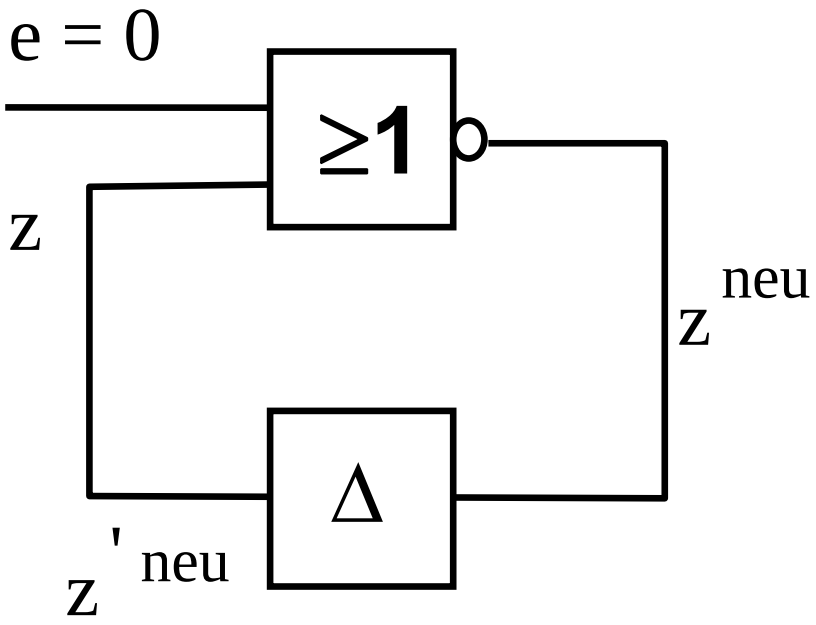
Mit dieser Verzögerung erhält man:

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

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

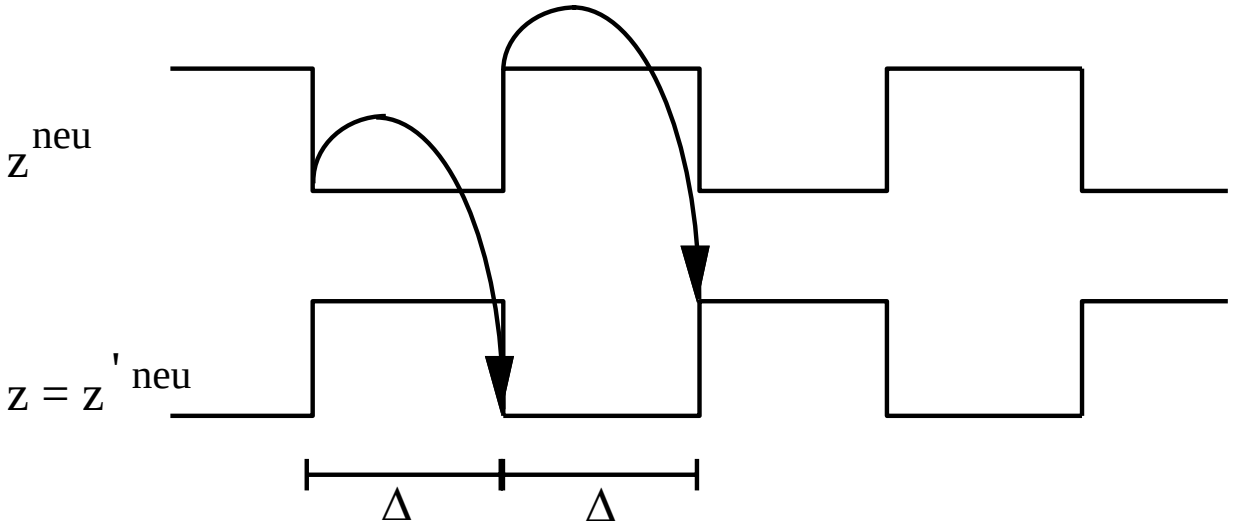
Rückgekoppeltes NOR-Gatter

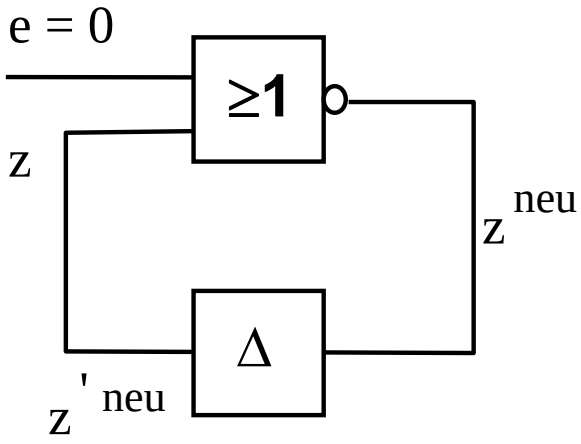
Zeichnet man die Schaltung etwas anders, so sieht man, dass das **Δ-Verzögerungsglied** dem **Speicher** entspricht.



Rückgekoppeltes NOR-Gatter

Das Zeitverhalten der Schaltung im Zeitdiagramm:





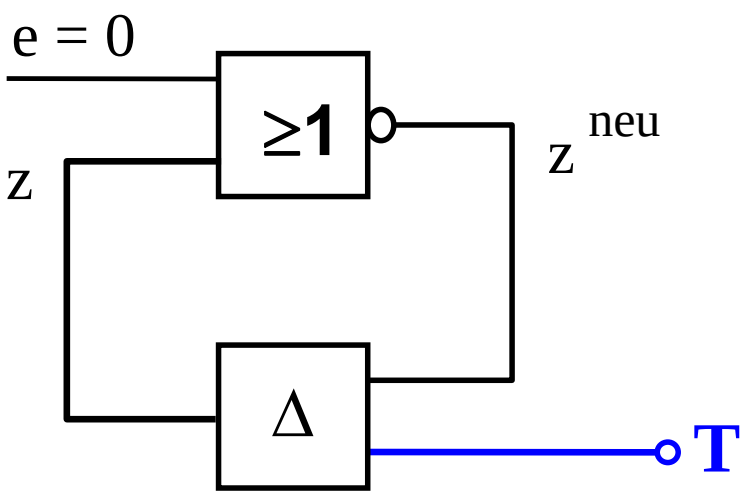
Das Verhalten der Schaltung ist stark abhängig von den Verzögerungszeiten.

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.

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.

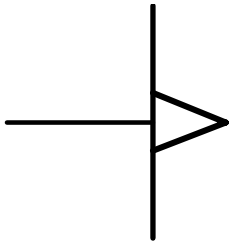
Flankensteuerung

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

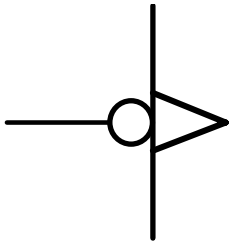
Vorteil:

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



01-Übergang



10-Übergang

Schaltsymbol für einen flankengesteuerten Takteingang

Synchrone Schaltwerke vs. Asynchrone Schaltwerke

Synchrone Schaltwerke:

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

Vorteil:

leichter zu analysieren und entwerfen als asynchrone Schaltwerke.

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

Synchrone Schaltwerke vs. Asynchrone Schaltwerke

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

Freie Universität Berlin

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 ?

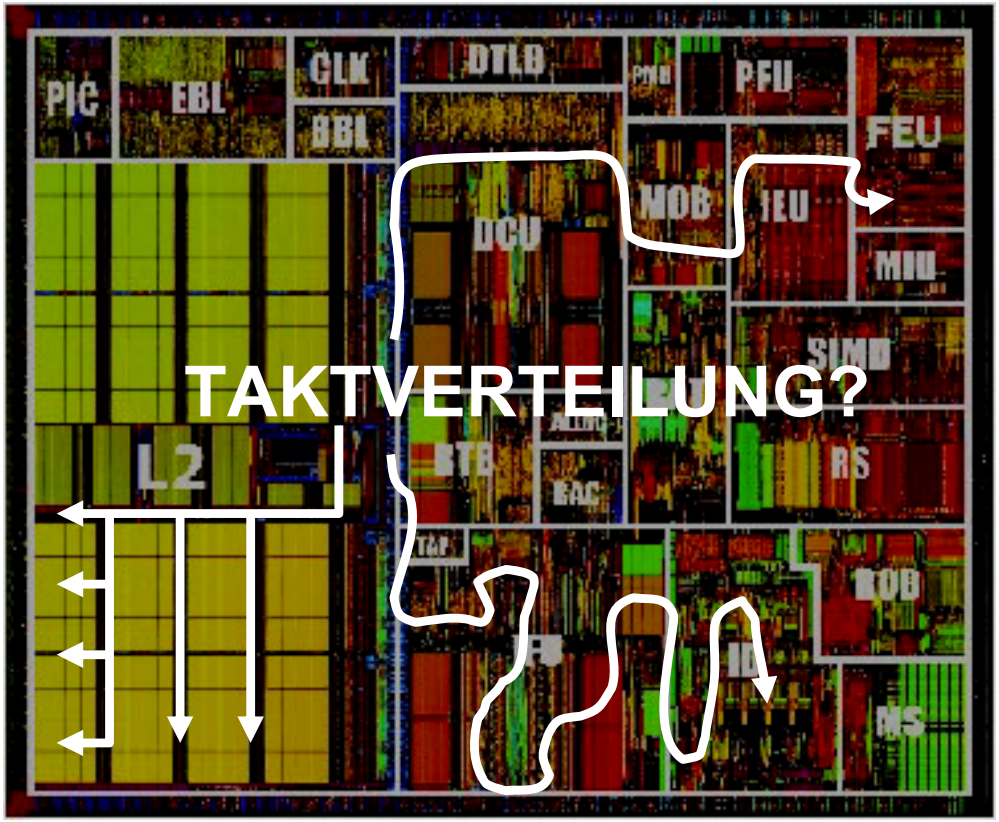
Synchrone Schaltwerke vs. Asynchrone Schaltwerke

Freie Universität Berlin

Begründung:

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

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

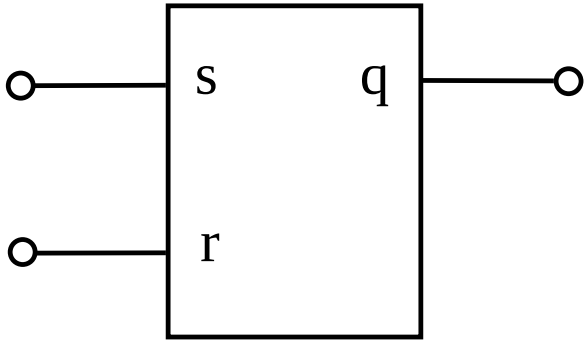


1 GHz Takt = 1 ns Taktdauer!

Entwurf asynchroner Schaltwerke
Beispiel RS-FF

Die Vorgehensweise und Problematik beim Entwurf asynchroner Schaltwerke soll an einem Beispiel erläutert werden.

Zu entwerfen sei ein einfacher Speicher mit folgendem Verhalten:



Eingang r soll den Speicher zurücksetzen (reset), d.h. wenn Eingang r zu 1 wird, wird Ausgang q zu 0.

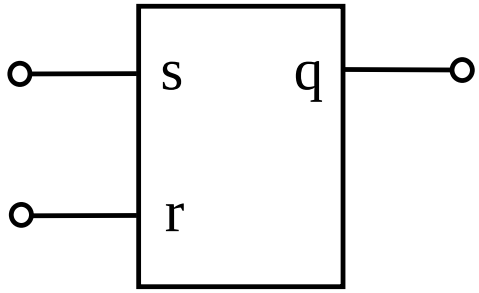
Eingang s soll den Speicher setzen (set), d.h. wenn Eingang s zu 1 wird, wird Ausgang q zu 1.

Sind r und s beide 0, so soll der aktuelle Wert gespeichert werden, d.h. q seinen letzten Wert behalten.

Aufstellen des Automatengraphen
Beispiel RS-FF

- Vorgehensweise:
  - Festlegung der Ein-/Ausgabevariablen
  - Festlegung der Zustände
  - Zeichnen des Automatengraphen

Im Beispiel:



Eingangsvariable: **r, s**

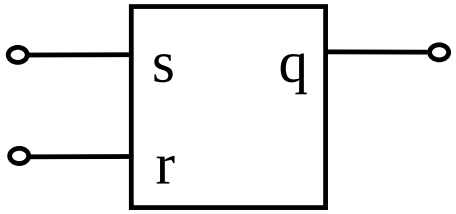
Ausgangsvariable: **q**

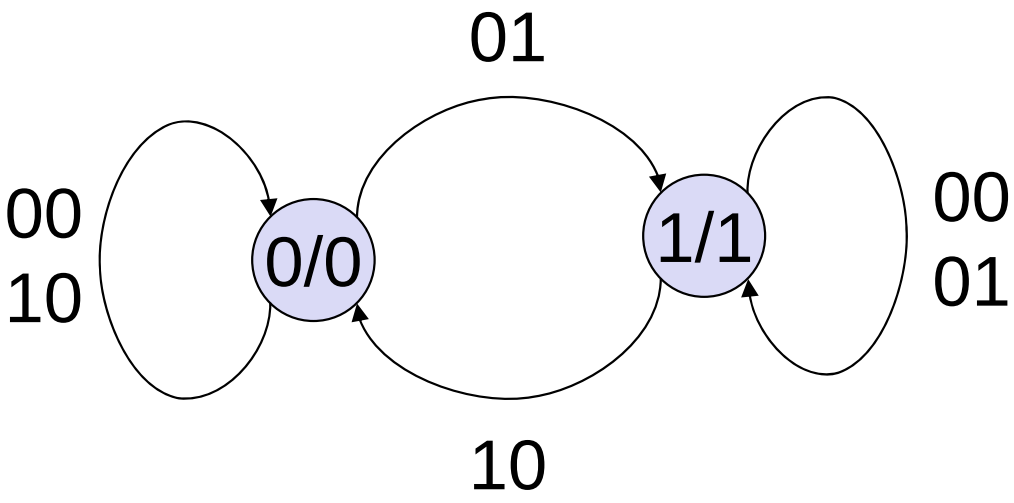
benötigte Zustände: **2**
(zum Speichern zweier verschiedener Werte)

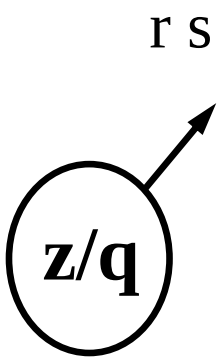
Automatengraph des einfachen Speichers Beispiel RS-FF

- Es wurde hier ein Moore-Schaltwerk entworfen, die Ausgabe q ist nur vom Zustand z abhängig.

- Die Ausgabe wurde deshalb ebenfalls in die Knoten des Automatengraphen eingetragen.







Funktionsweise
Beispiel RS-FF

In Zustand 0 hat auch q den Wert 0.

Dort verbleibt das Schaltwerk, solange s den Wert 0 hat.

Wird s zu 1, so wechselt das Schaltwerk in den Zustand 1 mit der Ausgabe von q = 1.

Dort verbleibt das Schaltwerk, solange r den Wert 0 hat.

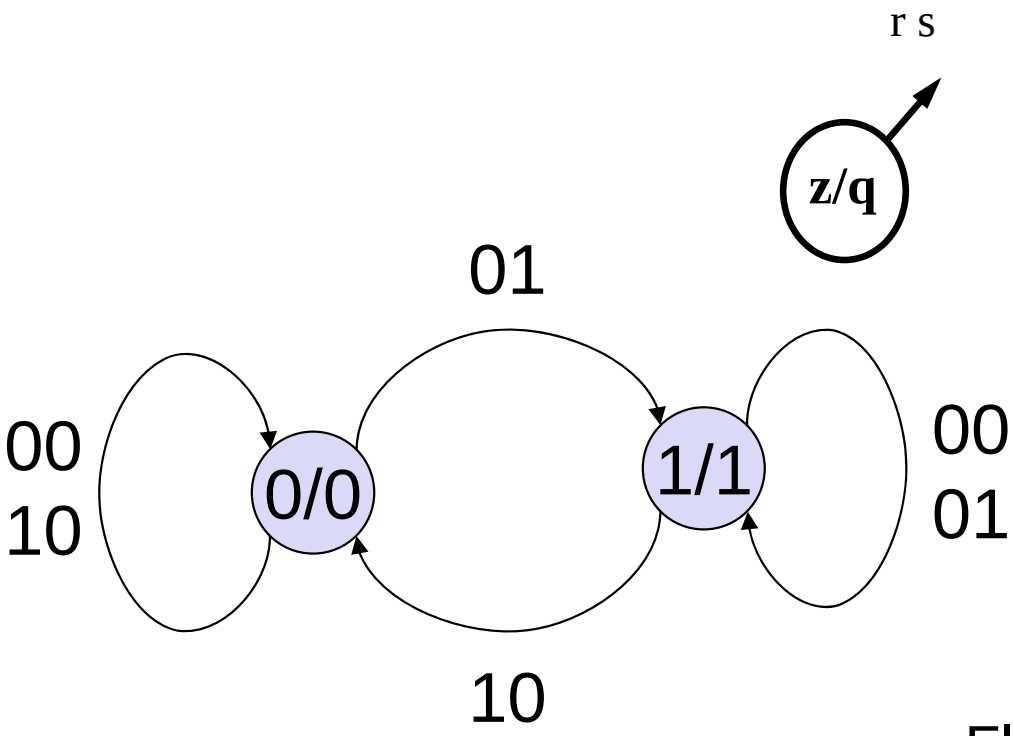
Wird r zu 1, so wechselt das Schaltwerk wieder in den Zustand 0.

Der Fall r und s gleichzeitig 1 wird in der Aufgabenstellung nicht beschrieben.

Er wurde deshalb auch nicht in den Automatengraphen übernommen und kann so im weiteren Entwurf zu Optimierungszwecken frei verfügt werden.

Aufstellen der Automatentafel
Beispiel RS-FF

- Nächster Schritt: Übertragen des Automatengraphen in die Automatentafel



z	z^+				q
	r s =				
	00	01	10	11	
0	0	1	0	-	0
1	1	1	0	-	1

Flussmatrix des einfachen Speichers

Bei asynchronen Schaltwerken heißt die Automatentafel auch **Flussmatrix**.

Stabilitätskriterium

- In asynchronen Schaltwerken sind nur solche Zustände stabil, in denen der aktuelle Zustand unter einer Eingabebelegung gleich dem Folgezustand ist.

