Technische Universität Braunschweig

Informatik für Ingenieure – VL 8

Übersicht

Letztes Mal:

Latches und Flip Flops

Einführung zu Endlich Automaten

Heute:

Endlich Automaten

Frequenzteiler, Zähler

Synchrone Schaltungen

Teile des heutigen Vortrags basiert auf der Vorlesungen von
Prof. H Michalik (TU Braunschweig)
Prof. M. Luisier (ETH Zurich)

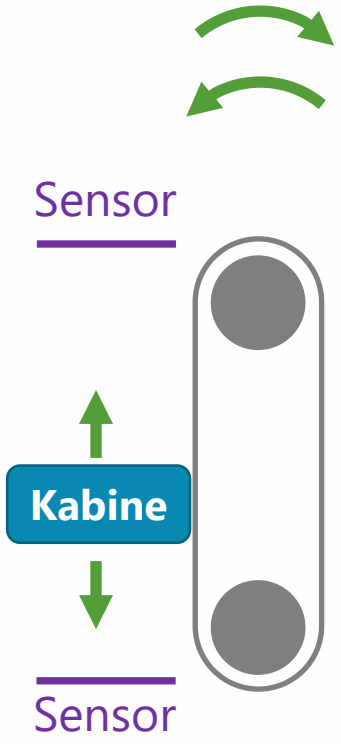
Antriebssteuerung eines einfachen Aufzuges

Es sei die Steuerung für den Antrieb eines Aufzuges zu entwerfen. Der Aufzug pendelt zwischen zwei Stockwerken (Paternosterprinzip).

Die Synchronisation des Antriebes mit dem Einsteigevorgang und Tür öffnen sei hier der Einfachheit halber nicht berücksichtigt.

Die Steuerung habe jeweils zwei Eingänge und zwei Ausgänge:

Eingänge	x_1 Stockwerkkontakt oben x_0 Stockwerkkontakt unten	}	= "1" für Kabine hat Stockwerk erreicht
Ausgänge	y_0 Antrieb	"0" "1"	entspricht aus entspricht an
	y_1 Antriebsrichtung	"0" "1"	entspricht nach unten entspricht nach oben



Beschreibung von Antriebssteuerung

y_0 - Antrieb
y_1 - Richtung

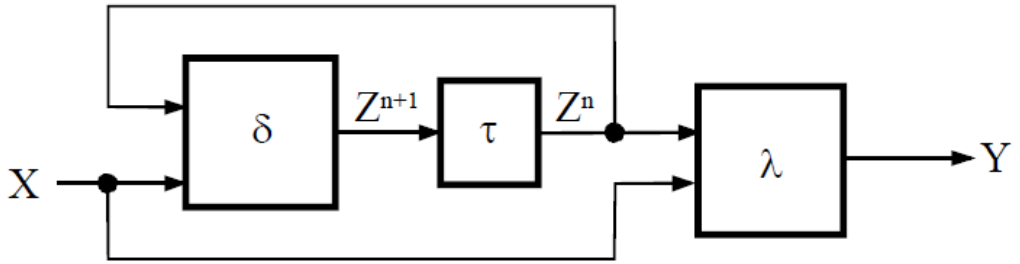
vollständige Funktionstabelle, (die Indizes n, n+1 kennzeichnen den zeitlichen Zustandsablauf)

Eingänge		Zustand	Folgezustand	Ausgänge		Bewegungszustand
x_1	x_0	z_0^n	z_0^{n+1}	y_1	y_0	
0	0	0	0	0	1	Fahrt nach unten
0	1	0	1	1	1	unten angekommen, Antrieb umgeschaltet auf Fahrt nach oben
1	0	0	0	0	1	Fahrt nach unten, gerade oben losgefahren
1	1	0	-	-	0	Fehler: Aufzug angehalten
0	0	1	1	1	1	Fahrt nach oben
0	1	1	1	1	1	Fahrt nach oben, gerade unten losgefahren
1	0	1	0	0	1	oben angekommen, Antrieb umgeschaltet auf Fahrt nach unten
1	1	1	-	-	0	Fehler: Aufzug angehalten

Antriebssteuerung Implementierung

Die Steuerung lässt sich also ganz allgemein mit zwei folgenden kombinatorischen Logikelementen realisieren:

1. Ausgabefunktion: $Y = \lambda(X, Z^n)$
2. Zustandsfunktion: $Z^{n+1} = \delta(X, Z^n)$



Es ergibt sich eine Struktur aus zwei Schaltnetzen wobei das Schaltnetz für die Zustandsfortschaltung δ die Rückkopplung aufweist.

Um hier sicher zu stellen, dass die Zustände auch zeitlich voneinander getrennt wechseln ist eine Zeitverzögerung (tau) in die Rückkopplung eingebaut.

Ohne die Zeitverzögerung wäre die Zustandswechselzeit gleich der Durchlaufzeit durch das Schaltnetz δ (könnte aber minimal = 0 sein).

Was Sind Automaten?

Ein Automat beschreibt ein System das:

auf seinen Eingang reagiert

einen Ausgang produziert, der von dem Eingangssignal
und von dem momentanen Zustand des Systems abhängt

In einem endlichen Automaten (auch finite state machine‘ (FSM) benannt), sind die Menge

der möglichen Eingabezeichen (Eingabealphabet),

der möglichen Ausgabezeichen (Ausgabealphabet) und

der intern gespeicherten Zustände

endlich, d.h. nur vorprogrammierte Lagen werden anerkannt

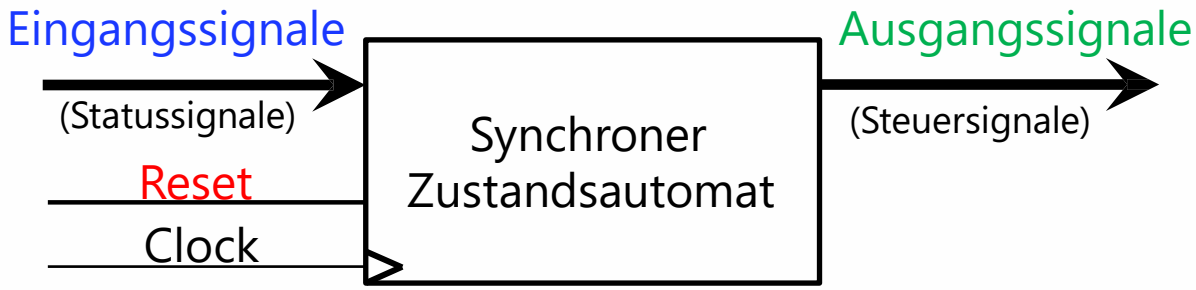
Schaltwerke (Schaltungen mit einer Rückkopplung)
sind typische Beispiele von endlichen Automaten

Typen von Schaltwerke

Synchrone Automaten:

Alle Speicherelemente (Flipflops), die die internen Zustände speichern, besitzen den gleichen Takteingang