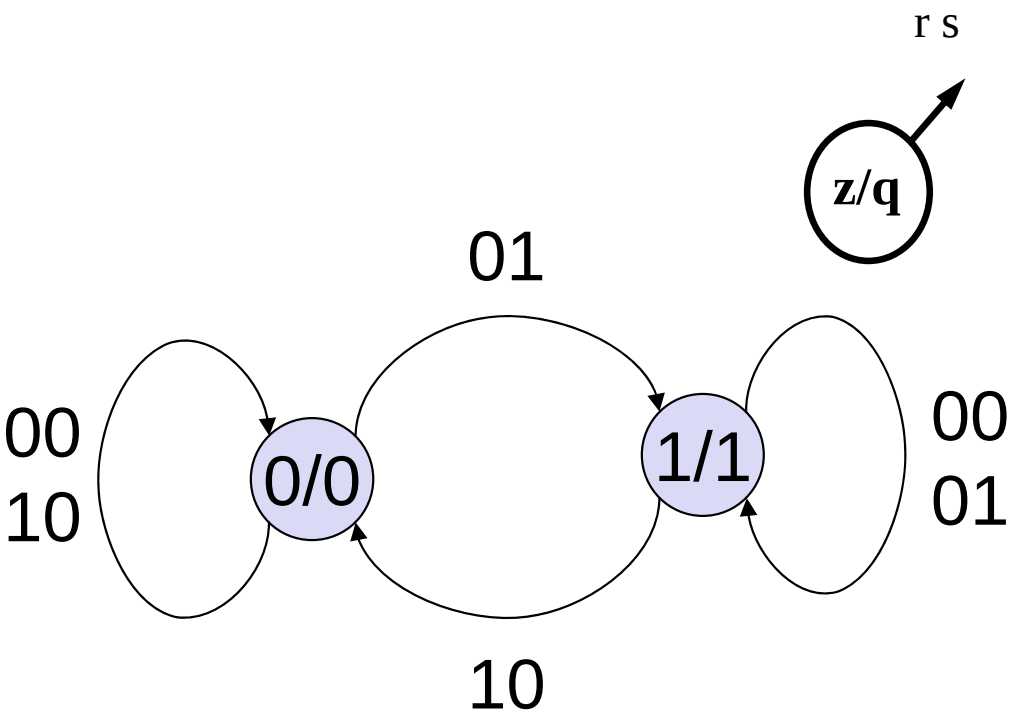
- Stabilitätskriterium:

□ $\delta(z_{stabil},e) = \delta(\delta(z_{stabil},e),e)$

- Anderenfalls ist der Zustand instabil, es erfolgt ein sofortiger weiterer Zustandsübergang.

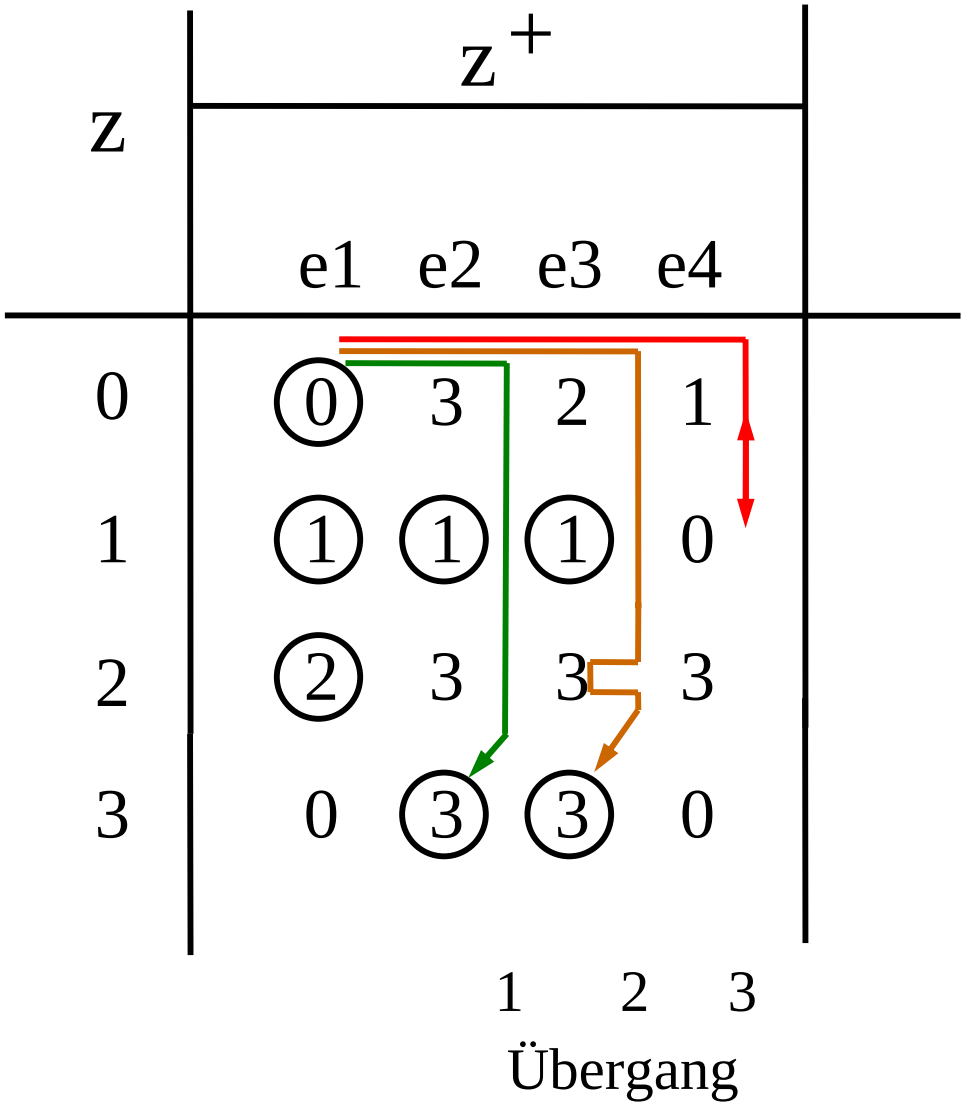
- Stabile Zustände werden in der Flussmatrix durch einen Kreis gekennzeichnet.

Beispiel: Flussmatrix des einfachen Speichers



z	z^+				q
	r s =				
	00	01	10	11	
0	0	1	0	-	0
1	1	1	0	-	1

Mögliche Formen eines Zustandsübergangs



Direkter Übergang

Indirekter Übergang

Oszillation

Mögliche Formen eines Zustandsübergangs

Freie Universität Berlin

Direkter Übergang:

Ein stabiler Zustand geht direkt in einen stabilen Folgezustand über (Übergang 1).

Indirekter Übergang:

Der stabile Folgezustand stellt sich über mehrere instabile Zwischenzustände ein (Übergang 2).

Oszillation:

Es stellt sich überhaupt kein stabiler Folgezustand ein (Übergang 3).

Wahl der Zustandskodierung

Bisher: Zustände wurden mit symbolischen Namen bezeichnet.

Zur digitalen Speicherung dieser Zustände:

- Die symbolischen Zustandsnamen müssen eindeutig auf eine binäre Zustandskodierung abgebildet werden.

Mit n Bit können maximal 2^n verschiedene Kodierungen gebildet werden.

- zwischen der Anzahl n der Zustandsbits und der Zahl k der Zustände existiert die Beziehung:

$$\mathbf{n \geq ld\ k} \qquad n, k \in \mathbb{N}$$

Die Anzahl der Zustände legt somit eine untere Schranke für die Anzahl der Zustandsbits fest.

Zustandsvariable / Zustandskodierung

- Die Zustandsbits werden auch als Zustandsvariable bezeichnet.
- minimale Anzahl der Zustandsvariablen bei k Zuständen:
  - $\lceil \mathrm{ld}\ k \rceil$
- Bei asynchronen Schaltwerken entspricht jede Zustandsvariable einer Rückkopplung.
- Bei asynchronen Schaltwerken kommt der Wahl einer geeigneten Zustandskodierung entscheidende Bedeutung zu!

Beispiel

Bereits bekannte Flussmatrix:

z	z^+			
	e1	e2	e3	e4
0	0	3	2	1
1	1	1	1	0
2	2	3	3	3
3	0	3	3	0
	1	2	3	
	Übergang			

Willkürlich gewählte Zustandskodierung:

z	$ü_2$	$ü_1$
0	0	0
1	0	1
2	1	0
3	1	1

Übergang 1

- Direkter Übergang von Zustand 0 unter Anlegen von e2 nach Zustand 3

- Bei der gewählten Zustandskodierung bedeutet dies den Übergang der Zustandsvariablen:

  - $(ü_2, ü_1)$: $(0, 0) \rightarrow (1, 1)$ Beide Zustandsvariablen ändern sich !

- Dies geschieht mit großer Sicherheit nicht gleichzeitig (unterschiedliche Gatterlaufzeiten) $\Rightarrow$ Es entsteht kurzfristig ein Zwischenwert:
  - Fall 1: $(0, 0) \rightarrow (0, 1) \rightarrow (1, 1)$ $ü_1$ ist schneller
  - Fall 2: $(0, 0) \rightarrow (1, 0) \rightarrow (1, 1)$ $ü_2$ ist schneller

- Dies nennt man einen Wettlauf (race)

Mögliche Fälle eines Wettlaufs

- Wir betrachten die möglichen Folgen eines solchen Wettlaufs:
  - Fall 2: $(ü_2, ü_1)$: $(0, 0) \rightarrow (1,0) \rightarrow (1, 1)$

- Der kurzfristig eingenommene Zwischenwert (1, 0) entspricht dem Zustand 2

- Dieser Zustand ist unter der Eingabebelegung e2 instabil und mündet in den Zustand 3 mit der Zustandskodierung (1, 1)

- Wir erreichen unser Ziel.

- Dieser Fall ist offensichtlich unkritisch.

z	z^+			
	e1	e2	e3	e4
0	0	3	2	1
1	1	1	1	0
2	2	3	3	3
3	0	3	3	0

Mögliche Fälle eines Wettlaufs

- Fall 1: $(ü_2, ü_1)$: $(0, 0) \rightarrow (0, 1) \rightarrow (1, 1)$

- Der kurzfristig eingenommene Zwischenwert (0, 1) entspricht dem Zustand 1

- Dieser Zustand ist unter der Eingabebelegung e2 jedoch stabil.

- Der kurzfristig durch einen Wettlauf eingenommene Zwischenzustand bleibt bestehen!

- Zustand 3 wird nicht erreicht.

- Das Schaltwerk funktioniert nicht!

z	z+			
	e1	e2	e3	e4
0	0	3	2	1
1	1	1	1	0
2	2	3	3	3
3	0	3	3	0

Kritischer Wettlauf

- Einen solchen Vorgang, bei dem abhängig von Verzögerungszeiten ein Folgezustand erreicht wird oder nicht, nennt man einen kritischen Wettlauf.

- Vermeidung solcher kritischer Wettläufe:
  - Bei asynchronen Schaltwerken wird meist eine sogenannte wettlauffreie Zustandskodierung angestrebt.

- Erreichen einer wettlauffreien Zustandskodierung:
  - Die Zustände werden derart kodiert, dass sich bei jedem Zustandsübergang nur eine Zustandsvariable ändert.

Beispiel

- Eine wettlauffreie Zustandskodierung für unsere Flussmatrix:

z	$ü_1$	$ü_2$
0	0	0
1	1	1
2	1	0
3	0	1

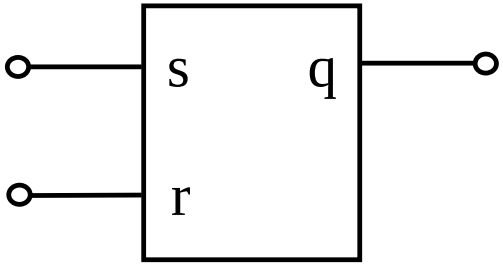
z	z^+			
	e1	e2	e3	e4
0	0	3	2	1
1	1	1	1	0
2	2	3	3	3
3	0	3	3	0

Anmerkungen

- Eine wettlauffreie Zustandskodierung erfordert in der Regel mehr als die minimale Anzahl der Zustandsvariablen.

- Zur systematischen Gewinnung einer solchen Zustandskodierung existieren eine Reihe heuristischer und optimaler Verfahren, die jedoch hier nicht näher behandelt werden können.
  - z.B: Tylor-Verfahren, Liu-Verfahren, ... usw.

- Für unser Beispiel des einfachen Speichers ist eine wettlauffreie Zustandskodierung jedoch problemlos, da nur zwei Zustände vorhanden sind.



z	ü
0	0
1	1

Erzeugen der Ausgabe- und Übergangsschaltnetze

- Setzt man die Zustandskodierung in die Flussmatrix ein, so erhält man die Erregungsmatrix.

- Erregungsmatrix des einfachen Speichers:

z	z^+ r s = 00	01	10	11	q
0	0	1	0	-	0
1	1	1	0	-	1

ü	$ü^+$ r s = 00	01	10	11	q
0	0	1	0	-	0
1	1	1	0	-	1

Nur eine Zustandsvariable: **Erregungsmatrix = Flussmatrix !!**

Aus der Erregungsmatrix lassen sich die Funktionstabellen der Ausgabe- und Übergangs-Schaltnetze ableiten.

Ausgabe- und Übergangsschaltnetze

- Funktionstabelle der Ausgabe- und Übergangsschaltnetze des einfachen Speichers

r	s	ü	$ü^+$	q
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	-	0
1	1	1	-	1

ü	$ü^+$				q
	r s =				
	00	01	10	11	
0	0	1	0	-	0
1	1	1	0	-	1

Hieraus lassen sich dann, z.B. mittels KV-Diagramm, die algebraischen Ausdrücke der Schaltnetze ermitteln.

Übergangsschaltnetz:

$ü^+$	— ü —			
	0	1	0	0
	1	1	X	X
			— r —	s

$$ü^+ = \overline{r}\,(ü \vee s) = \overline{r \vee \overline{ü \vee s}}$$

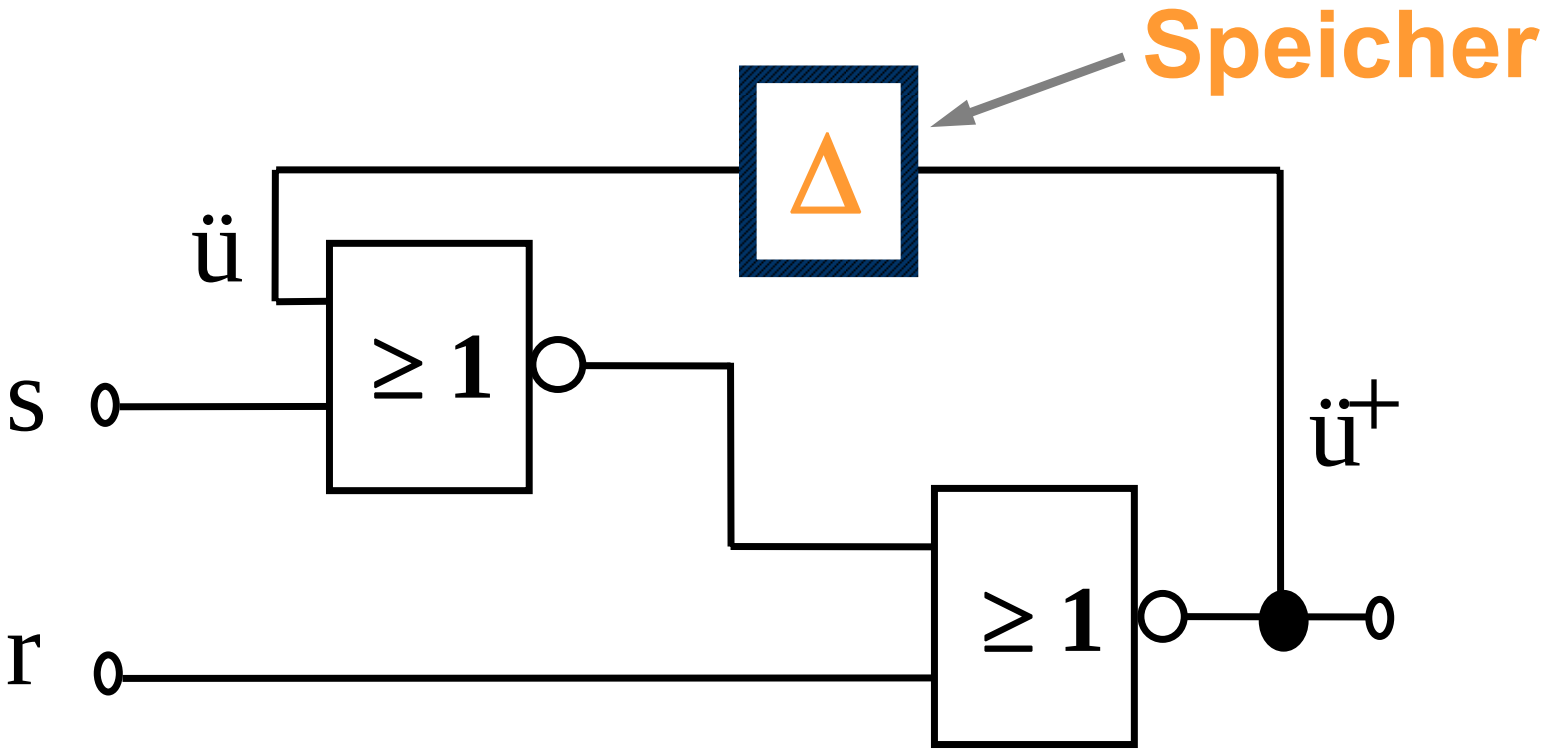
Ausgabeschaltnetz: *trivial*

$$q = ü$$

Schaltbild des Speichers

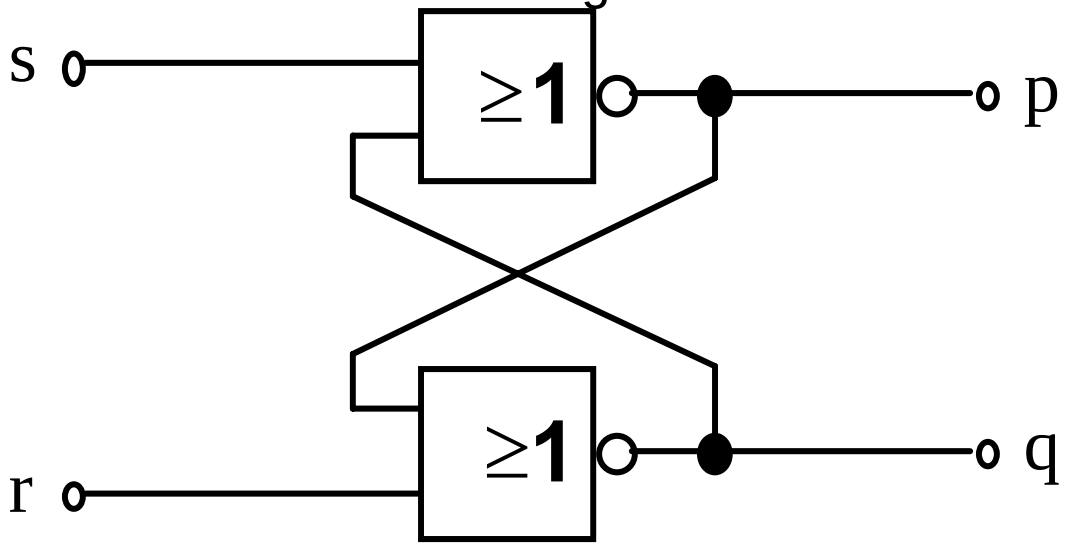
Übergangsgleichung: $\ddot{u}^{+} = \overline{r \vee \overline{\ddot{u} \vee s}} = \overline{r}\,(\ddot{u} \vee s)$

Ausgangsgleichung: $q = \ddot{u}$



Asynchrones RS-Flipflop

- Dieser Speicher ist ein Standardelement.
  - Es wird als asynchrones RS-Flipflop 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