Interne Zustandsänderungen laufen synchron mit dem gemeinsamen Taktsignal



Asynchrone Automaten: (hier nicht betrachtet)

Sie haben kein gemeinsames Taktsignal

Zustandsänderungen werden durch Änderungen der Eingangssignale initiiert

Formale Beschreibung von Automaten

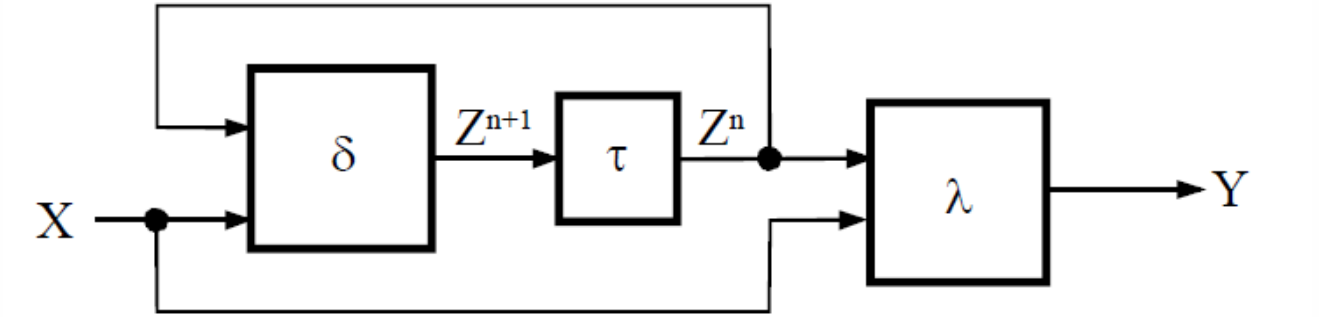
Ein endlicher Automat wird durch ein **6-Tupel** charakterisiert:

$X = (x_1, x_2, \cdots, x_e)$	Eingabealphabet mit e Eingängen x_i, die durch binäre Eingangsvariablen 0,1 repräsentiert werden
$Y = (y_1, y_2, \cdots, y_b)$	Ausgabealphabet mit b Ausgängen y_i, die als Bits mit Wert 0,1 dargestellt werden
$Z = (z_1, z_2, \cdots, z_m)$	Zustandsmenge mit m inneren Zustandsvariablen z_i, die einen Wert {0,1} haben können
$Z_0 \in Z$	Anfangszustand
$f_{c1}: (X_n, Z_n) \rightarrow Z_{n+1}$	Zustands-, Übergangsfunktion
$f_{c1}: (X_n, Z_n) \rightarrow Y_n$	Ausgabe-, Ausgangsfunktion

Mealy Automat

Der **Mealy-Automat** ist durch seine Eingänge, Ausgänge, Zustände sowie den beiden Schaltnetzen λ für die Erzeugung der Ausgänge und δ für die Erzeugung der Folgezustände als „**Automatenfunktion A“** vollständig definiert:

$$A_{mealy} = [X, Y, Z, \lambda, \delta]$$



Mealy Schaltnetz

y_0 - Antrieb
y_1 - Richtung

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Für unser Beispiel ergibt sich für das Schaltnetz δ zur Fortschaltung:

z_0^{n+1} x_1x_o:	00	01	11	10
z_0:				
0	0	1	X	0
1	1	1	X	0

$$z_0^{n+1} = x_0 \vee (\overline{x}_1 \wedge z_0)$$

Für das Schaltnetz λ zur Erzeugung der Ausgänge ergibt sich:

$$y_0 = \overline{x_0 \wedge x_1}$$

$$y_1 = x_0 \vee (\overline{x}_1 \wedge z_0)$$

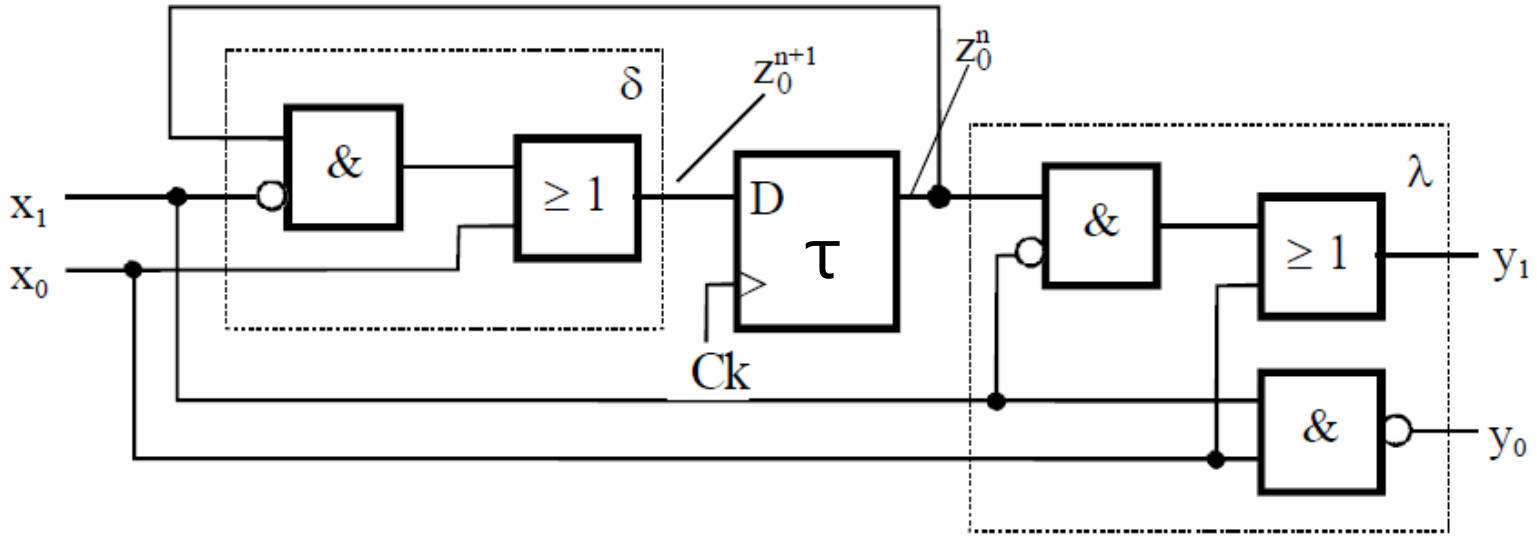
da der Antrieb nur aus sein soll (y_0=0), wenn x_1 und x_0 beide „1“ sind (Fehlerfall). Die Antriebsrichtung y_1 entspricht logisch dem Zustand z_0^{n+1}.