Freie Universität Berlin

- 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 (wie bereits besprochen)
  - 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

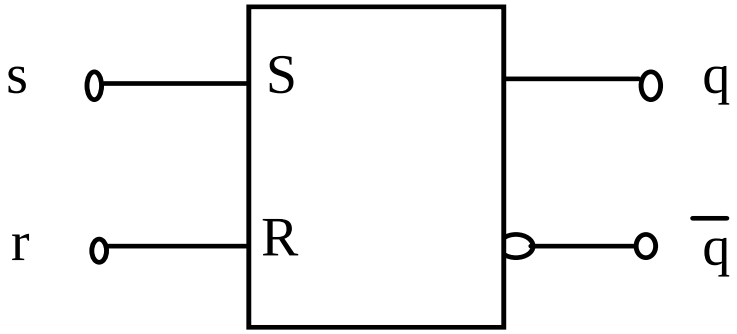
Flipflops als Zustandsspeicher

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

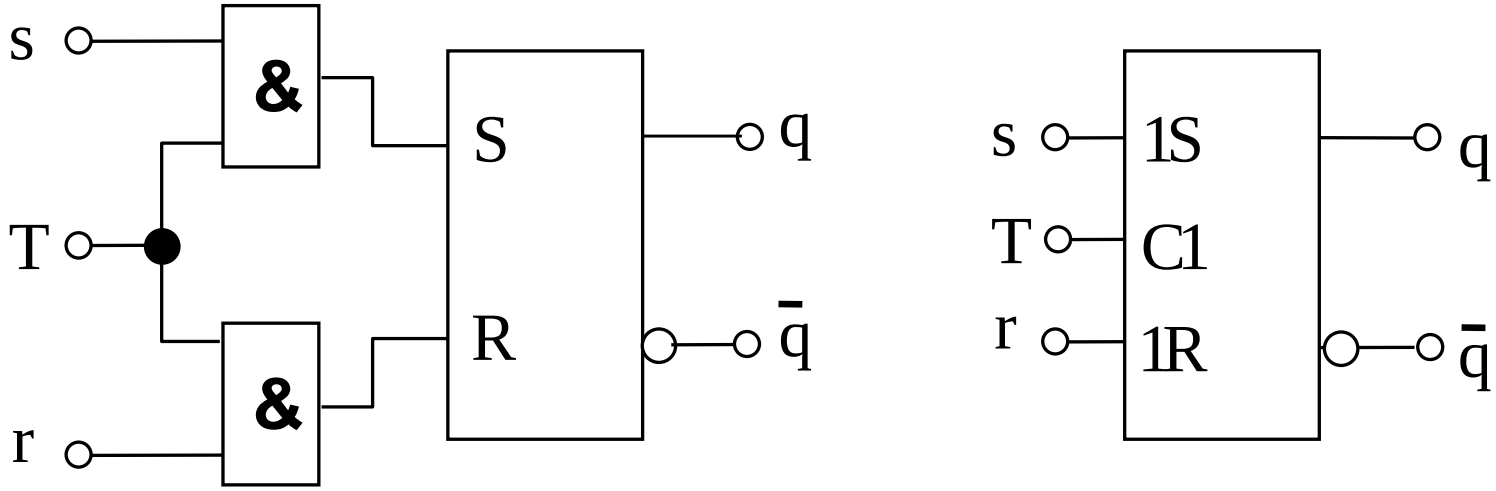
- Schaltsymbol des asynchronen RS-Flipflops:



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:

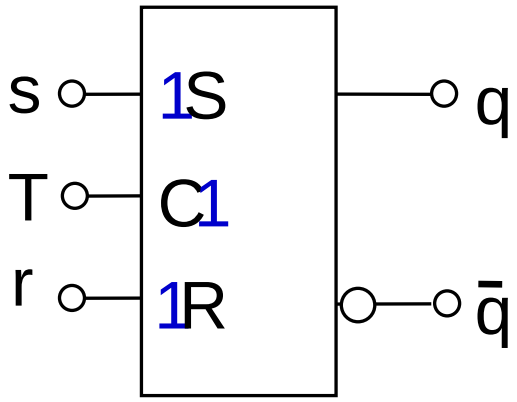
- Wir erhalten das pegelgesteuerte RS-Latch



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.Allg. 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
0	1	0	1
1	0	1	0
1	1	0	-

Voraussetzung:

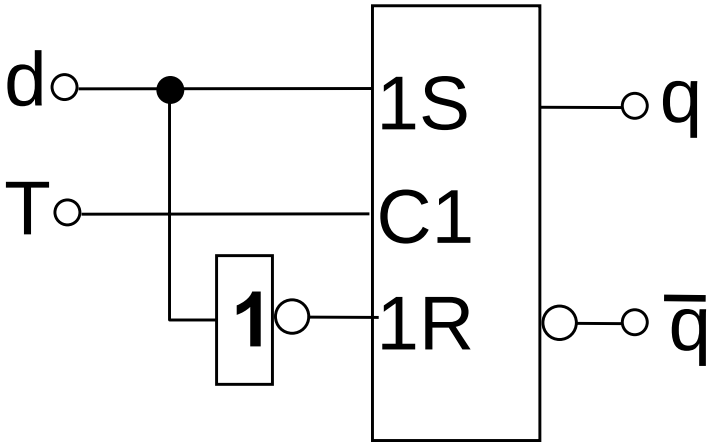
Es dürfen keine unerlaubten Eingangsbelegungen auftreten.

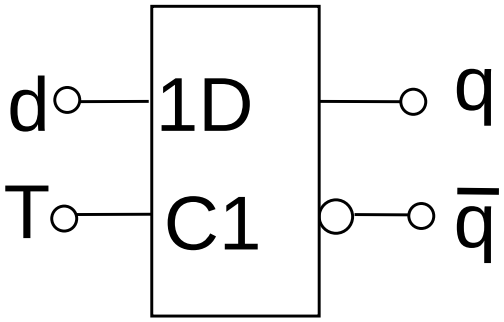
D-Flipflop

- 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 sog. D-Latch





Verhalten des D-Flipflops

Funktionstabelle:

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

- 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

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

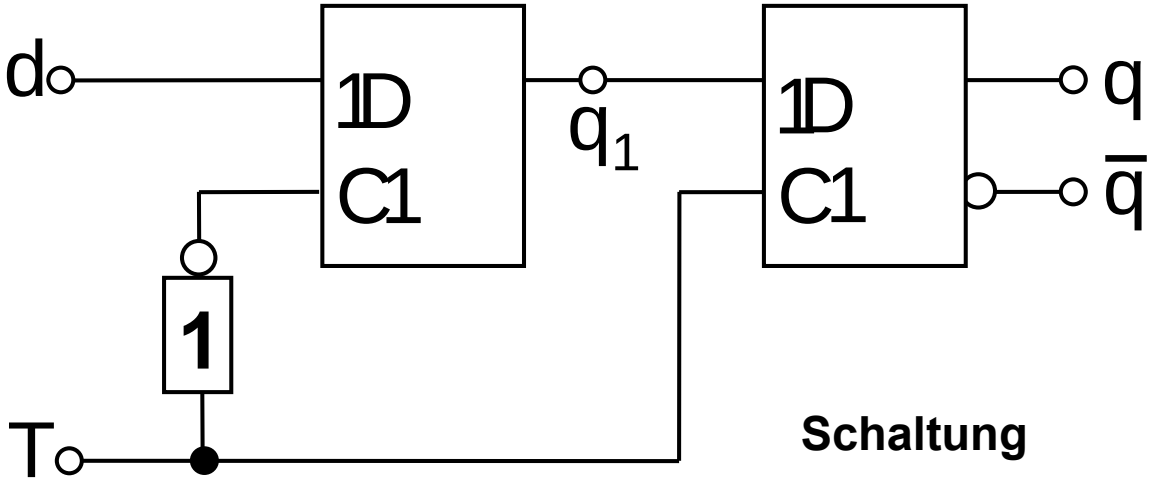
- Ein taktflankengesteuertes D-Flipflop erhält man durch die Zusammenschaltung zweier D-Latches, die mit komplementären Taktpegeln gesteuert werden.

- Das erste Latch wird Master-Latch, das zweite Slave-Latch genannt.

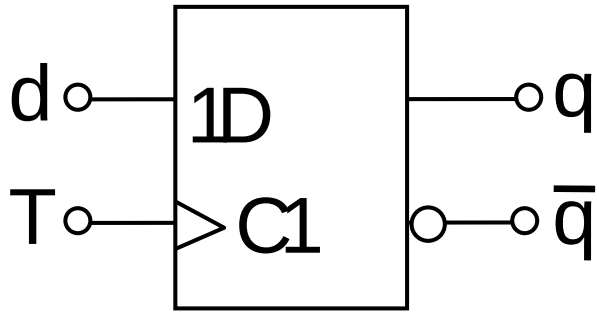
- Ein solches Flipflop wird auch als Master-Slave-Flipflop bezeichnet.

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

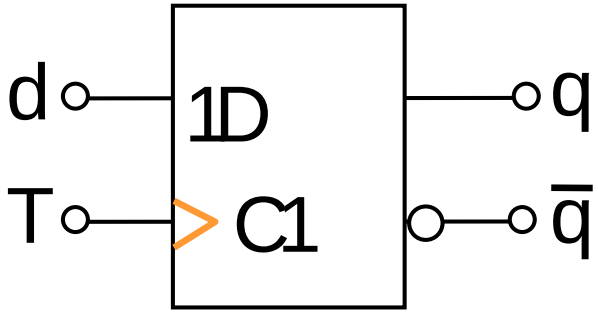


Symbol

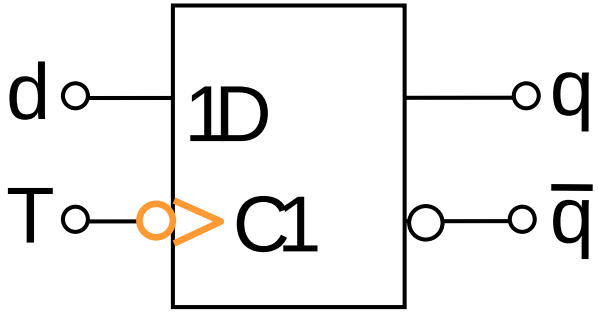
Anmerkungen

- 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.

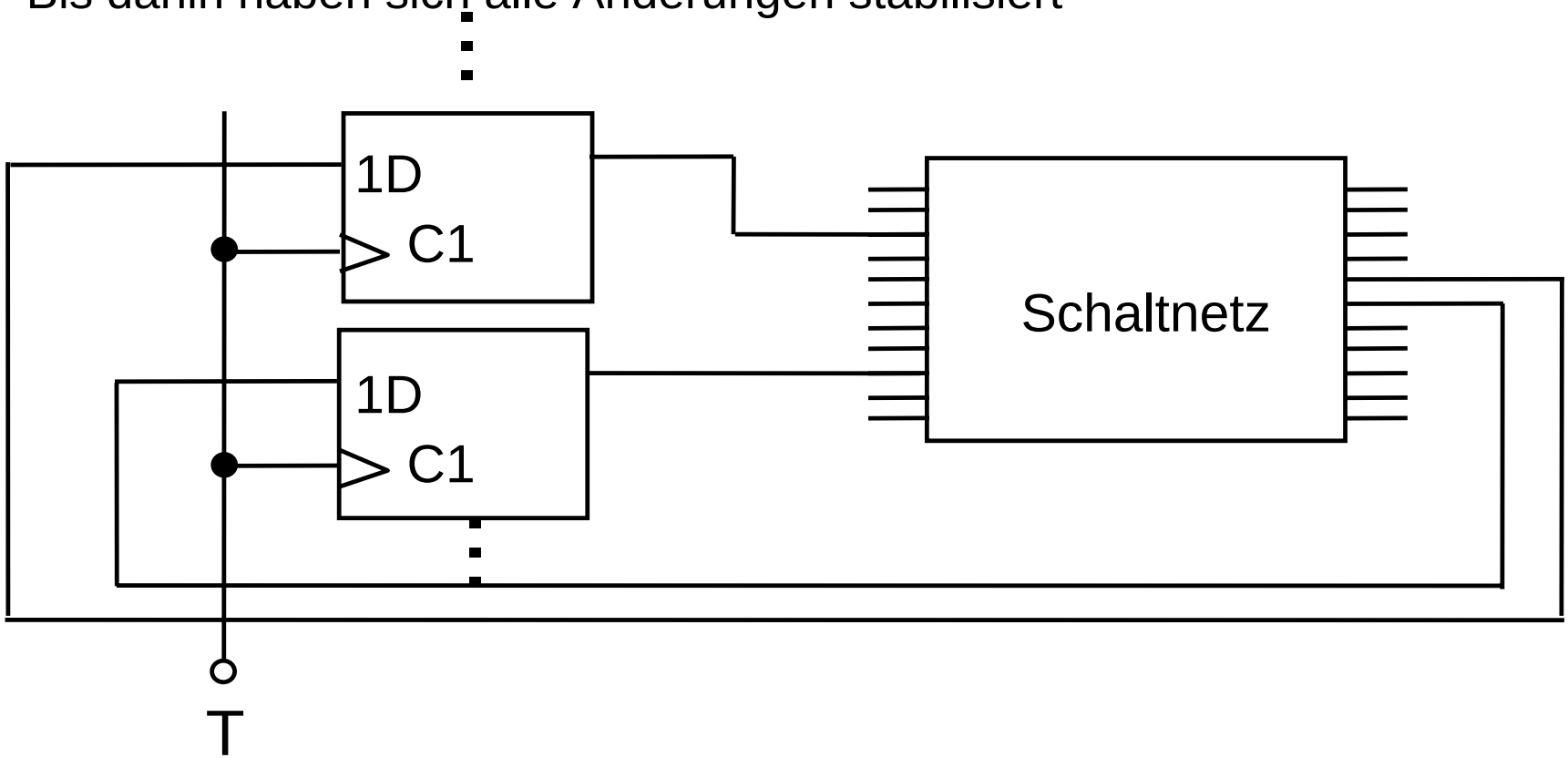


- Bei einer Steuerung mit der negativen Taktflanke wird ein Negationszeichen vor das Dreieck gesetzt.



Synchrones Schaltwerk mit flankengesteuerten D-Flipflops

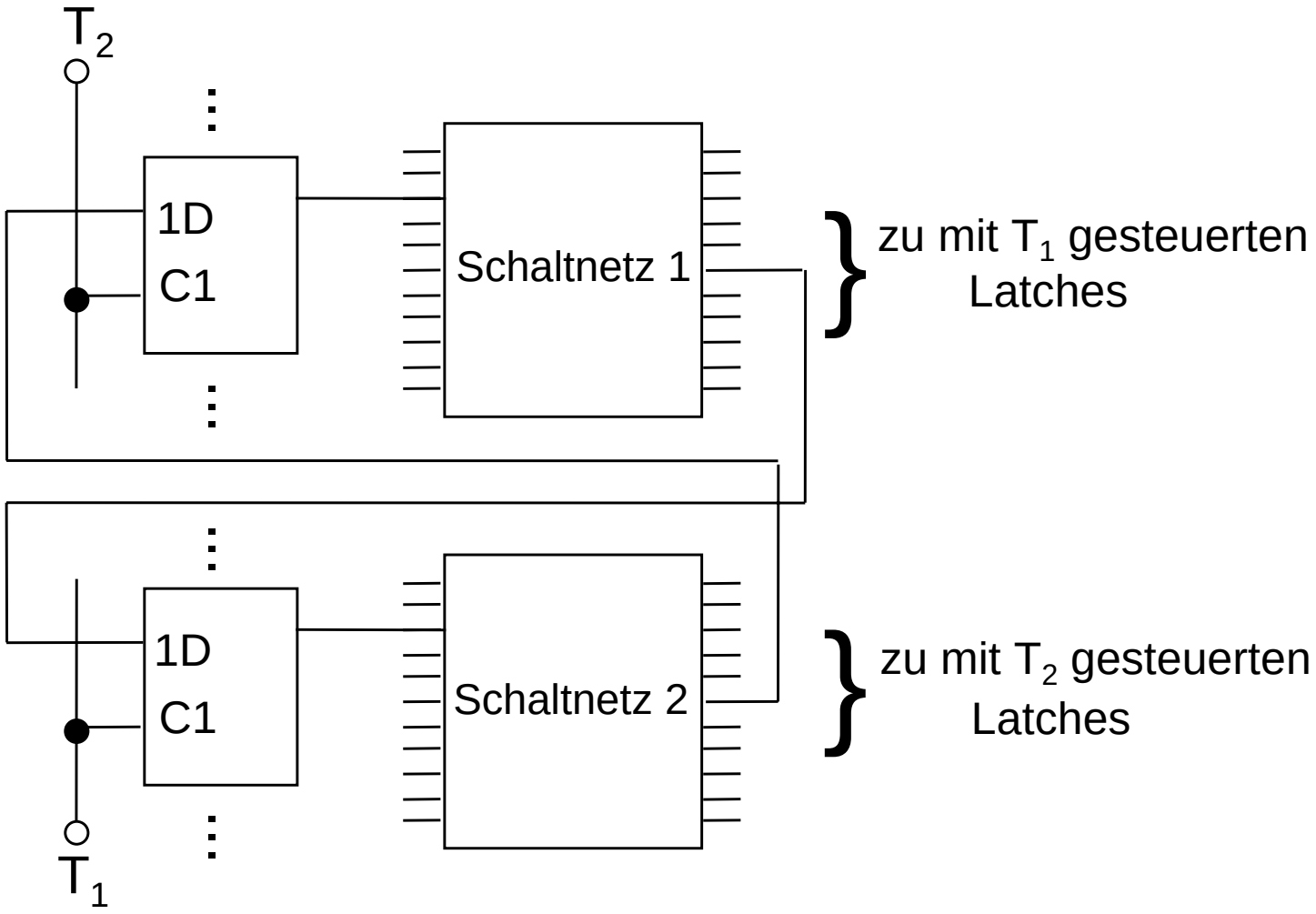
- Alle Flipflops werden mit der 0-1 Flanke von T gleichzeitig geladen
- Dies führt zur Änderung der Ausgabe und der Flipflop-Eingänge
- Änderungen an diesen Eingängen können sich jedoch erst bei der nächsten 0-1 Flanke auswirken
- Bis dahin haben sich alle Änderungen stabilisiert



Synchrones Schaltwerk mit Latches und Zweiphasentakt

- Synchrone Schaltungen lassen sich aber nicht nur mit flankengesteuerten Flipflops, sondern auch mit pegelgesteuerten Latches aufbauen.

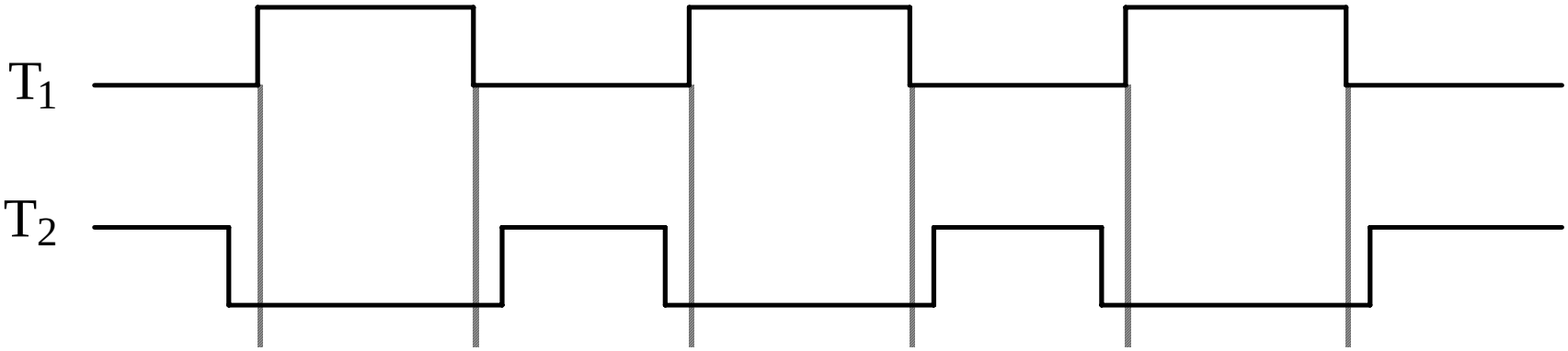
- Hierzu muss das Schaltnetz entsprechend aufgeteilt werden:



Funktionsweise

- Die mit T_2 getakteten Latches bleiben solange unverriegelt, solange $T_2 = 1$ ist (Pegelsteuerung).
- Wird jedoch gleichzeitig dafür gesorgt, dass $T_1 = 0$ ist, können sich die Ausgabeänderungen von Schaltnetz 1 nicht weiter auswirken.
- Umgekehrt bleiben durch $T_1 = 1$ bewirkte Änderungen der Ausgabe von Schaltnetz 2 solange ohne Wirkung, solange $T_2 = 0$ ist.

- Folgerung
  - zur Steuerung eines solchen Schaltwerks ist ein nichtüberlappender Zweiphasentakt erforderlich
  - Beide Takte dürfen nie gleichzeitig aktiv sein, da sonst eine asynchrone Rückkopplung entstehen würde.



Zusatzbetrachtung für flankengesteuerte Flipflops

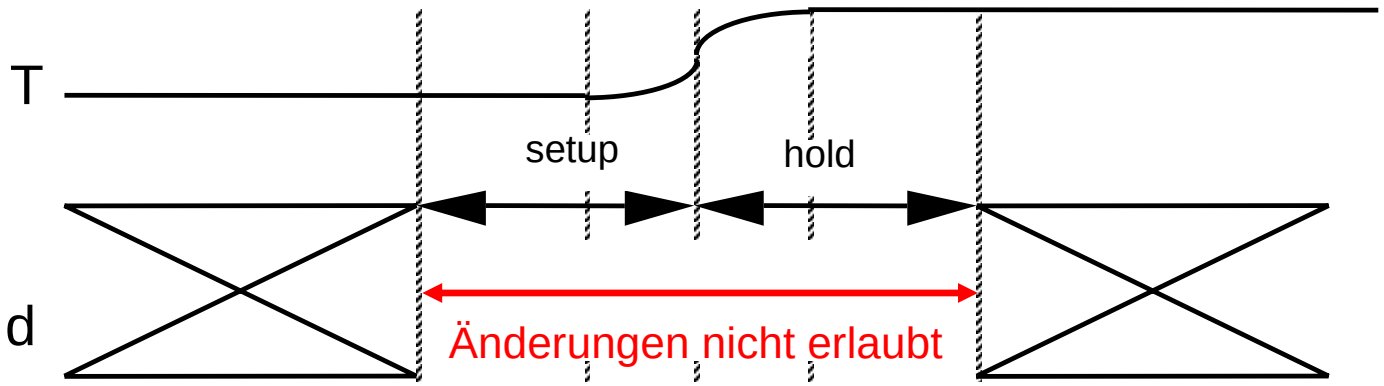
Freie Universität Berlin

- In der Theorie:
  - Das Flipflop ist nur zu einem genau festgelegten Zeitpunkt der 0-1 Flanke entriegelt
- In der Realität:
  - Das Taktsignal wechselt nicht unendlich schnell von 0 auf 1 und die Schaltglieder, aus denen das Flipflop aufgebaut ist, reagieren nicht unendlich schnell.
- Für das sichere Funktionieren eines Flipflops muss garantiert werden, dass sich seine Eingangssignale während eines gewissen Zeitintervalls um die Taktflanke herum nicht ändern.

Setzzeit (Setup-) und Haltezeit (Hold-Time)

- Setup Time: Vorbereitungszeit, spezifiziert die Zeit, in der das am d-Eingang des Flipflops anliegende Signal *vor* einer Taktflanke mindestens stabil sein muss.

- Hold Time: Haltezeit, spezifiziert die Zeit, in der das am d-Eingang des Flipflops anliegende Signal *nach* einer Taktflanke mindestens stabil sein muss.



Empfindlichkeitsintervall eines Flipflops

Zweiflankengesteuertes Flipflop

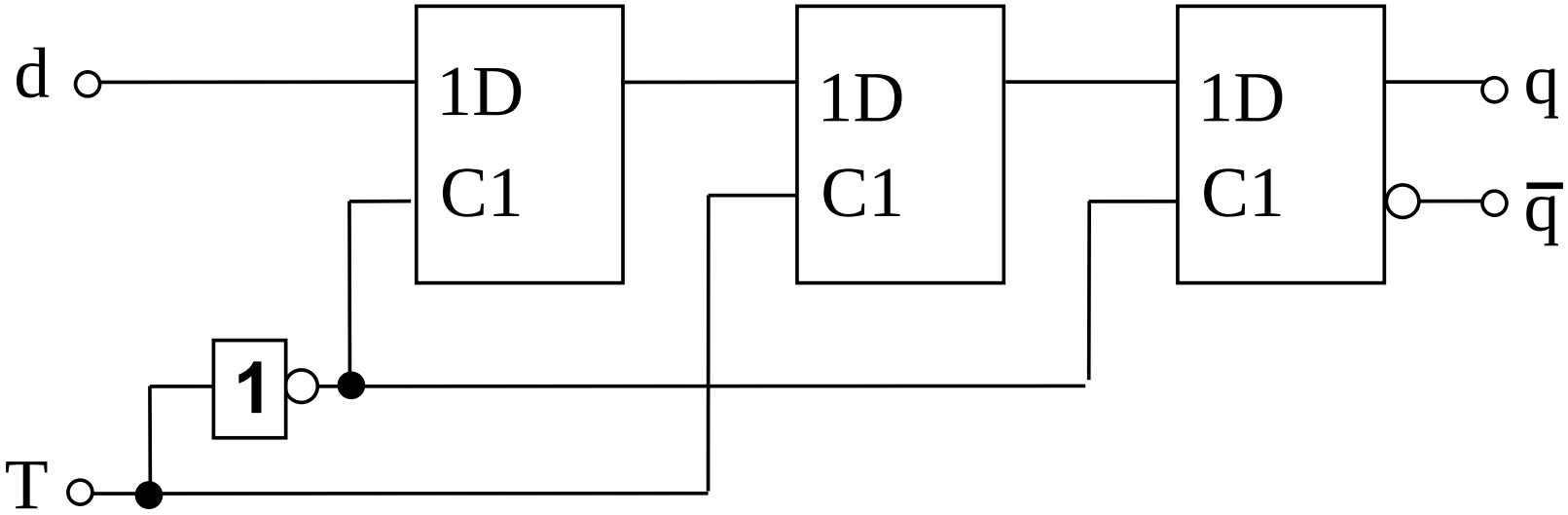
- Ist das Empfindlichkeitsintervall zu groß, so können bei direkt hintereinandergeschalteten Flipflops Probleme entstehen (z.B. bei Schieberegistern).

- Lösung: Ein zweiflankengesteuertes Flipflop

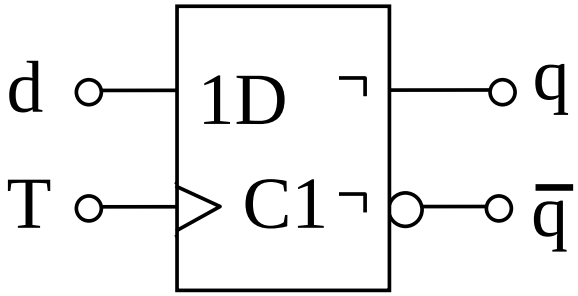
- Verhalten: Das Flipflop übernimmt und speichert die Daten z.B. bei der steigenden Taktflanke, gibt den gespeicherten Wert aber erst bei der nächsten fallenden Taktflanke aus.

- Der neue Zustand erscheint erst einen halben Takt später am Flipflop-Ausgang, wenn das folgende Flipflop wieder sicher verriegelt ist.

Zweiflankengesteuertes Flipflop



Die Verzögerung der Ausgabe um einen halben Takt wird durch das Zeichen "¬" am Ausgang des zugehörigen Schaltbildes symbolisiert.



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 "rücksetzen" 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.

Entwicklung eines JK-Flipflops aus einem D-Flipflop

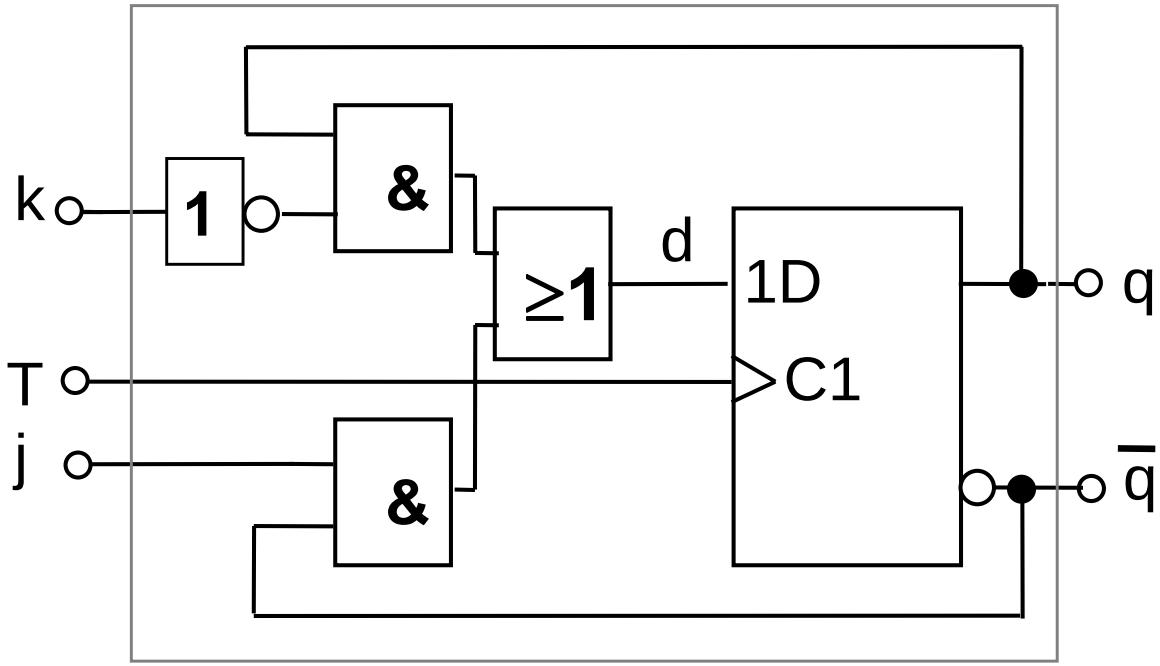
Freie Universität Berlin

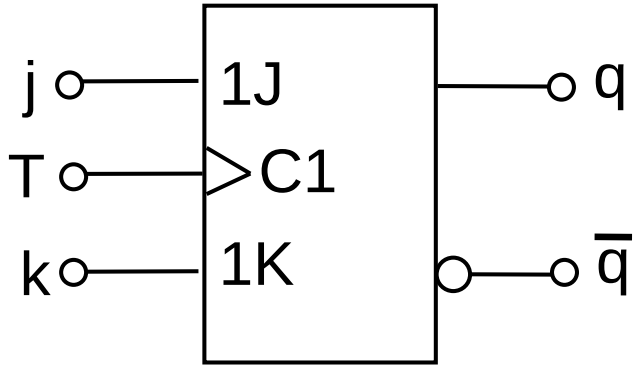
- Herleitung der jk-Ansteuerung eines D-Flipflops:
- Links:
  - Alle möglichen Kombinationen des aktuellen Zustandes q^t und der Eingangsvariablen j und k
- Rechts:
  - Folgezustand q^{t+1} und daraus resultierende Ansteuerwerte des D-Flipflops

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

- Die Ansteuervariable d lässt sich aus den Variablen q, j und k erzeugen. Nach der Minimierung (z.B. mit KV-Diagrammen) erhält man:
- $$d = q^t \, \overline{k} \vee \overline{q}^t \, j$$

Schaltbild des synchronen JK-Flipflops





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

Funktions- und 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

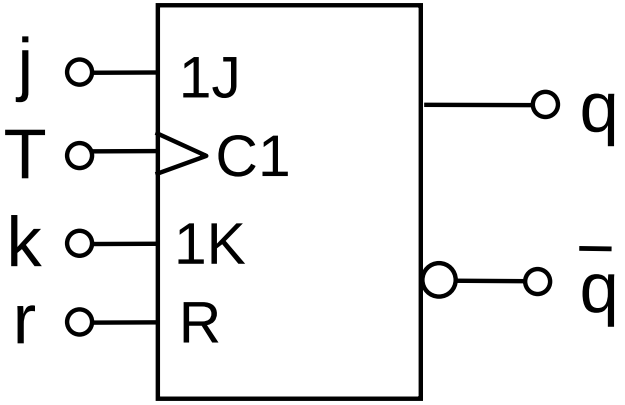
Aus obiger Tabelle lässt sich auch leicht die Ansteuertabelle des JK-Flipflops gewinnen:

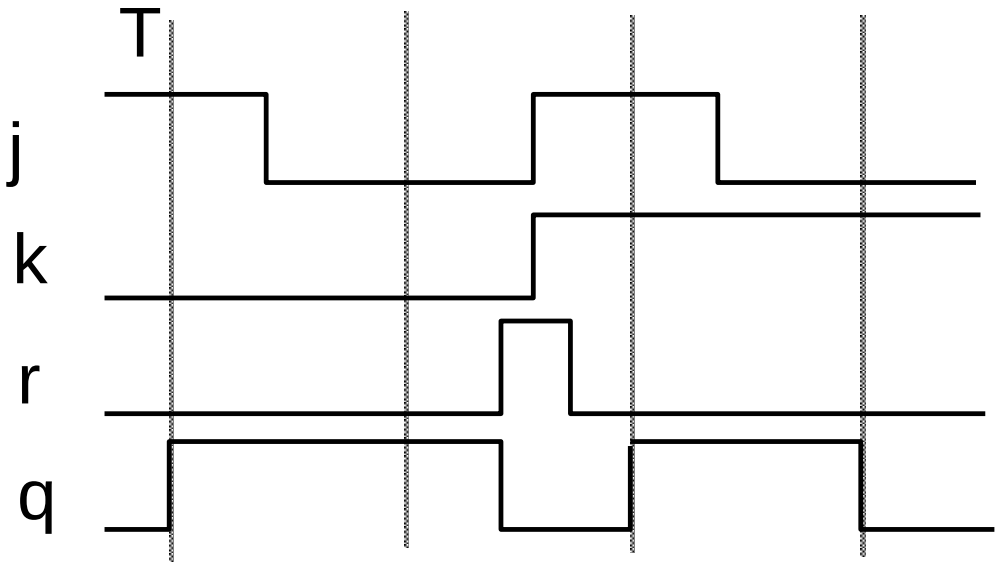
q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

Beispiel: Asynchron rücksetzbares JK-Flipflop

- Neben den synchronen Eingängen können Flipflops auch zusätzliche Asynchroneingänge besitzen.

- Hier: JK-Flipflop mit einem asynchronen Rücksetzeingang R, der das Flipflop unabhängig vom Taktsignal in den Zustand q = 0 bringt.





Asynchron rücksetzbares JK-Flipflop

- Anwendung solcher asynchroner Eingänge:
  - Bei Initialisierung eines Schaltwerks alle Flipflops in einen definierten Grundzustand bringen
-

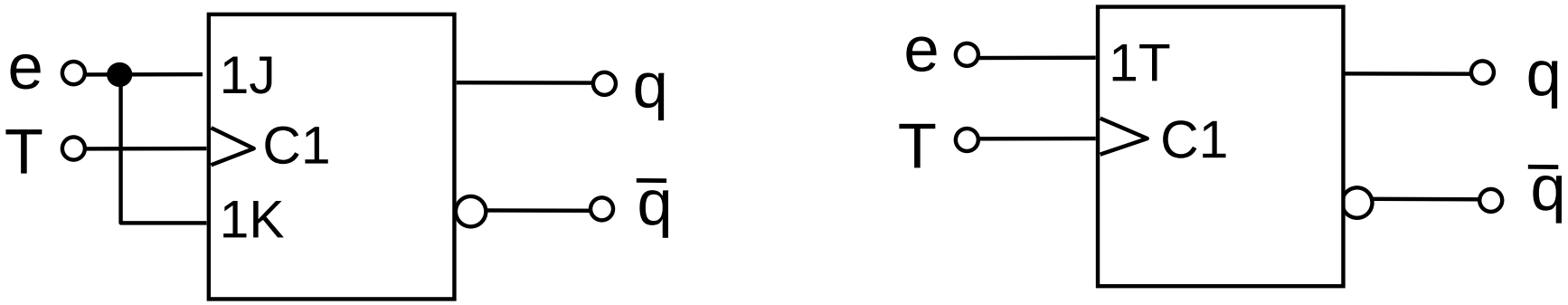
  Aus der Bezeichnung der anderen Eingänge mit 1J und 1K geht hervor, dass diese Eingänge vom Taktsignal C1 abhängen, also synchron sind, während der asynchrone Eingang R nicht mit 1 bezeichnet wurde.

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.

T-Flipflop: Ansteuertabelle

- Ansteuertabelle des T-Flipflops:

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

- Um einen Zustand zu speichern ($0 \rightarrow 0$ oder $1 \rightarrow 1$), ist e = 0 notwendig

- Für das Kippen des Flipflops (Zustandsübergänge $0 \rightarrow 1$ oder $1 \rightarrow 0$) muss e = 1 gelten.

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

Freie Universität Berlin

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

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.