Mealy Schaltbild

y_0 - Antrieb
y_1 - Richtung

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Für unser Beispiel ergibt sich damit das folgende Schaltbild eines Mealy-Automaten:



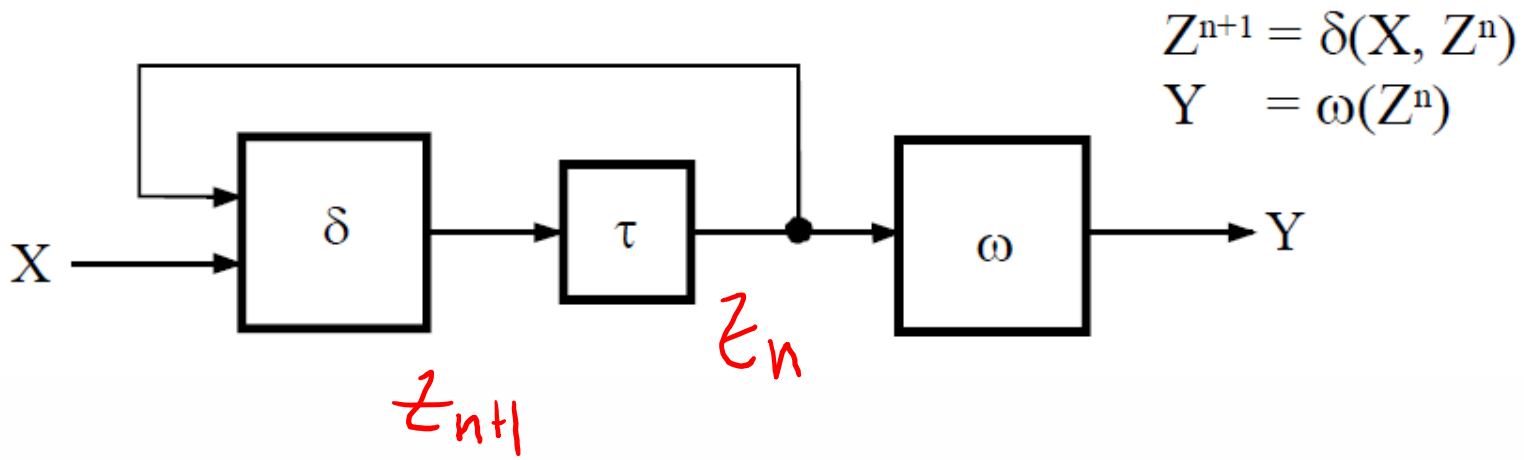
Die Verzögerung τ kann zum Beispiel mit einem flankengesteuerten D-Flipflop realisiert werden.

Moore Automat

Beim Moore Automaten basiert die Ausgangsfunktion nur auf dem aktuellen Zustand Z^n.

Moore und Mealy Automaten können ineinander überführt werden, da der Eingangsvektor X durch das Netz δ im Zustandsvektor Z enthalten ist. Die Ausganstransformation (hier ω) muss somit entsprechend gegenüber dem Mealy Automaten angepasst werden.

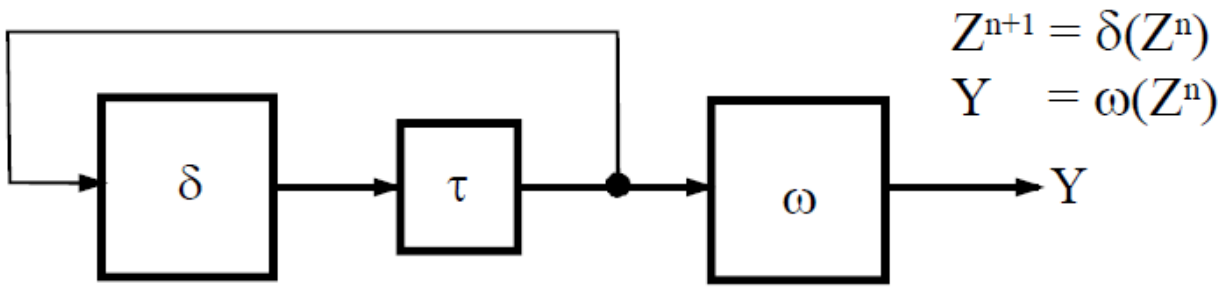
$$A_{Moore} = A[X, Y, Z, \delta, \omega]$$



Autonomer Automat

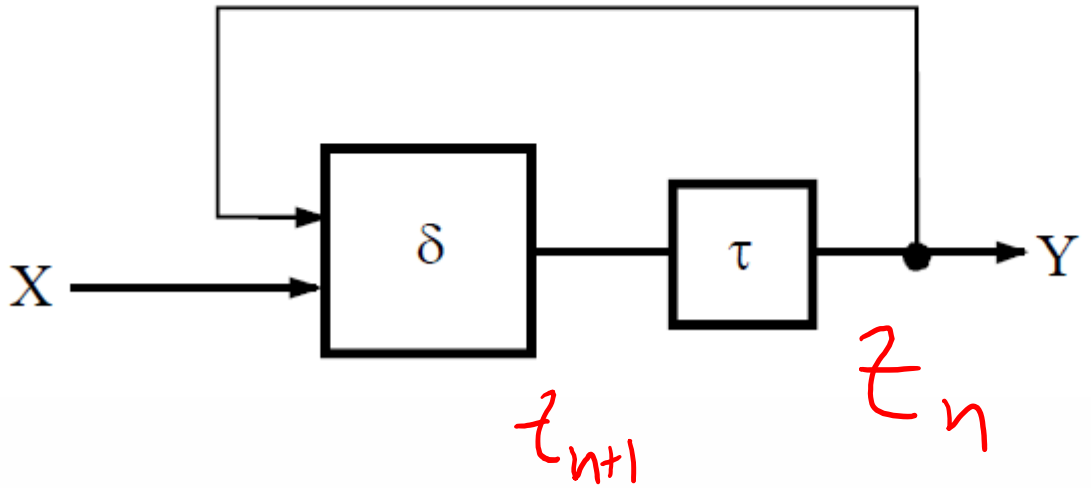
Beim autonomen Automaten gibt es keine Eingangssignale.

$$A_{Aut} = A[Y, Z, \delta, \omega]$$



Medwedjew Automat

$$A_{Med} = A[X, Y, Z, \delta]$$



$$Z^{n+1} = \delta(X, Z^n)$$
$$Y = Z^n$$

Beschreibung der Funktion eines Automaten

Es gibt verschiedene, **äquivalente** Beschreibungsmöglichkeiten für die Funktion eines endlichen Automaten:

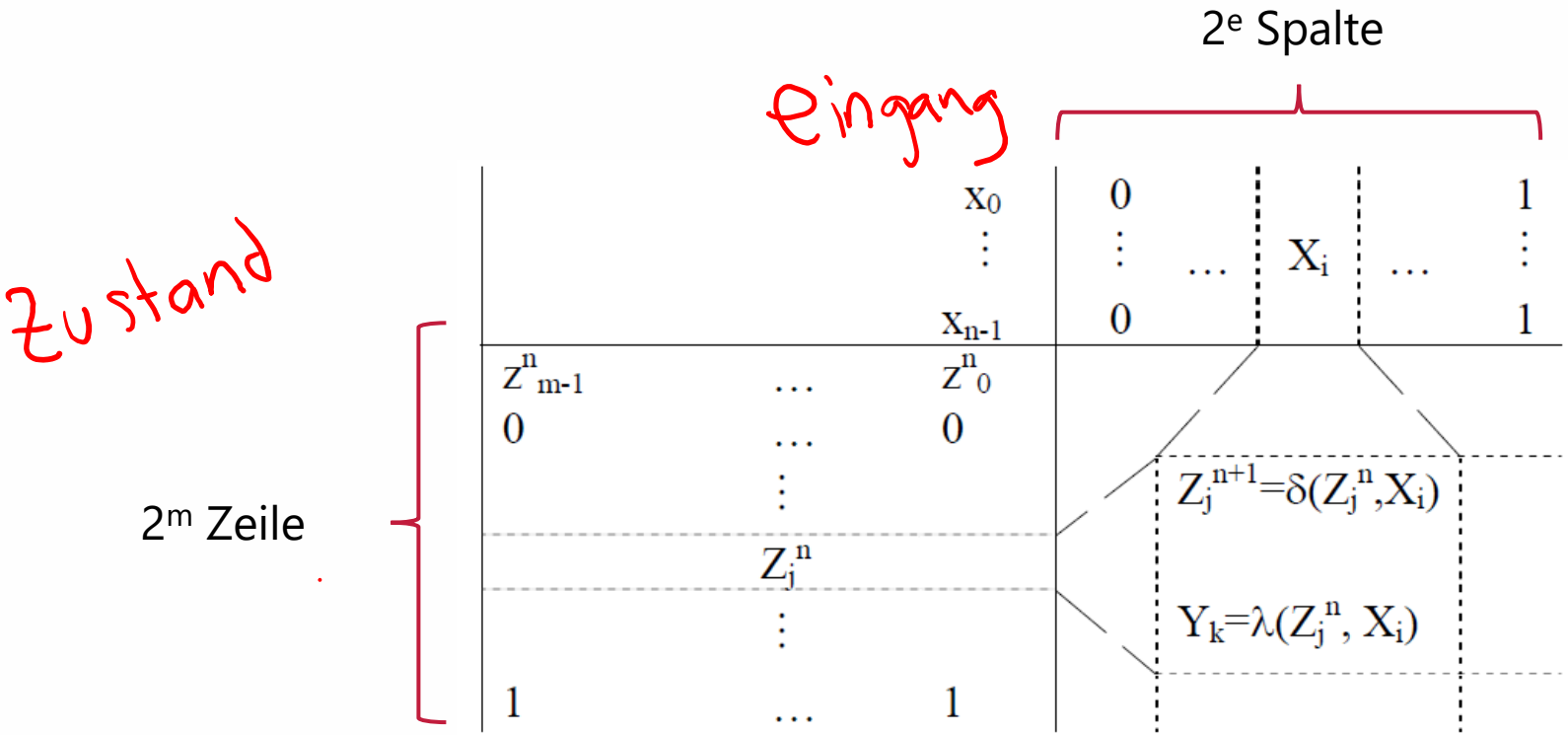
- Ausgangs- und Zustandsfunktion
- Automatentabellen (oder Zustandsfolgetabellen)
- Zustandsdiagramme oder Zustandsgraphen
- Karnaugh-Diagramme (wenn Minimierung nötig)

Aus einem **Zustandsdiagramm** kann zuerst eine **Automatentabellen** konstruiert werden. Wenn diese vorhanden ist, kann ein **Karnaugh-Diagramm** für jede innere Zustandsvariable und für jeden Ausgang aufgesetzt werden. Damit werden die Gleichungen der **Ausgangs- und Zustandsfunktion** minimiert. Am Ende wird ein Schaltwerk erzeugt

Automatentabelle Beschreibung

In der **Automatentabelle** sind die Zustandsübergangsfunktion und die Ausgabefunktion vollständig dargestellt, z. B. Mealy-Automat, m Zustände, n Eingänge:

e: Anzahl Eingangsvariablen **m**: Anzahl Zustandsvariablen → 2^{e+m} Kombinationen



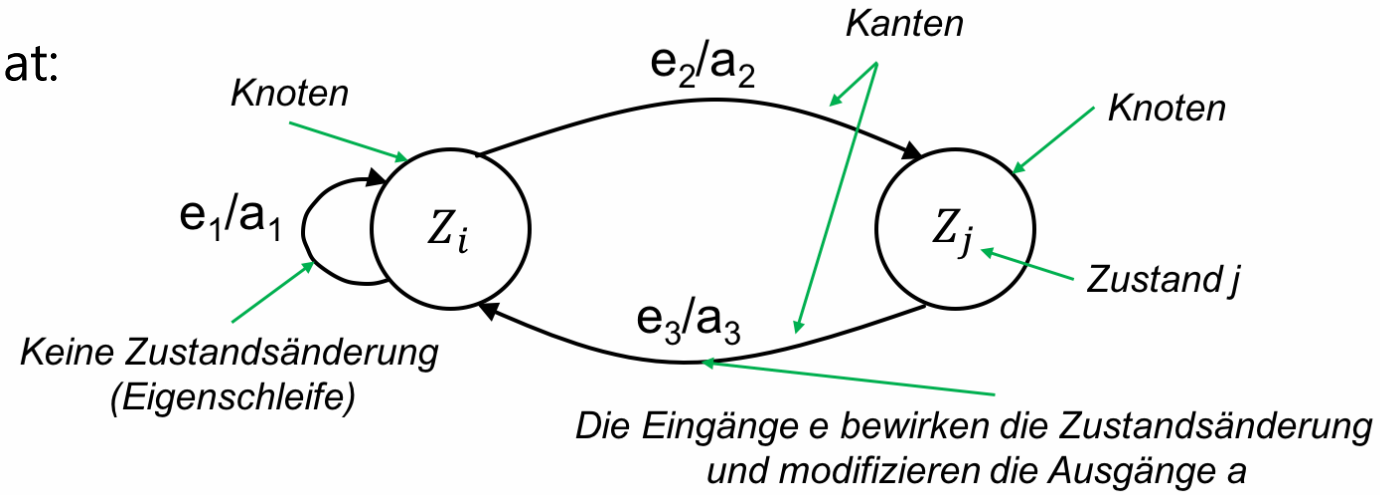
Aufzugssteuerung Automatentabelle

Zustand	Z_0	X_0	0	1	0	1
		X_1	0	0	1	1
Z_0	0		$Z^{n+1} = (0)$ $Y = (0,1)$	$Z^{n+1} = (1)$ $Y = (1,1)$	$Z^{n+1} = (0)$ $Y=(0,1)$	$Z^{n+1} =(-)$ $Y=(-,0)$
Z_1	1		$Z^{n+1} = (1)$ $Y = (1,1)$	$Z^{n+1} =(1)$ $Y = (1,1)$	$Z^{n+1} = (0)$ $Y = (0,1)$	$Z^{n+1} =(-)$ $Y = (-,0)$

Zustandsdiagramm

Ein Zustandsdiagramm (oder *Zustandsgraph*, *Automatengraph*, ‚*state graph*', ‚*state transition diagram*') ist eine graphische (äquivalente) Darstellung der Folgezustandstabelle, die aus Knoten und gerichteten Kanten besteht.