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

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:

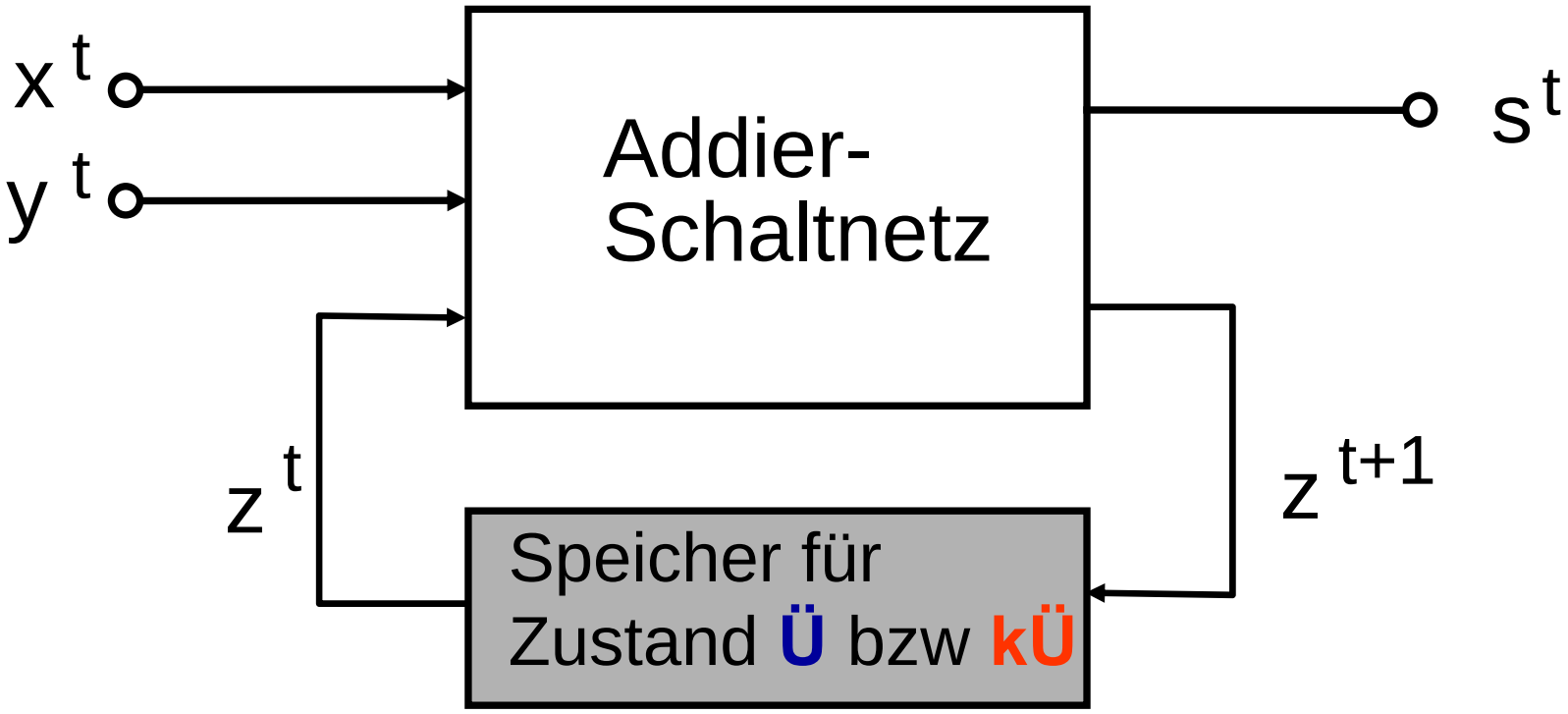
- Zustand Ü: wird in dem Fall eingenommen, dass ein Übertrag aus der vorhergehenden Stelle zu berücksichtigen ist

- Zustand kÜ: repräsentiert den anderen Fall (kein Übertrag)

Blockschaltbild des Serienaddierers

Eingabe: x und y Ausgabe: s

Zustandsspeicher für Ü bzw. kÜ

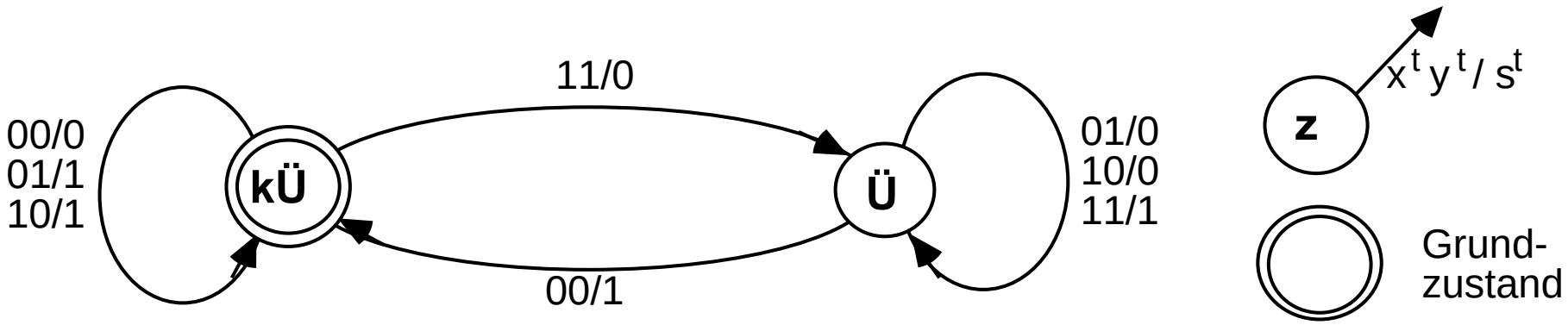


Automatengraph

- Es ist ein Mealy-Schaltwerk nötig, da die Ausgabe von den aktuellen Werten der Eingabevariablen abhängen soll

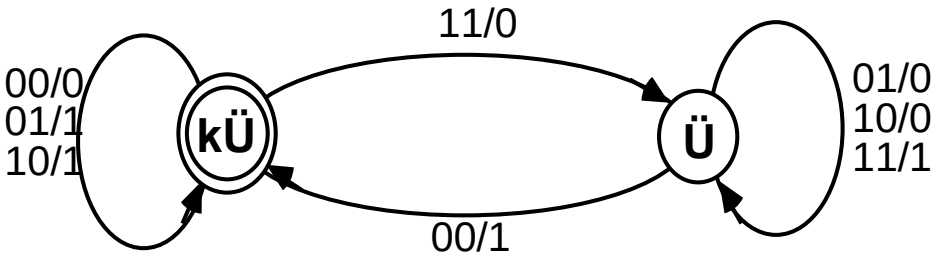
- Im Automatengraphen werden deshalb die Ausgabebelegungen an die Kanten geschrieben

- 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 entscheidend für das Funktionieren des Schaltwerks

- 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

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	•	$ü_1^{t+}$	•	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

- 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

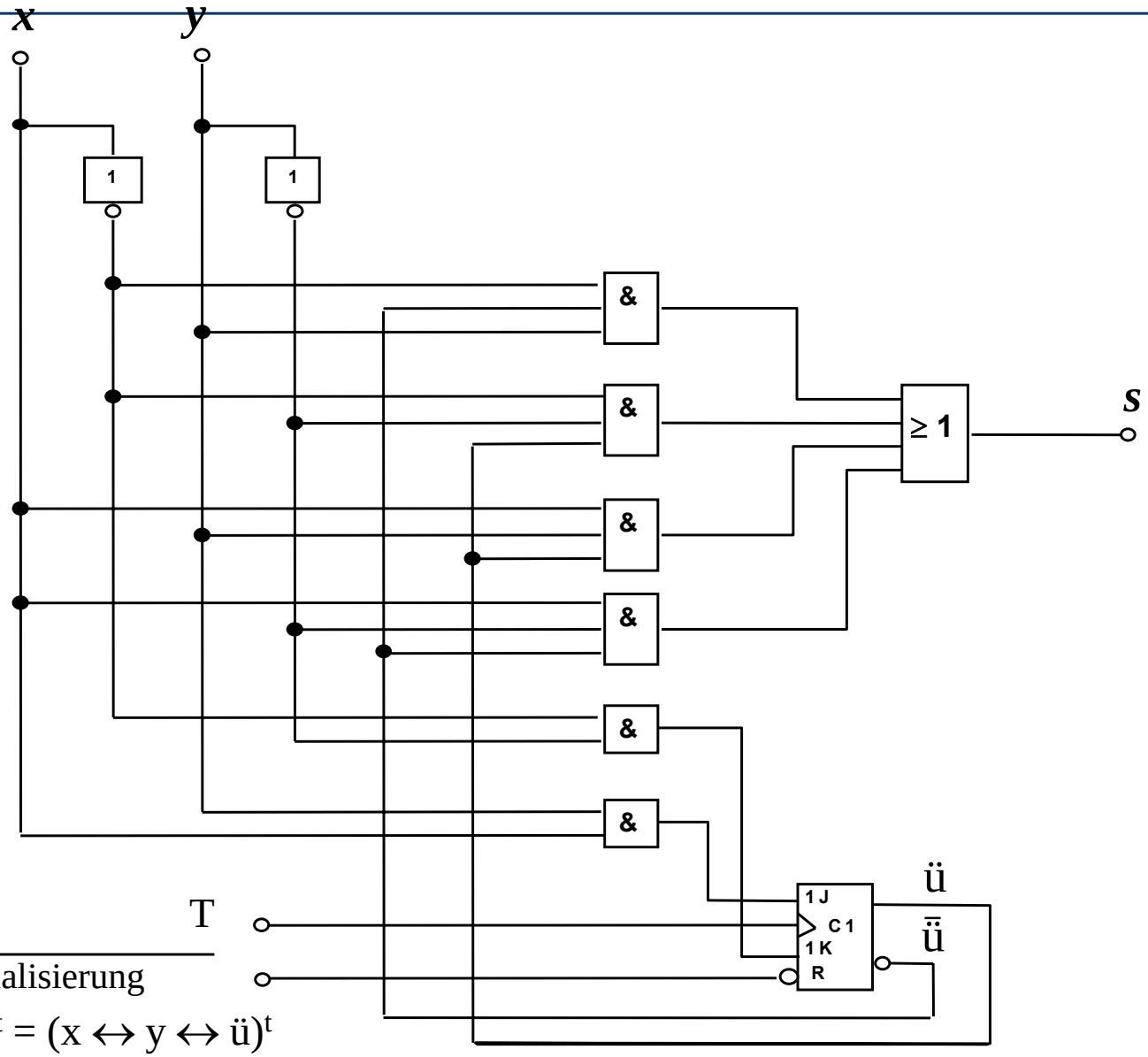
ü

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

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

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

Realisierung des Serienaddierers



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

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

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

Beispiel: Folgenerkenner

- Weiteres Beispiel für den Entwurf eines synchronen Schaltwerks

- Verbale Aufgabenstellung:
- Synchrones Mealy-Schaltwerk mit zwei Eingangssignalen x_1 und x_2 und einem Ausgangssignal y.

- Am Eingang x_1 wird eine Folge von Binärzeichen empfangen.

- Während der zweite Eingang $x_2 = 0$ ist, wird die Folge von Binärzeichen beobachtet.

- Der Ausgang y wird genau dann für einen Takt gleich 1, wenn während eines Beobachtungszeitraumes am Eingang x_1 die Folge 101 auftritt und die Beobachtung mit $x_2 = 1$ beendet wird.

Technische Randbedinungen

- Möglichst geringe Gatteranzahl
- Nur NAND- und NOR-Gatter
- Minimale Anzahl von JK-Flipflops

- Anmerkung:
- Schaltungen dieser Art finden sich beispielsweise auf Netzwerkkarten, um den Anfang eines Datenpakets zu erkennen (Hier wird z.B. das Muster 01111110 erwartet).

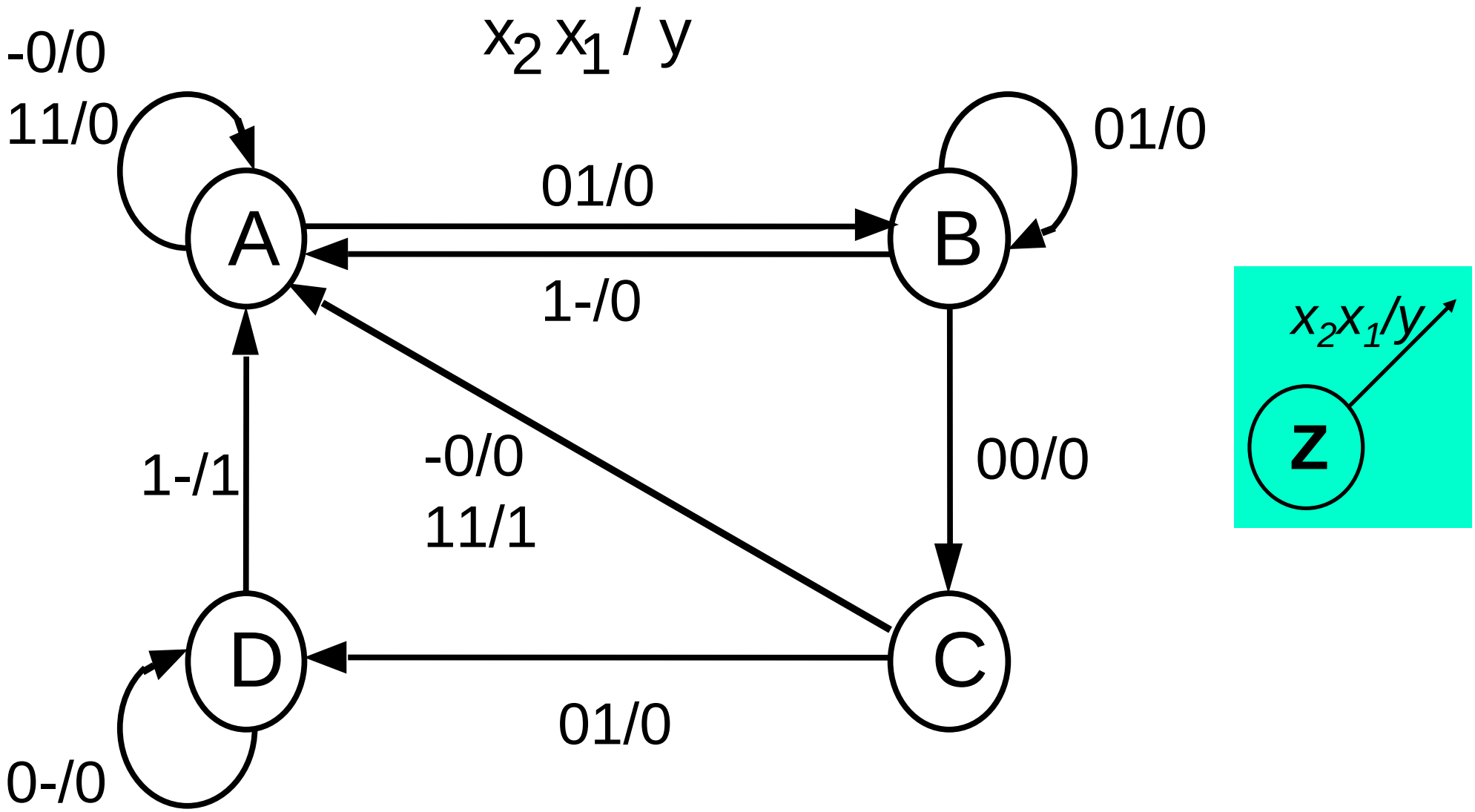
1. Schritt: Formalisierung der Aufgabe

- Eingaben: x_2, x_1

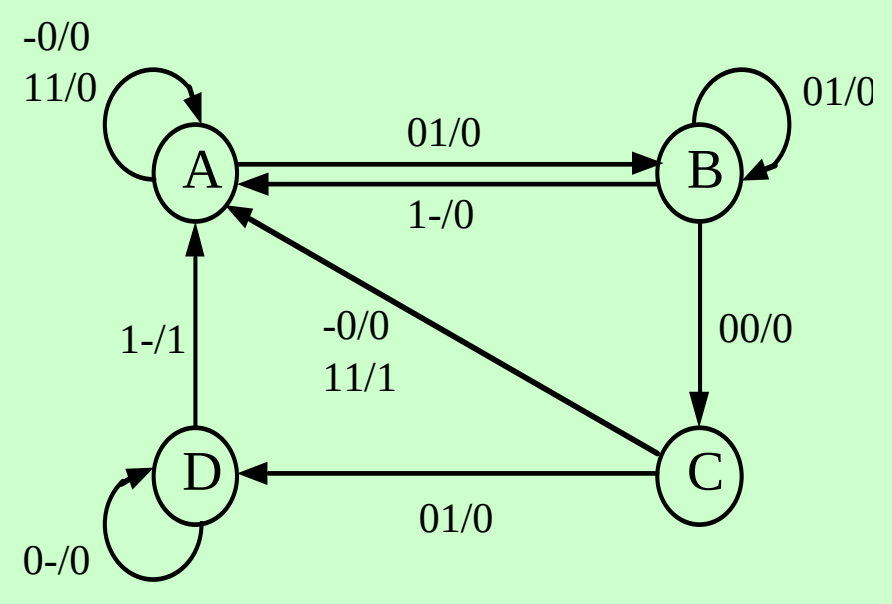
- Ausgabe: y

- Zustände:
- A Grundzustand, Warten auf Start der Beobachtung $x_2 = 0$ und das erste Zeichen $x_1 = 1$ der Folge 101
- B Erstes Zeichen $x_1 = 1$ der Folge 101 (evtl. erneut) empfangen, Warten auf zweites Zeichen $x_1 = 0$ oder Abbruch der Beobachtung $x_2 = 0 \rightarrow 1$
- C Zweites Zeichen $x_1 = 0$ empfangen, Warten auf drittes Zeichen $x_1 = 1$ oder Abbruch der Beobachtung $x_2 = 0 \rightarrow 1$ (falls Beobachtungsabbruch und drittes Zeichen gleichzeitig, dann Ausgabe von y = 1)
- D Drittes Zeichen $x_1 = 1$ empfangen, Warten auf Abbruch der Beobachtung $x_2 = 0 \rightarrow 1$ und Ausgabe von y = 1

Automatengraph



Automatentabelle



Z^t	Z^{t+1}/y bei $x_2x_1=$			
	00	01	10	11
A	A/0	B/0	A/0	A/0
B	C/0	B/0	A/0	A/0
C	A/0	D/0	A/0	A/1
D	D/0	D/0	A/1	A/1

2. Schritt: Zustandskodierung

- Die Mindestzahl der Flipflops beträgt bei vier Zuständen
- $\lceil \mathrm{ld}\ 4 \rceil = 2$
- Bei der Kodierung werden also jedem Zustand Z^t zwei Werte von binären Zustandsvariablen z_1 und z_2 zugewiesen.
- Diese Zuweisung erfolgt hier willkürlich, aber eindeutig:
- $A \triangleq (z_1=1, z_2=0)$; $B \triangleq (z_1=0, z_2=0)$;
- $C \triangleq (z_1=0, z_2=1)$; $D \triangleq (z_1=1, z_2=1)$;

3.Schritt: kodierten Ablauftabelle

Eingabe		Z^t		Z^{t+1}		Flipflops				Ausgabe
x_2	x_1	z_2	z_1	z_2	z_1	J_2	K_2	J_1	K_1	y
1	-	0	1	0	1	0	-	-	0	0
0	0	0	1	0	1	0	-	-	0	0
0	1	0	1	0	0	0	-	-	1	0
1	-	0	0	0	1	0	-	1	-	0
0	1	0	0	0	0	0	-	0	-	0
0	0	0	0	1	0	1	-	0	-	0
-	0	1	0	0	1	-	1	1	-	0
1	1	1	0	0	1	-	1	1	-	1
0	1	1	0	1	1	-	0	1	-	0
0	-	1	1	1	1	-	0	-	0	0
1	-	1	1	0	1	-	1	-	0	1

4.Schritt: Ansteuerfunktionen der Flipflops

Ansteuerfunktionen des Flipflops 2:

$z_2 \backslash z_1 x_1 x_2$			
1	0	0	0
-	-	-	-
-	-	-	-
0	0	0	0

$$J_2 = \overline{x}_2\,\overline{x}_1\,\overline{z}_1 = \overline{x_2 \vee x_1 \vee z_1}$$

$z_2 \backslash z_1 x_1 x_2$			
-	-	-	-
1	0	1	1
0	0	1	1
-	-	-	-

$$K_2 = x_2 \vee \overline{x}_1\,\overline{z}_1 = \overline{\overline{x}_2 \;\&\; \overline{\overline{x}_1\,\overline{z}_1}}$$

4.Schritt: Ansteuerfunktionen der Flipflops

Ansteuerfunktionen des Flipflops 1:

z_2 \ z_1				
	0	-	-	1
	1	-	-	1
	1	-	-	1
	0	-	-	1

x_1, x_2

$$J_1 = x_2 \vee z_2 = \overline{x_2\,\bar{z}_2}$$

z_2 \ z_1				
	-	0	0	-
	-	0	0	-
	-	0	0	-
	-	1	0	-

x_1, x_2

$$K_1 = x_1\,\bar{x}_2\,\bar{z}_2 = \overline{\bar{x}_1 \vee x_2 \vee z_2}$$

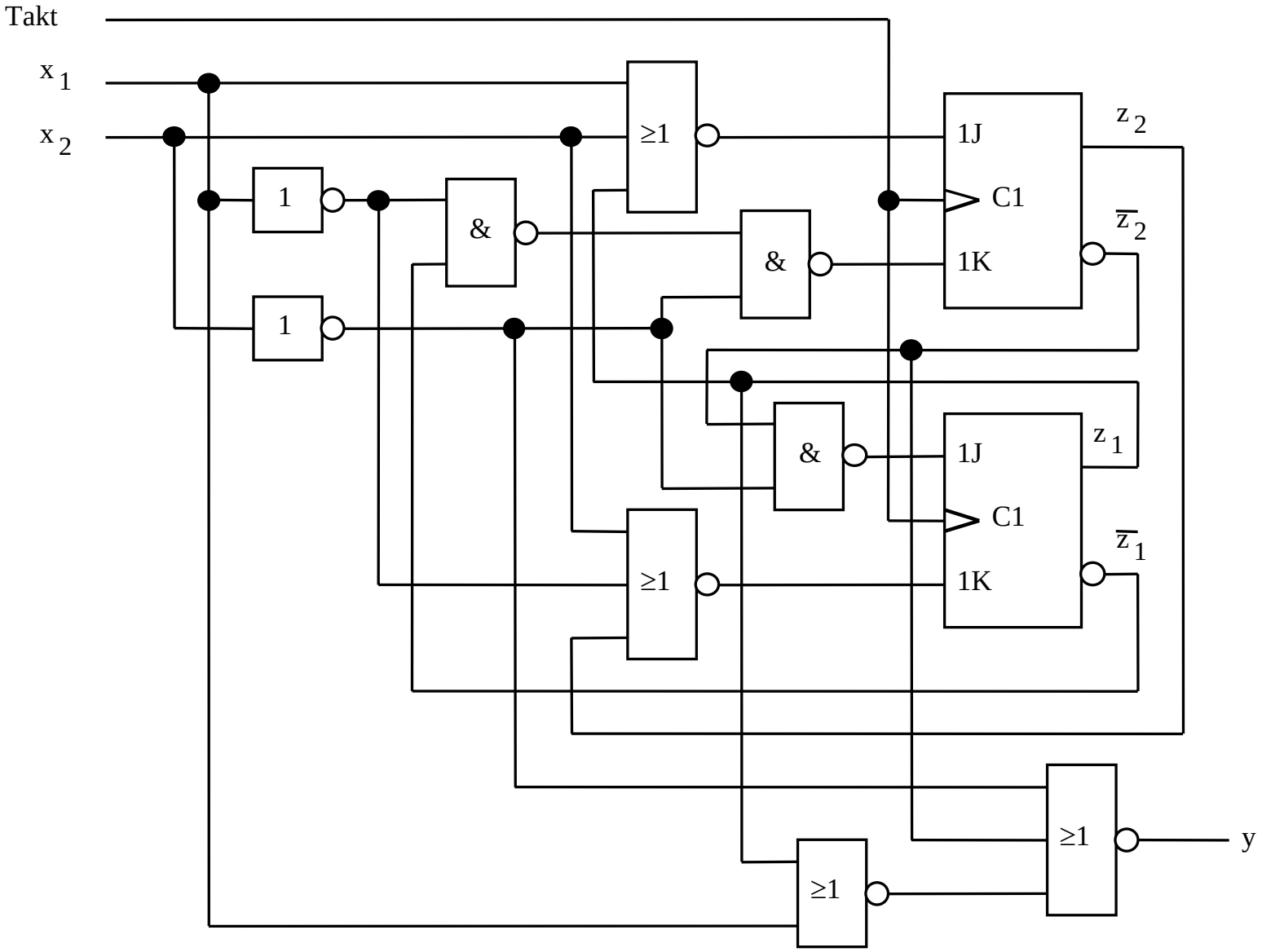
4.Schritt: Ausgabefunktion

Ausgabefunktion

		z_1		
	0	0	0	0
z_2	0	0	1	0
	0	0	1	1
	0	0	0	0
		x_2		x_1

$$y = x_2\, x_1\, z_2 \vee x_2\, z_2\, z_1 = \overline{\overline{x}_2 \vee \overline{z}_2 \vee \overline{z_1 \vee x_1}}$$

5. Schritt: Realisierung des Schaltwerks mit den gegebenen Bauelementen



Zusammenfassung: Entwurf synchroner Schaltwerke

Freie Universität Berlin

- Ausgangsbasis: verbalen 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

Register-Transfer-Ebene

Freie Universität Berlin

- Für den Entwurf komplexerer Systeme erweist es sich als notwendig, vom detaillierten Verhalten konkreter Bausteine zu abstrahieren.

- Übergang von der Beschreibung der Schaltung auf der Gatterebene zu einer Beschreibung auf der sog. Register-Transfer-Ebene (RT-Ebene)

- Grundelemente der RT-Ebene
  - Register zur Speicherung der zu verarbeitenden Daten
  - Funktionale Einheiten, wie Zähler, Addierer, Multiplexer und Demultiplexer, mit denen die Verarbeitung durchgeführt werden kann.

- Die Einheit des Informationsflusses zwischen diesen Elementen ist das Datenwort, das eine feste Anzahl zusammengehörender Bits umfasst.
  - Die zu verarbeitenden Daten sind oft also keine einzelnen Bits, sondern Bit-Gruppen oder Bitvektoren.

Register-Transfer-Ebene
Komponenten

Freie Universität Berlin

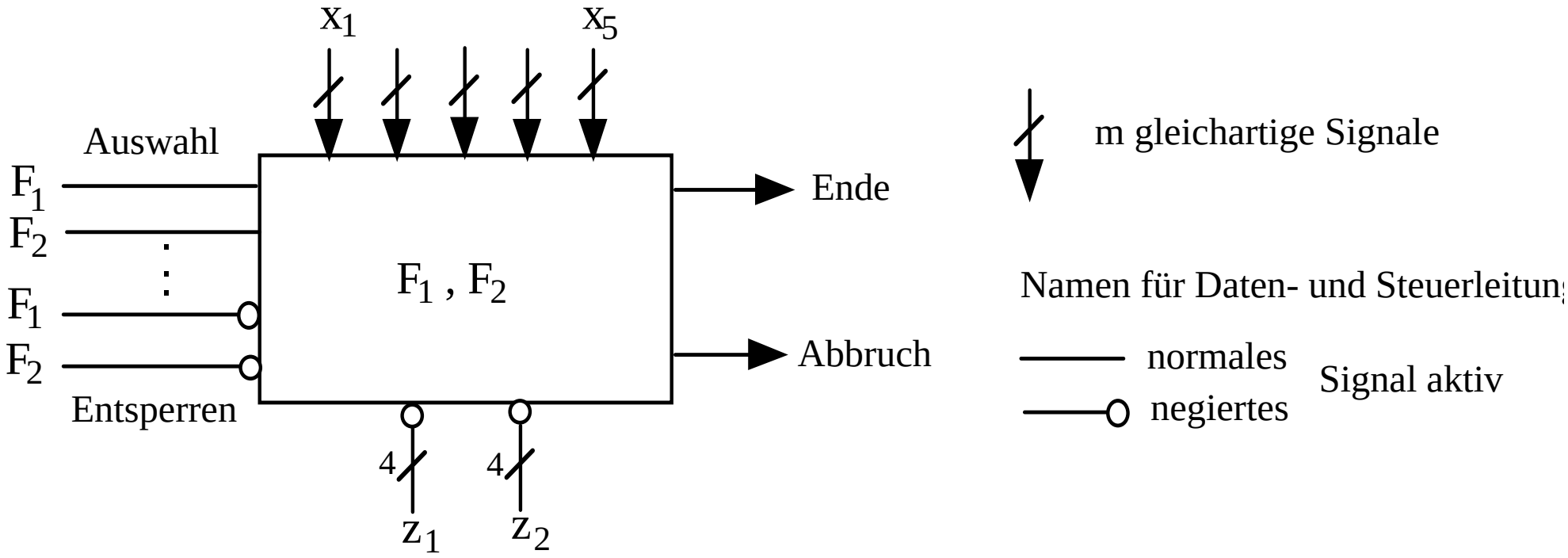
Typ	**Komponente**	**Funktion**
Schaltnetze	Wortverknüpfung	Boolesche Operation
	Multiplexer (MUX)	Datenübertragung, allg. Funktionen
	Kodierer/Dekodierer	Codeprüfung, Codewandlung
	Programmierbare Matrizen (PLA)	Allgemeine Funktionen
	Arithmetische Elemente (Addierer, arithmetisch-Logische Einheit ALU)	Numerische Operationen
Schaltwerke	Register	Informationsspeicherung
	Schieberegister	Serien-/Parallel-Umwandlung
	Zähler	Erzeugung von Steuer- und Zeitgebersignalen

Multiplexer und PLA-Bausteine sind funktional vollständig, d.h. sie sind zusammen mit Registern für den Entwurf eines allgemeinen Schaltwerks ausreichend.

Register-Transfer-Ebene
Symbole auf der RT-Ebene

- Als Schaltsymbole zur Kennzeichnung der Funktion eines Bausteins werden Kästen mit Funktionsangabe oder Normsymbole verwendet.

- Beispiel:



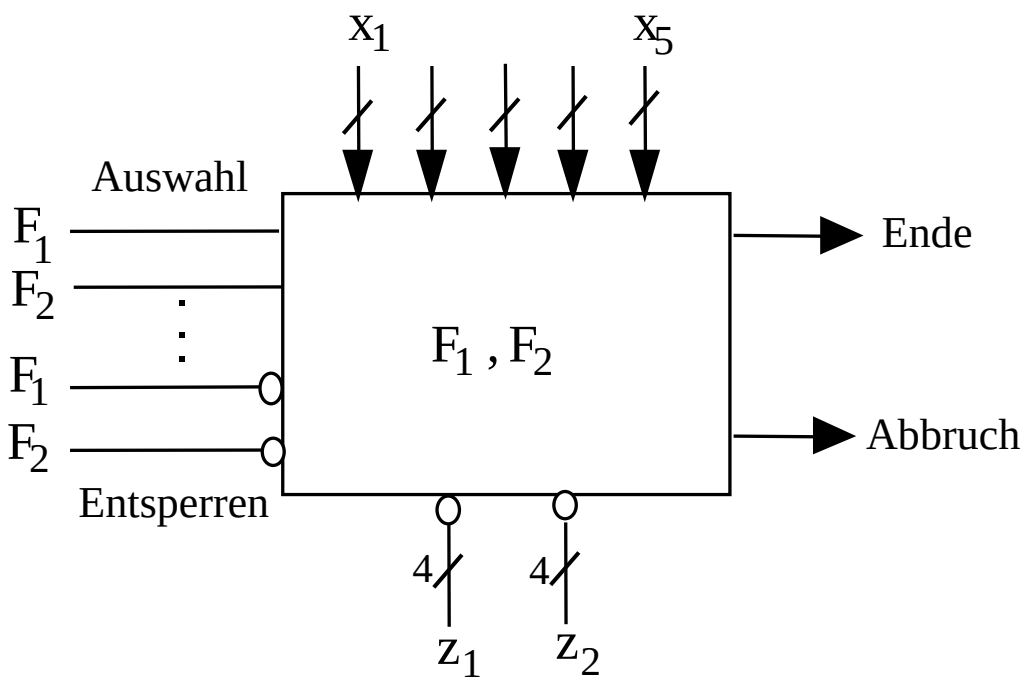
Symbole auf der RT-Ebene

Auswahlsignale:

Legen fest, welche Funktion realisiert werden soll.

Entsperrsignale:

Definieren den Zeitpunkt für den Beginn einer Operation oder erlauben deren Ablauf.



Register-Transfer-Ebene
Beschreibung von RT-Operationen

Freie Universität Berlin

- Eine Beschreibung von Register-Transfer-Operationen enthält die Grobstruktur des Datenflusses zwischen den speichernden und den verarbeitenden Einheiten:

- Blockdiagramm, in dem Leitungsbündel Register und Funktionsbausteine miteinander verbinden oder

- Beschreibung mit einer Hardware-Beschreibungssprache wie z.B. VHDL, VerilogHDL (HDL=Hardware Description Language).

Register-Transfer-Ebene
Anweisungen

- Als Basis zur Beschreibung von RT-Operationen dienen sog. Register-Transfer-Anweisungen der Form:

- $Z \leftarrow F(X_1, X_2, ..., X_n)$

- $Z, X_1, X_2 ..., X_n$ stellen Register dar.

- F ist eine numerische oder Boolesche Funktion.

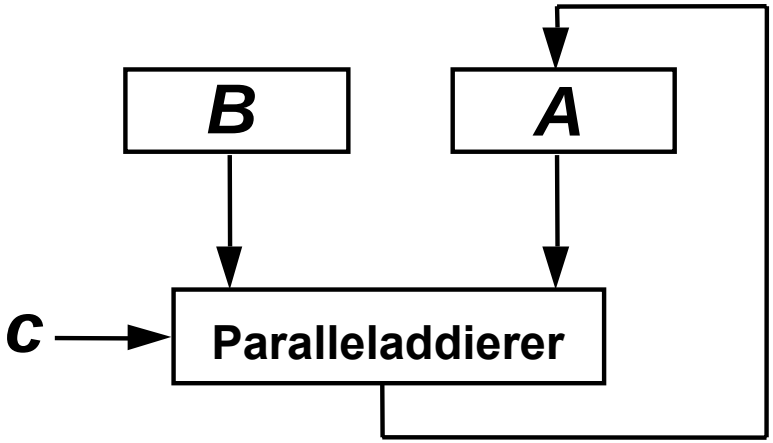
- Die Anweisung kann wie folgt interpretiert werden:
  - Die Verknüpfung der Eingangsgrößen $X_1, X_2 ..., X_n$ durch F liefert den Registerinhalt für Z in den nächsten Taktschritten.

Register-Transfer-Ebene
Anweisungen

- Eine Beeinflussung der Ausführung der Anweisung erfolgt durch eine Steuergröße c:
- c: $Z \leftarrow F(X_1, X_2, ..., X_n)$
- oder
- if (c=1) then $Z \leftarrow F(X_1, X_2, ..., X_n)$
- Die Anweisungen einer Register-Transfer-Sprache können durch Hardware-Komponenten interpretiert werden.

Register-Transfer-Ebene
Anweisungen

- Beispiel:
- c: A ← A + B lässt sich als Blockdiagramm darstellen.



- Die Summe A+B wird nach A übernommen, wenn die Bedingung c = 1 erfüllt ist.
- Die Steuervariable c ist dabei offen und muss spezifiziert werden.

Register-Transfer-Ebene
Logische Bausteine

- Um Register-Transfer-Operationen implementieren zu können, werden Bausteine verwendet, die Variablenbündel statt einzelner Variablen verarbeiten und somit mehrstellige Operationen ausführen.

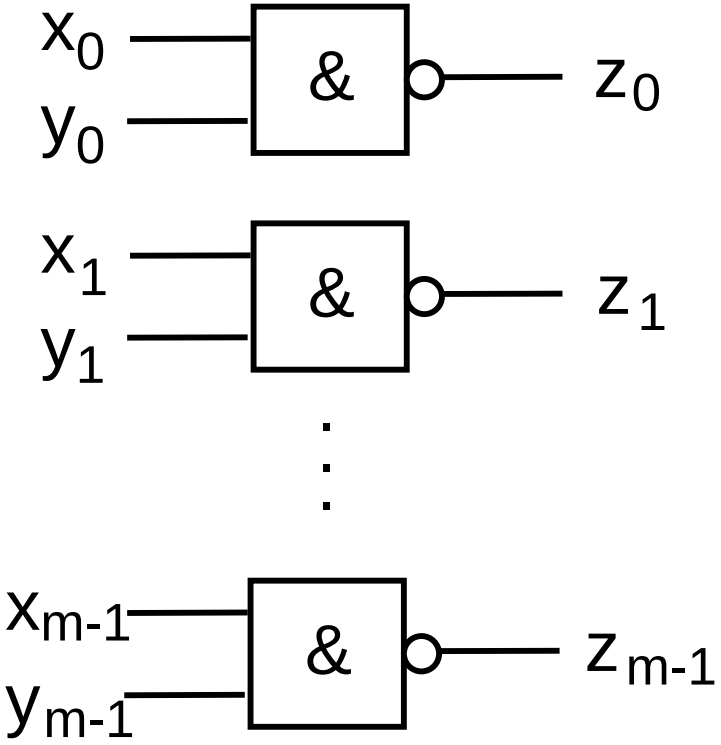
- Beispiel 1: Mehrstellige, parallele, logische Operationen

- $Z = \overline{X \wedge Y}$

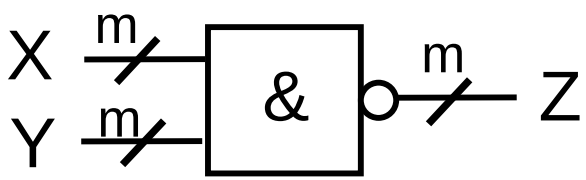
- sei gleich bedeutend mit

- $(z_0, z_1, \ldots z_{m-1}) = (\overline{y_0\, x_0}, \overline{y_1\, x_1}, \ldots \overline{y_{m-1}\, x_{m-1}})$