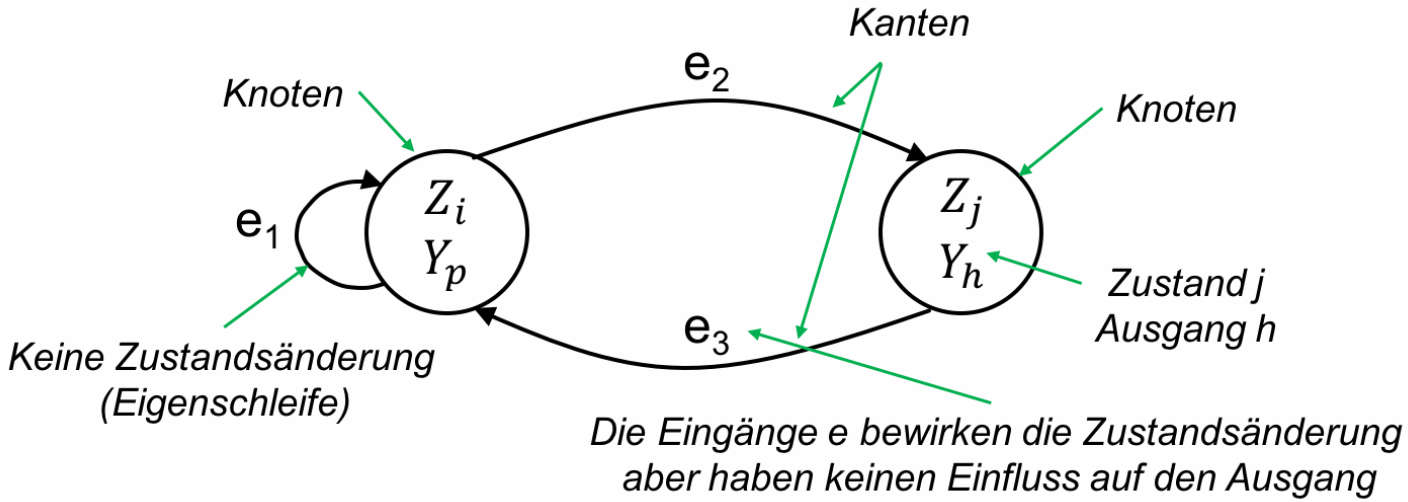
Mealy-Automat:



Die **Knoten** (Kreise) bezeichnen die internen Zustände, die **Kanten** den Übergang zwischen zwei Zuständen. Die Eingangskombination ***e***, die die Zustandsänderung bewirkt, und der Ausgang ***a*** werden an der jeweiligen Kante vermerkt.

Zustandsdiagramm

Moore-Automat:



Bei **Moore-Automaten** bezeichnen die **Knoten** sowohl die internen Zustände wie die Ausgänge, die nur von **Z_n** abhängen

Um einen Automaten zu entwerfen sollte zuerst ein **Zustandsdiagramm** erzeugt werden, dann die entsprechende **Folgezustandstabelle**, λ/δ und daraus ein **Schaltwerk**.

Wenn das Schaltwerk eines Automaten vorgegeben ist, wird die Reihenfolge vertauscht: zuerst f_{C1}/f_{C2}, dann die Folgezustandstabelle und schlussendlich das Zustandsdiagramm

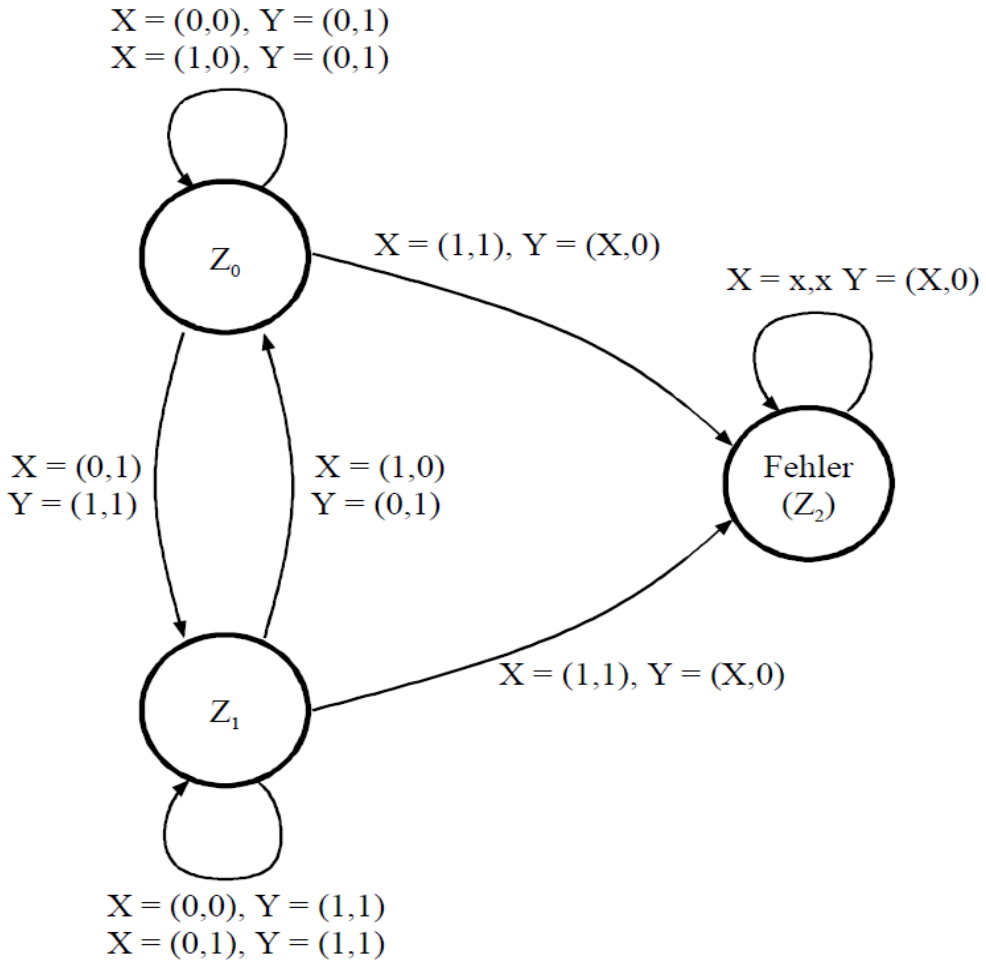
State Diagramm der Aufzugssteuerung

y_0 - Antrieb
y_1 - Richtung

$X=(x_1,x_0)$ $Y=(y_1,y_0)$

Ist es Mealy oder Moore?

Mealy, weil der Ausgang vom Zustand ***und*** dem Eingang abhängt



Beschreibungsformen für Automaten

Wir wollen im folgenden die Automaten hauptsächlich durch Zustands-graphen erfassen. Diese Darstellung ist sehr kompakt, lässt aber auf den ersten Blick nicht klar werden, ob alle Zustandsänderungen erfasst sind oder ob die Änderungen widerspruchsfrei sind.

Es muss daher der Graph auf Vollständigkeit und Widerspruchsfreiheit geprüft werden:

Vollständigkeit:

Der Zustandsgraph muss alle Eingangsbelegungen X_i in den wegführenden Kanten oder in den Eigenschleifen in jedem Knoten enthalten. Dabei werden wie bei den Schaltnetzen nicht relevante Belegungen mit "don't cares" versehen.

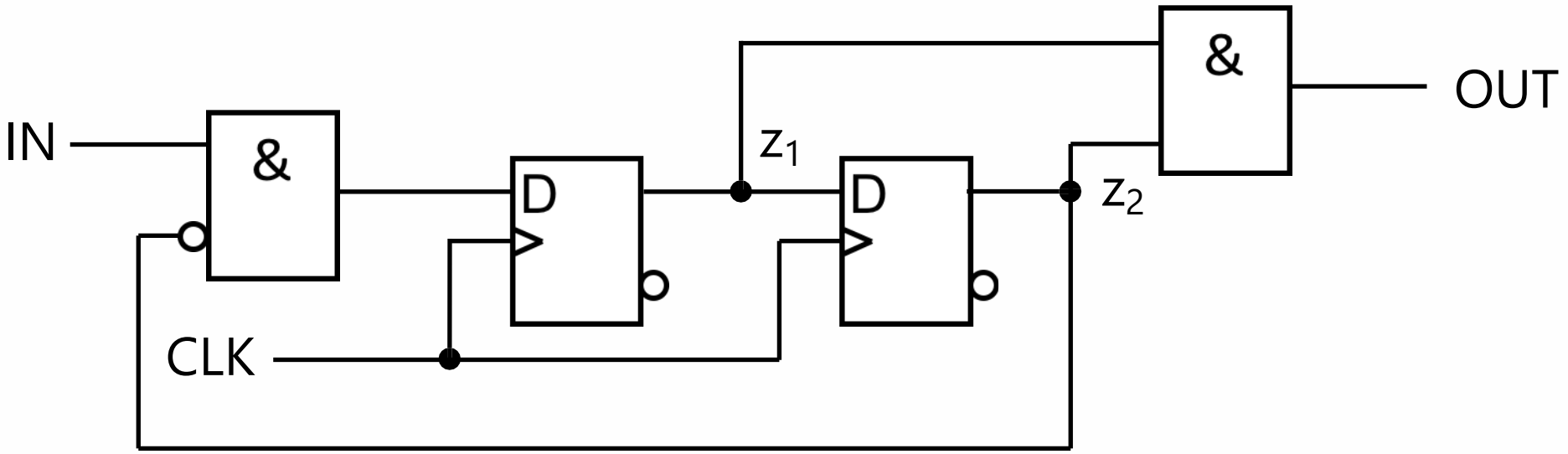
Widerspruchsfreiheit:

Der Graph ist nicht widerspruchsfrei, wenn zwei wegführende Kanten oder die Eigenschleife gleiche Belegungen X enthalten.

Analyse eines Automaten: Schaltwerk

Ein Schaltwerk ist vorgegeben und sollte mit der folgenden Vorgehensweise analysiert werden:
Bestimmung der Ausgabe- und Zustandfunktion

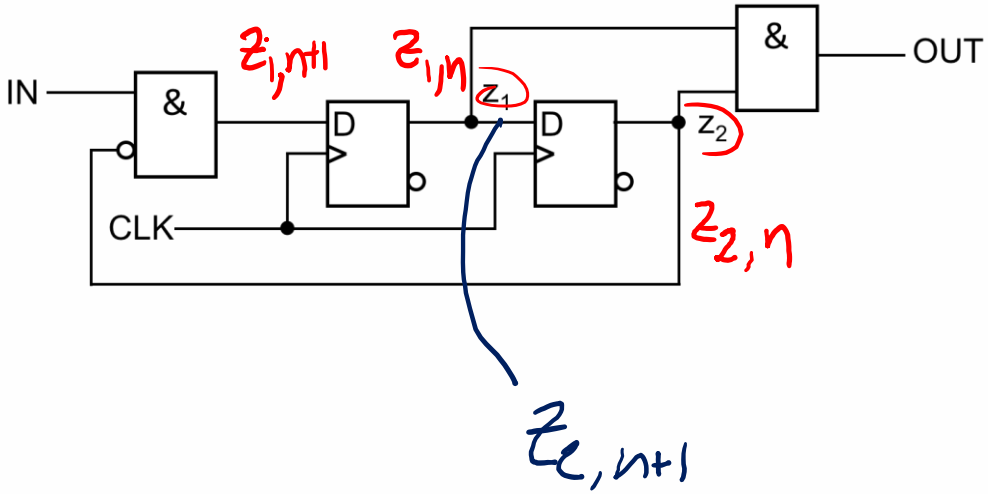
Beispiel:



Analyse eines Automaten: Schaltwerk

Automatendaten:

- Eingabealphabet: X_n=IN
- Ausgabealphabet: Y=OUT
- Zustandsmenge: Z=(z_1,z_2)



Fragen:

- Automatentyp: Moore
- Anzahl Zustände: $2^2 = 4$
- Ausgabefunktion λ:

$$Y_n = z_{1,n} \wedge z_{2,n}$$

- Zustandfunktion δ:

$$z_{1,n+1} = x_n \wedge \overline{z_{2,n}}$$

$$z_{2,n+1} = z_{1,n}$$

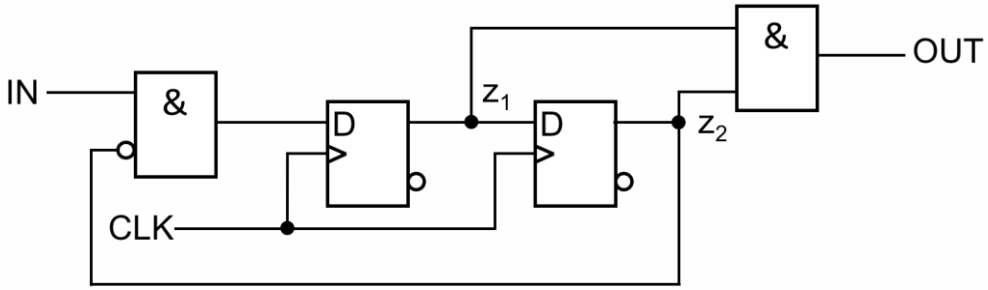
Analyse eines Automaten: Schaltwerk

Automatendaten:

Eingabealphabet: X=IN

Ausgabealphabet: Y=OUT

Zustandsmenge: Z=(z_1,z_2)



Fragen:

Automatentyp: Moore

Anzahl Zustände: $2^2 = 4$

Ausgabefunktion λ:

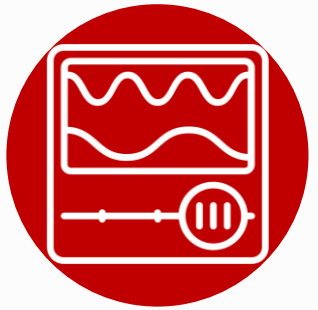
$$Y_n = z_{1,n} \wedge z_{2,n}$$

Zustandfunktion δ:

$$z_{1,n+1} = x_n \wedge \overline{Z_{2,n}}$$

$$z_{2,n+1} = Z_{1,n}$$

Frequenzteiler



Warum Frequenzteiler?

Frequenzteiler sind elektronische Bauelemente, die eine Eingangsfrequenz in einem bestimmten ganzzahligen Teilverhältnis verringern. Die Schaltungen dazu werden digitaltechnisch realisiert, z.B. mit Flipflops

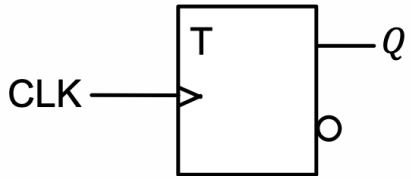
f_E: Eingangsfrequenz
f_T: Geteilte Ausgangsfrequenz
K: Teilverhältnis

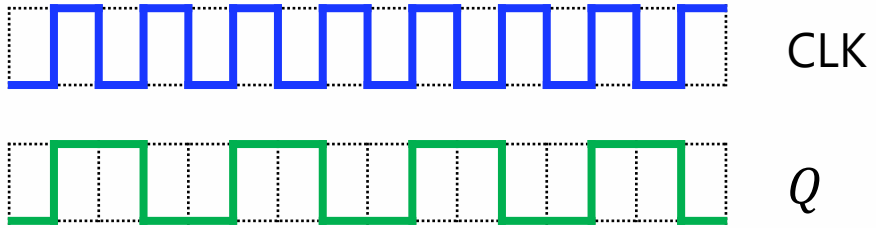
Frequenzteiler werden häufig dazu verwendet, zeitgenaue Taktgeber zu bauen, wobei ein niederfrequenter Takt aus einem präzisen höherfrequenten Takt abgeleitet wird. Als Referenztaktgeber für **Sekundentakte** werden sehr oft Quarzoszillatoren (fE=215 Hz) oder sogar das Standard 50-Hz Wechselstromnetz benutzt.

Flipflops als Frequenzteiler

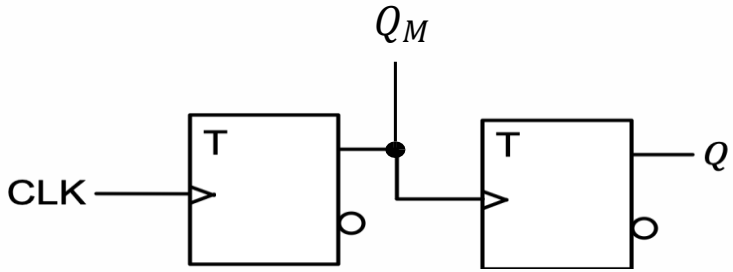
Flipflops können verwendet werden, um die Periode **T** eines sich widerholenden Taktsignals (Clock) zu reduzieren. **T-Flipflops** mit einem einzigen Eingang sind als **Frequenzteiler** sehr geeignet. **JK-Flipflops** mit J=K=1 auch.

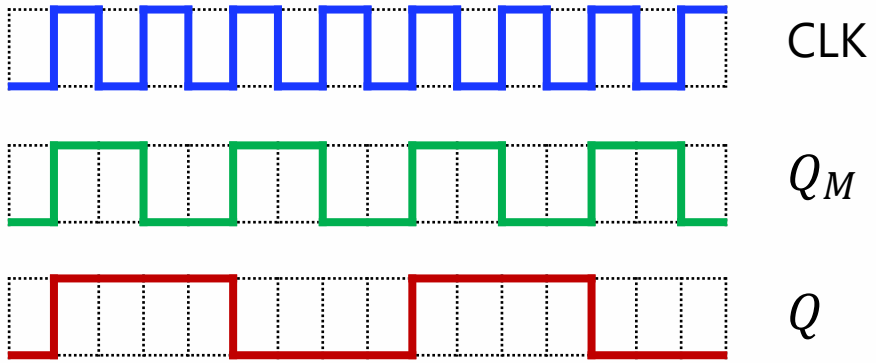
Frequenzreduktion um einen Faktor 2





Frequenzreduktion um einen Faktor 4

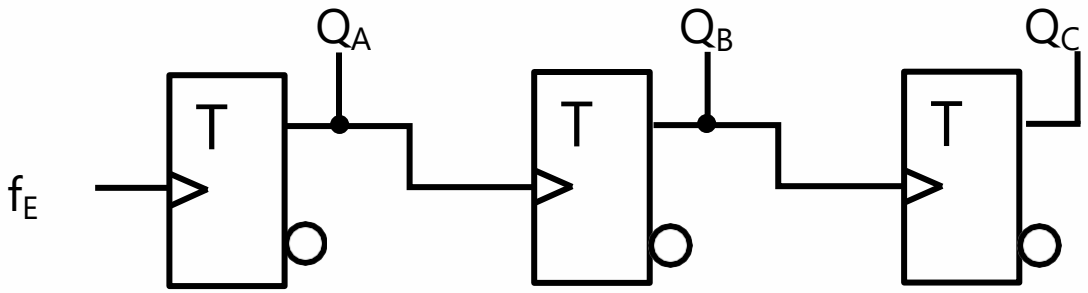


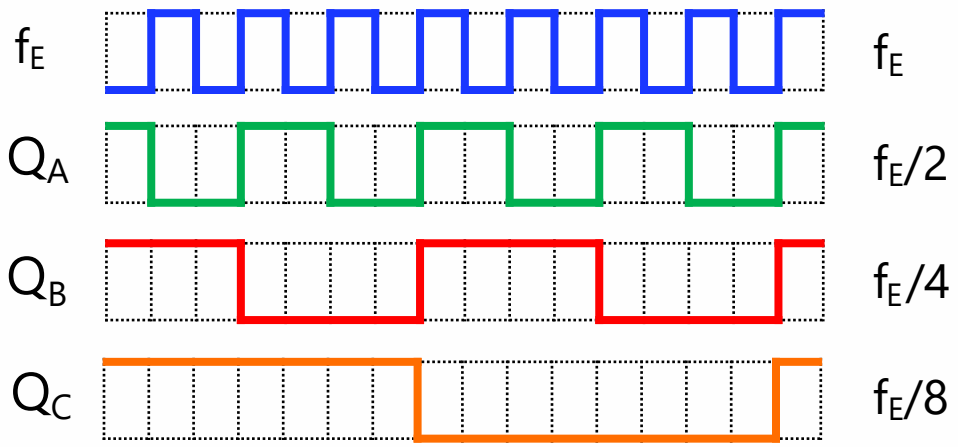


Allgemeiner Frequenzteiler mit T-Flipflops

In den meisten Anwendungen werden ***n* T-Flipflops** kaskadiert, um eine beliebige Frequenzreduktion von 2^n zu erhalten