Register-Transfer-Ebene
Beispiel



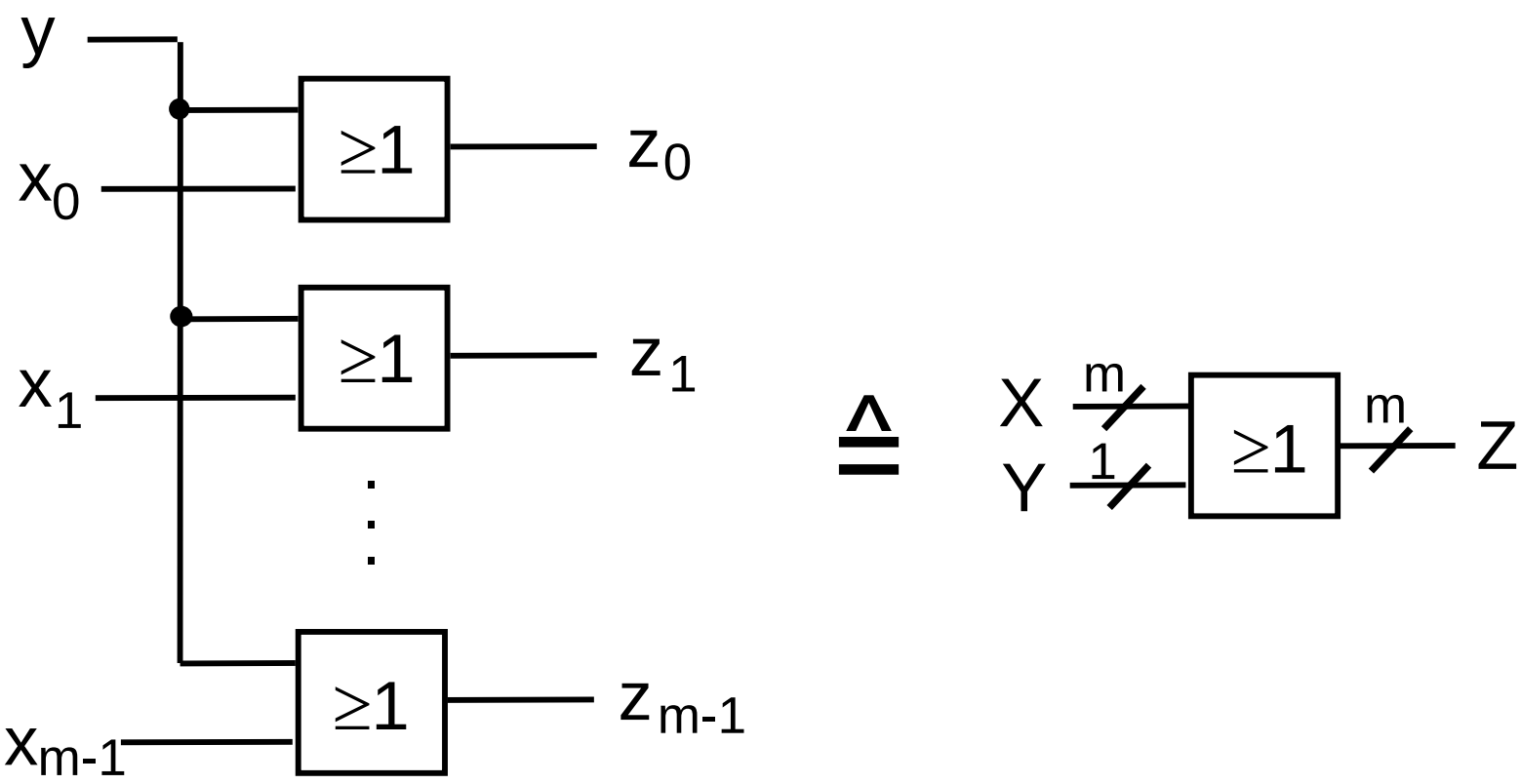
≙



m parallele Leitungen

Register-Transfer-Ebene
Beispiel

- Skalare Operationen sind in gleicher Weise darstellbar.



Register-Transfer-Ebene
Weitere Bausteine

Freie Universität Berlin

- Multiplexer,
- Demultiplexer/Dekodierer,
- Speicherbausteine (ROM, PROM, EPROM, RAM),
- Programmierbare logische Matrizen (PLA, FPLA) und
- Register, Zähler und Arithmetikelemente
- Ähnlich wie bei den kombinatorischen Schaltungen (Schaltnetze) lassen sich auch bei sequentiellen Schaltungen (Schaltwerken) komplexe Bausteine aus elementaren Elementen zusammensetzen.

Register-Transfer-Ebene
Register

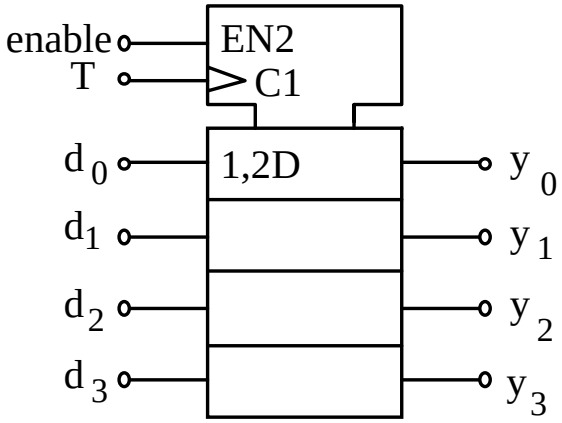
- Lineare Anordnung von Flipflops zur Speicherung mehrerer Bits (Bitvektor).

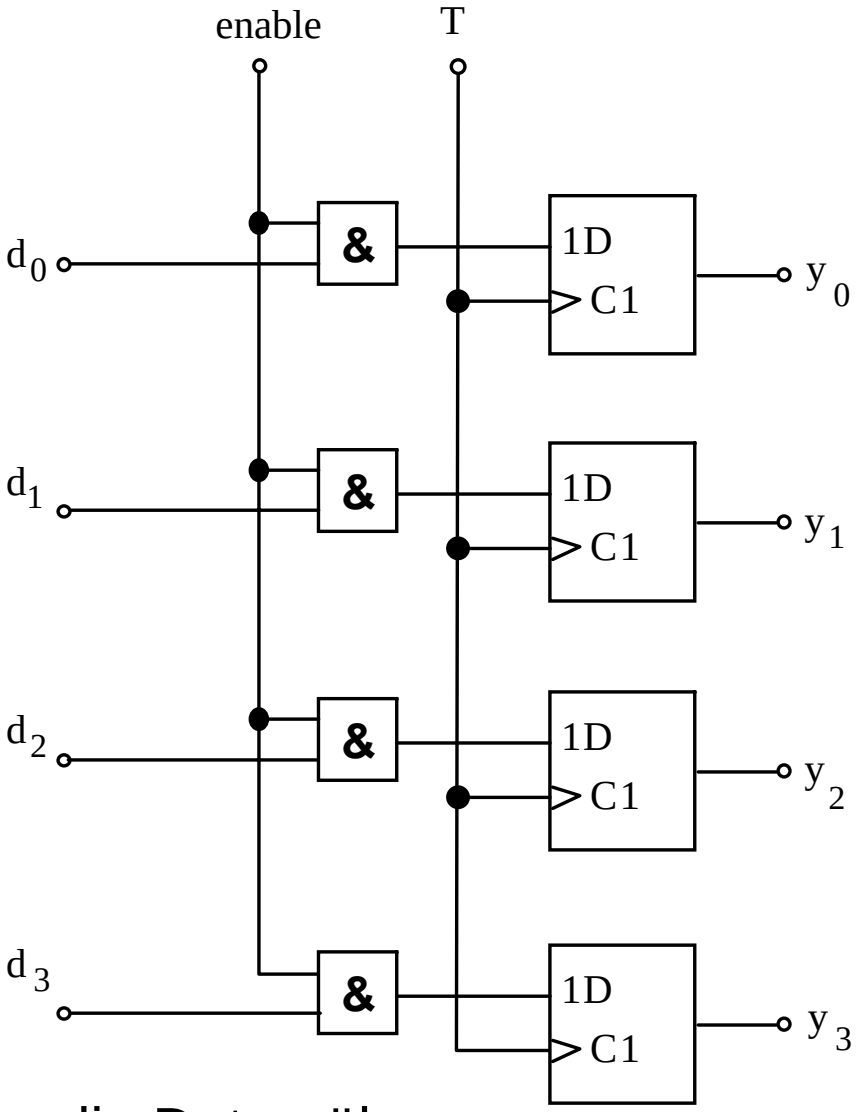
- Die Flipflops werden mit einem gemeinsamen Takt angesteuert.

- Einfachstes Register:
  - Unverkoppelt nebeneinandergesetzte D-Flipflops.

- Im Allgemeinen werden die Flipflops durch zusätzliche gemeinsame Steuersignale beeinflusst.

Register-Transfer-Ebene
4-Bit-Register aus D-Flipflops



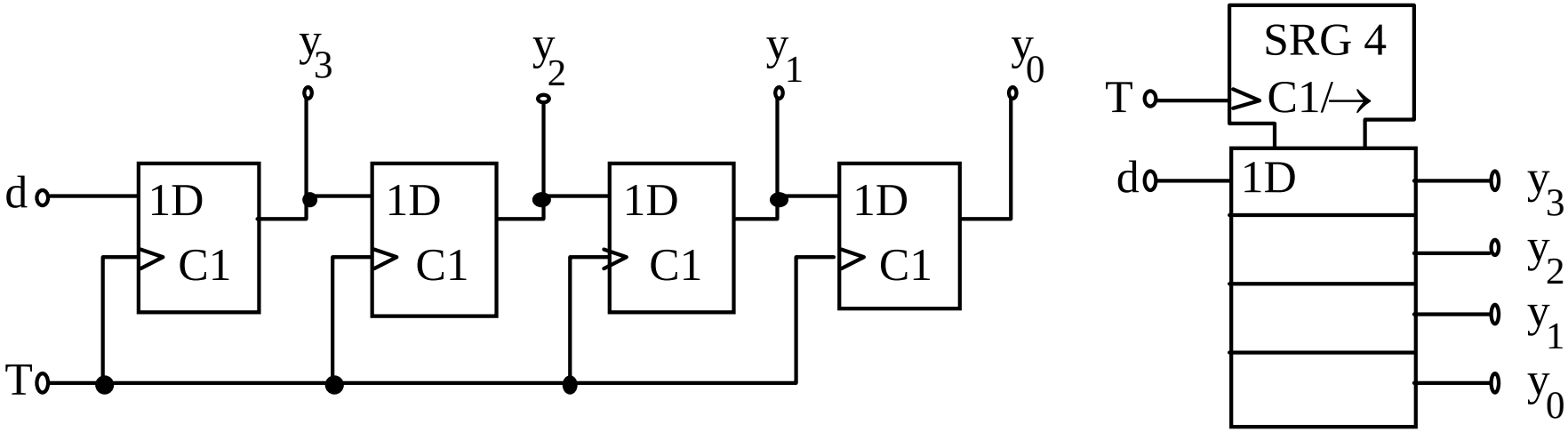


Nur wenn "enable" = 1 ist, werden die Daten übernommen.

Schieberegister
Realisierung

Kette von in Reihe geschalteten Registern oder D-Flipflops

Der Ausgang eines Speicherelements ist jeweils mit dem Eingang des nächsten verbunden.



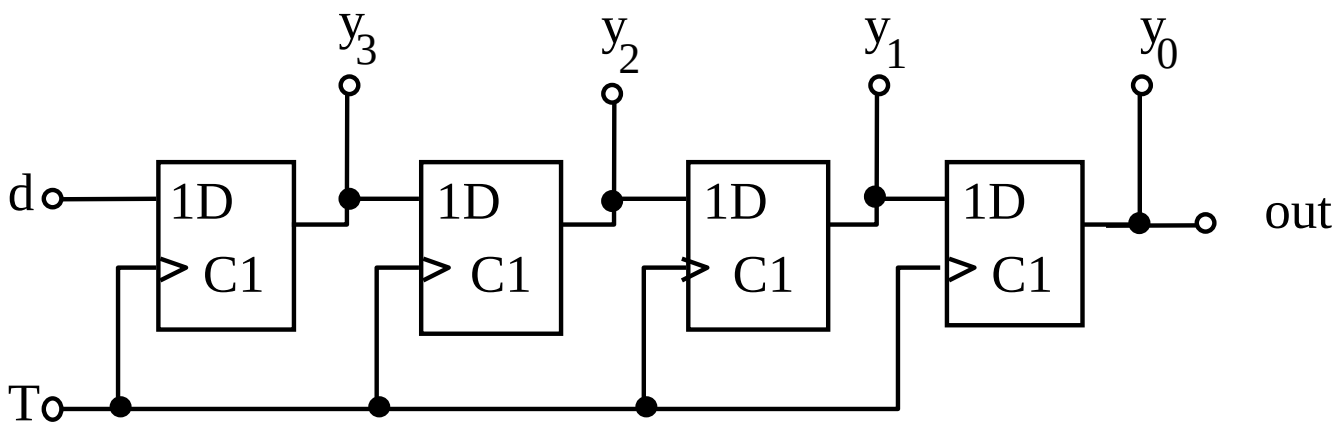
Schieberegister Modifikationen

Zusätzliche Steuereingänge ermöglichen mehr als das reine Schieben von Daten. Weitere mögliche Funktionen:

- parallele Eingabe mehrerer Bits
- wahlweise Rechts-/Links-Schieben
- Verriegeln der Eingänge

- Serien-Parallelwandlung
  Der serielle Bitstrom wird auf den Serieneingang d des Schieberegisters geleitet.

- Ist das Schieberegister gefüllt (nach 4 Takten), stehen alle Bits gleichzeitig an den Parallelausgängen (y_0, y_1, y_2, y_3) zur Verfügung, oder sie lassen sich mit weiteren Takten wieder seriell am Ausgang out entnehmen.



Freie Universität Berlin

Parallel-Serienwandlung

Ein Schieberegister mit parallelen Eingängen kann in ähnlicher Weise zur Parallel-Serien-Wandlung eingesetzt werden.

Warteschlange/Stapelspeicher

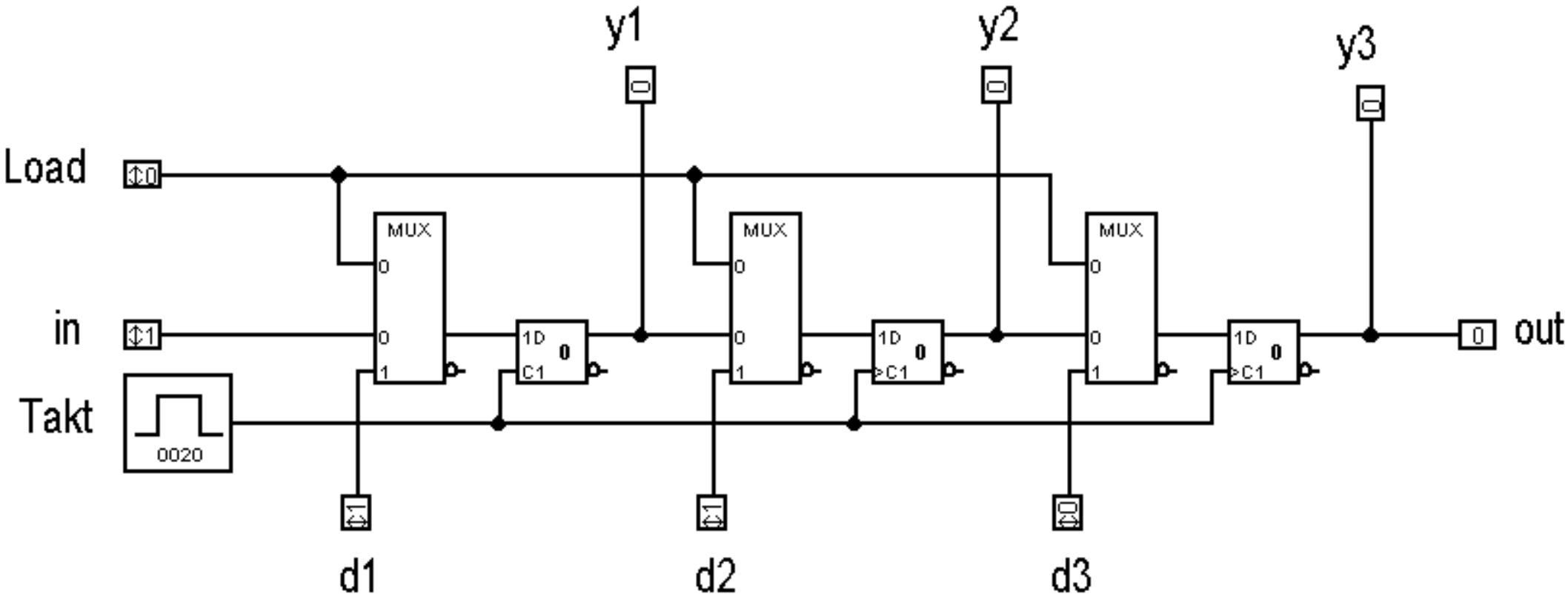
Je nach Schieberichtung kann eine Warteschlange oder ein Stapelspeicher realisiert werden.

Anwendung:

- Netzwerkpuffer zur Übertragungsratenanpassung (z.B. Eingang mit 1 Gbit/s, Ausgang mit 100 Mbit/s)
- 32-bit-Systembus auf 1-bit-USB-Schnittstelle

Schieberegister mit parallelen Eingängen

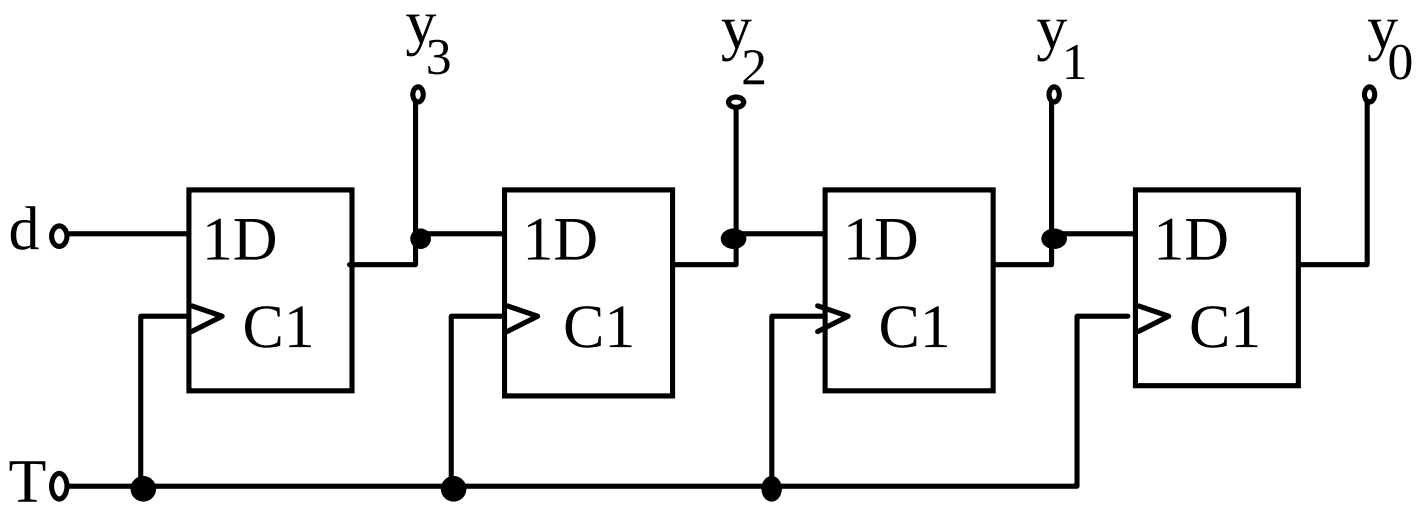
- Über den Load-Eingang kann man auf Parallel-Eingabe umschalten.
- Mit dem nächsten Takt werden dann die Daten d1, d2, d3 parallel geladen und erscheinen an den Ausgängen y1, y2, y3
- Auf diese Weise ist nicht nur eine Serien-Parallel-Wandlung, sondern auch eine Parallel-Serien-Wandlung möglich



Schieberegister Multiplikation/Divison

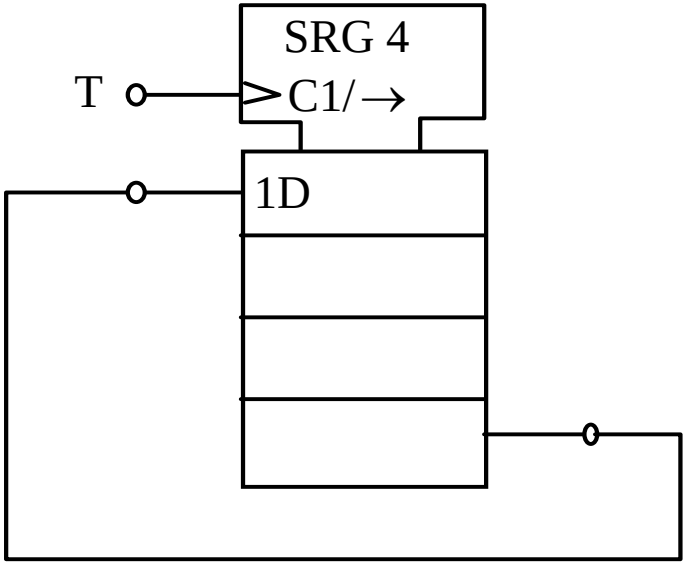
- Division/Multiplikation durch/mit 2

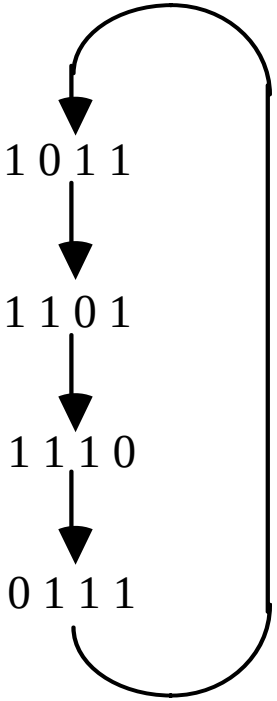
- Interpretiert man die Bitfolge $y_3\ y_2\ y_1\ y_0$ als Dualzahl, entspricht ein Rechtsschieben (mit d = 0) einer Division durch 2 (ohne Rest).
- Schiebt man die Bitfolge ein Bit nach links (mit 0 als neuem letztem Bit), erhält man eine Multiplikation mit 2.



Schieberegister
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

Zähler

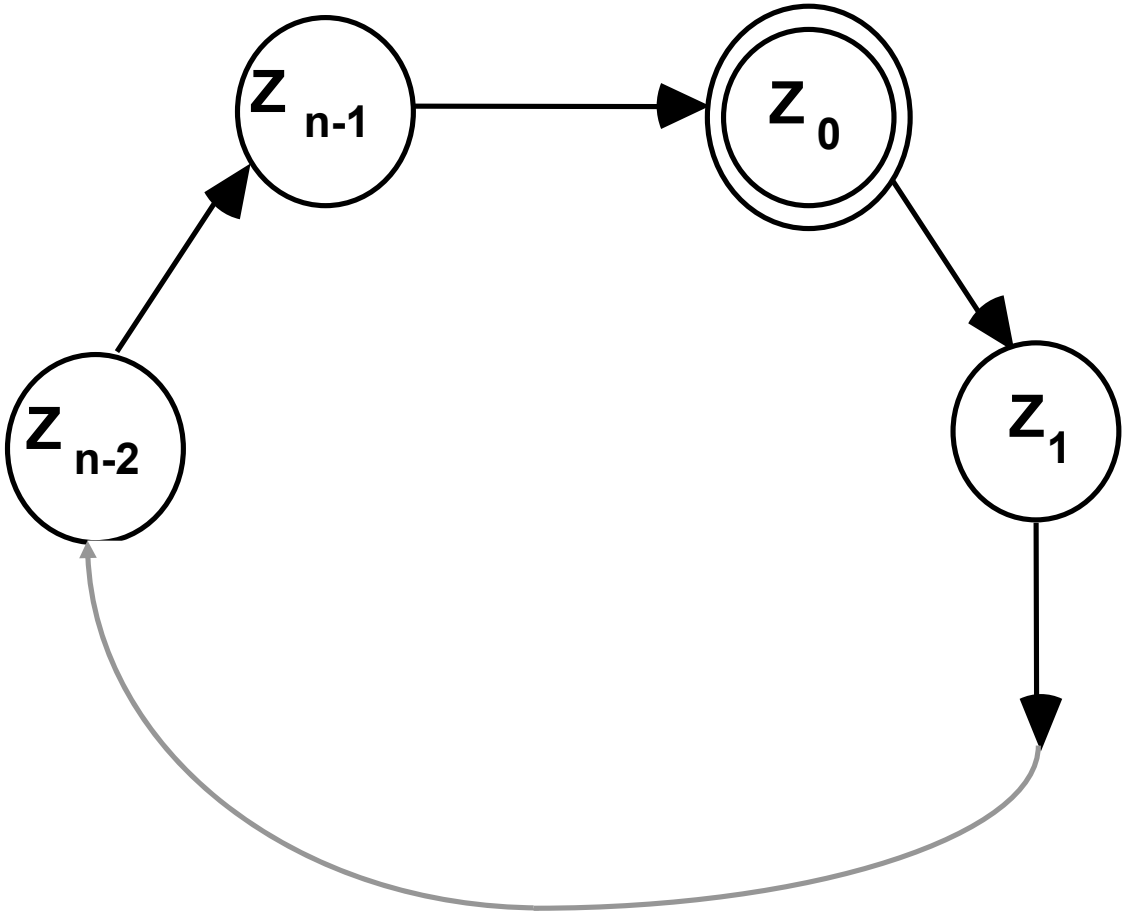
- Zähler erfüllen in digitalen Systemen mehrere Aufgaben:

- 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



Grundlegendes Übergangsdiagramm

Zähler
Anmerkungen

Freie Universität Berlin

Durch Modifizierung der Grundstruktur

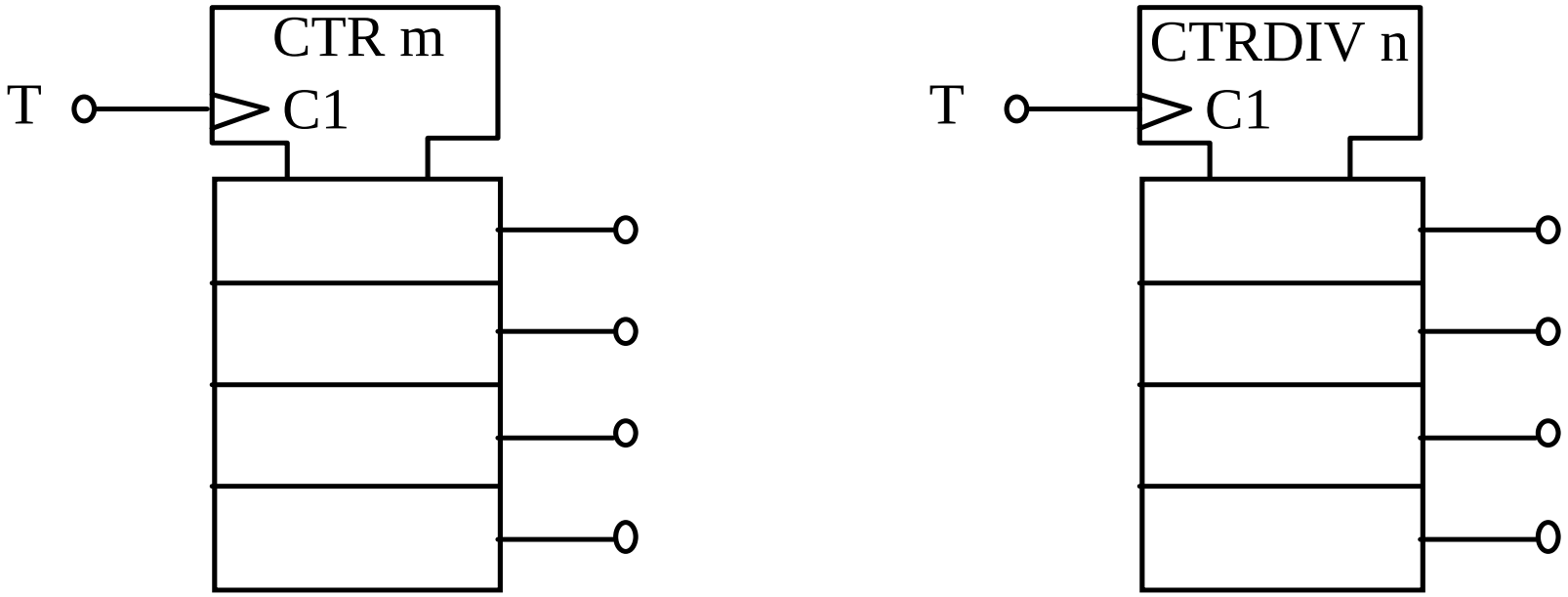
➔ Setz- oder Rücksetzeingänge, Freigabeeingang oder eine Möglichkeit, vorwärts und rückwärts zu zählen.

Sind die Zustände dual kodiert (Zustand Z_i wird mit der Dualzahl i codiert), 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.

Zähler
Schaltsymbole



Schaltsymbole für m-stellige und Modulo-n-Zähler

Synchroner Zähler
Beispiel

Freie Universität Berlin

Entwirft man einen dreistelligen Dualzähler als synchrones Schaltwerk nach der beschriebenen Methode mit Hilfe von D-Flipflops, so erhält man:

$$d_0 = \overline{z}_0$$
$$d_1 = z_1\,\overline{z}_0 \vee \overline{z}_1\,z_0$$
$$d_2 = z_2\,\overline{z}_0 \vee z_2\,\overline{z}_1 \vee \overline{z}_2\,z_1\,z_0$$

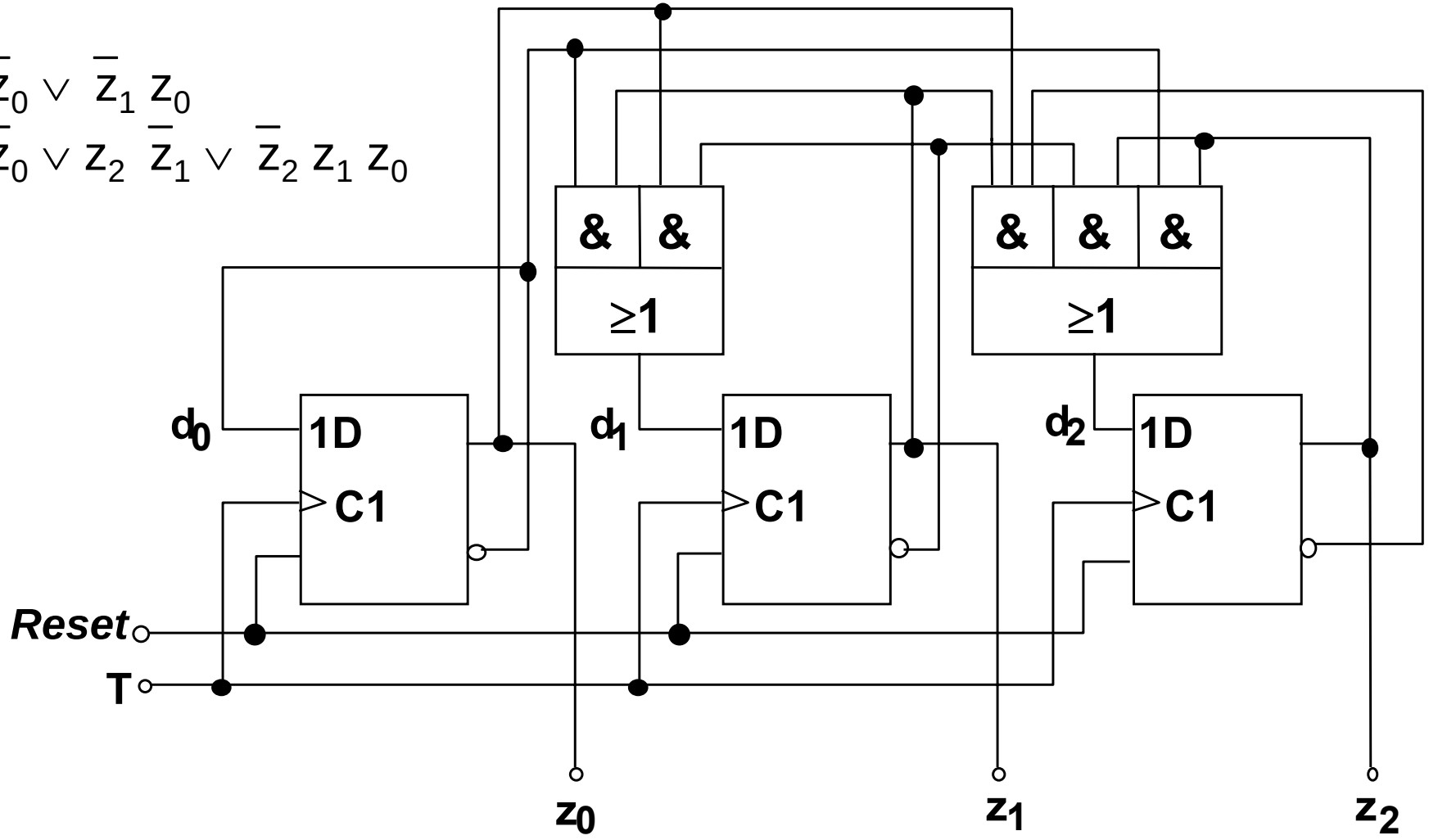
Zunehmende Bitzahl ➔ Größe des Ansteuernetzwerks wächst

Synchroner Zähler
Synchroner 3-stelligen Dualzähler

$d_0 = \overline{z}_0$

$d_1 = z_1\ \overline{z}_0 \vee \overline{z}_1\ z_0$

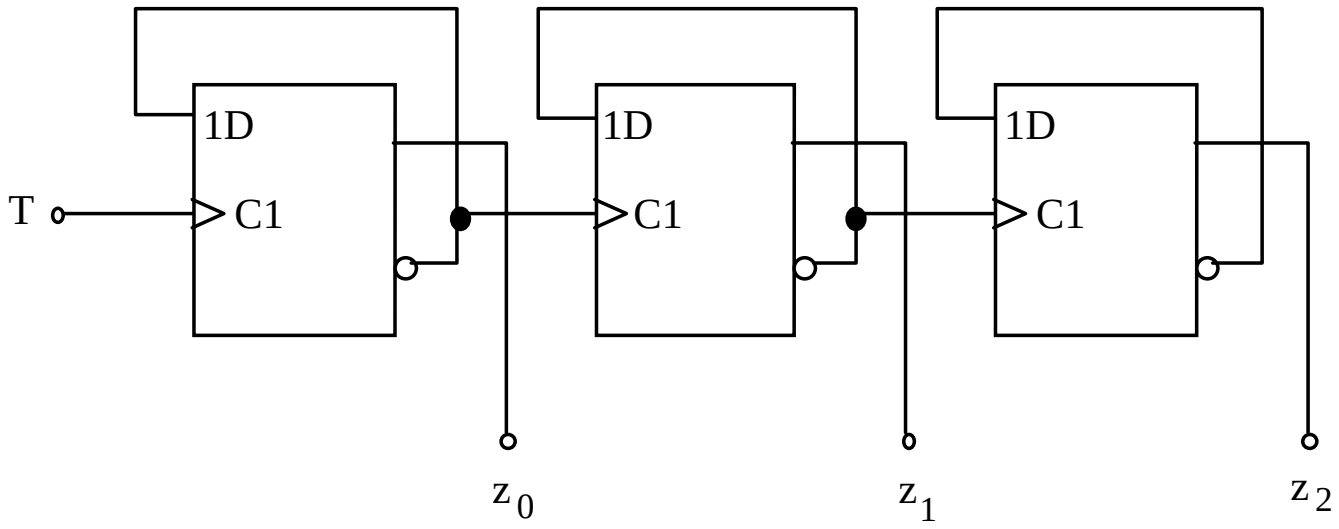
$d_2 = z_2\ \overline{z}_0 \vee z_2\ \overline{z}_1 \vee \overline{z}_2\ z_1\ z_0$



Asynchrone Zähler
Aufbau

Die Größe des Ansteuerschaltnetzes bei synchronen Zählern wächst mit zunehmender Bitanzahl stark an.

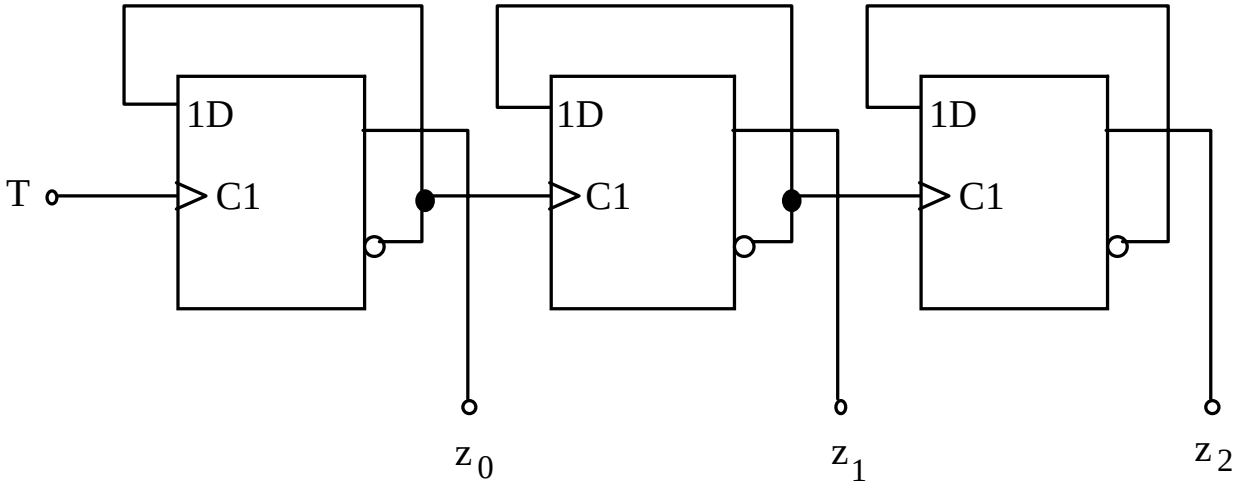
Aus diesem Grund sind asynchrone Zähler (engl.: ripple counter) attraktiv.



Realisierung eines asynchronen 3-stelligen Dualzählers

Asynchroner Zähler
Nachteile

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.



Im Allgemeinen sind für Zähler mit größerer Stellenzahl Asynchronzähler zu langsam und Synchronzähler zu aufwändig.
Lösung:

Zähler in kleinere Einheiten partitionieren, die jeweils synchron arbeiten.
Die kleineren Teilzähler werden dann asynchron miteinander verkoppelt.