Frequenzreduktion um einen Faktor 8



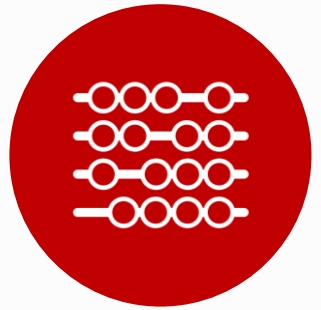


Mögliche Teilverhältnisse:

$$f_T = f_E / 2^n$$

Mit f_E: Eingangsfrequenz, f_T: geteilte Frequenz und n: Anzahl der Flipflops

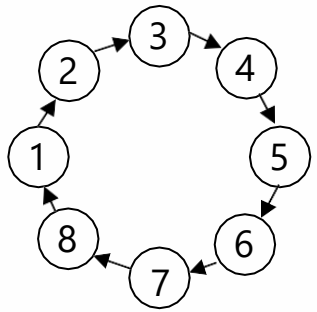
Zähler



Was heisst zählen?

Zählen ist im allgemeinen Sinn das Addieren (Vorwärtszählen) oder Subtrahieren (Rückwärtszählen) einer fortlaufenden 1 bis der Zählvorgang beendet ist. Zähler unterscheidet man nach dem zu verwendeten Code und nach der Zählrichtung.

3-Bit Dualzähler



Zustandsgraph
Vorwärtszähler

Zustands-Nr	Dualzahl	III	II	I
1	0	0	0	0
2	1	0	0	1
3	2	0	1	0
4	3	0	1	1
5	4	1	0	0
6	5	1	0	1
7	6	1	1	0
8	7	1	1	1

Rückwärts

Vorwärts

Stelle I kippt bei jedem Zustandswechsel,
Stelle II bei jedem 2. Zustandswechsel,
Stelle III bei jedem 4. Zustandswechsel

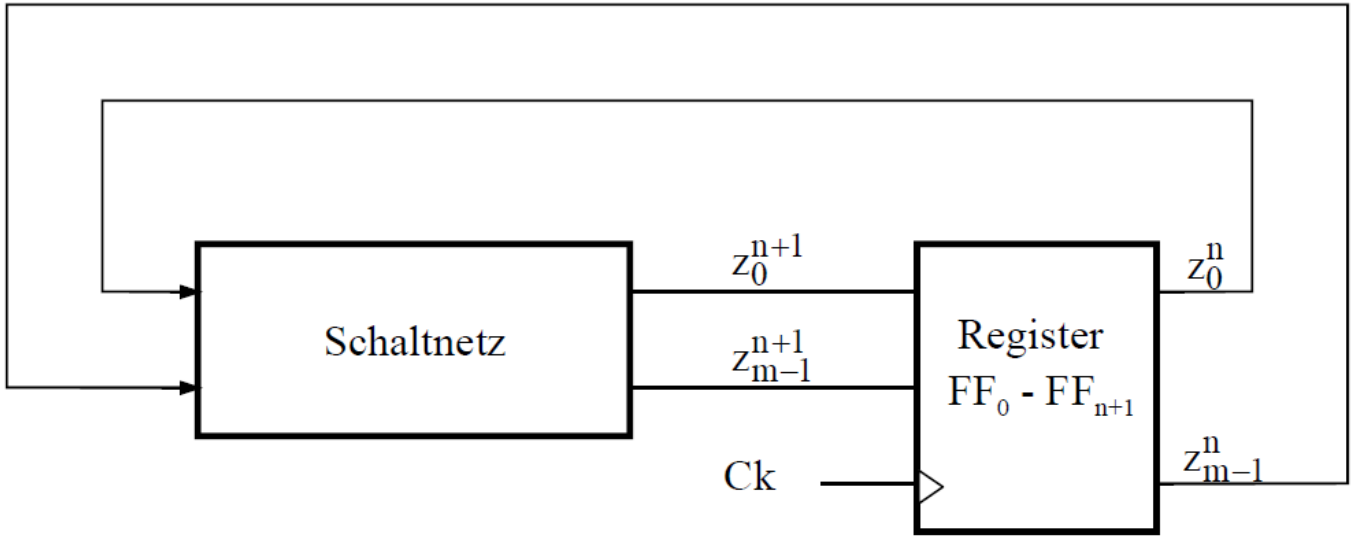
Zähler

Zähler können als autonome Automaten mit rückgekoppelten Schaltnetzen realisiert werden.

Als Register bezeichnet man eine Reihe von Flipflops mit gemeinsamen Takt zur Speicherung eines binären Zustandsvektors.

Dabei ergibt sich der Folgezustand aus dem vorhergehenden Zustand: $Z^{n+1} = \delta(Z)^n$.

Je nach Schaltnetzrealisierung können damit unterschiedliche Zählstrukturen aufgebaut werden.



Binärzähler

Mit einem Binärzähler können 2^n verschiedene Zustände generiert werden, wobei die Ausgangskombination des Folgezustands jeweils um 1 inkrementiert wird (siehe Tabelle). Andere Zählerkönnen andere Zählcodes realisieren, bei denen die Zustandsanzahl auch kleiner als 2^n sein kann.

Für die Funktionsbeschreibung eines Zählers kann die Übergangs- oder Flusstabelle aufgestellt werden. Dargestellt werden die Zustände mit ihren zugehörigen Folgezuständen. In der Tabelle ist als Beispiel die Übergangstabelle eines vorwärts-zählenden Binärzählers angegeben.

Zustand	**Zustandsvariable**				**Folge-zustand**	**Zustandsvariable**			
	z^n_{m-1}	…	z^n_1	z^n_0		z^{n+1}_{m-1}	…	z^{n+1}_1	z^{n+1}_0
Z_0	0	…	0	0	Z_1	0	…	0	1
Z_1	0	…	0	1	Z_2	0	…	1	0
⋮	⋮		⋮	⋮	⋮	⋮		⋮	⋮
Z_{2^n-2}	1	…	1	0	Z_{2^n-1}	1	…	1	1
Z_{2^n-1}	1	…	1	1	Z_0	0	…	0	0

Binärzähler

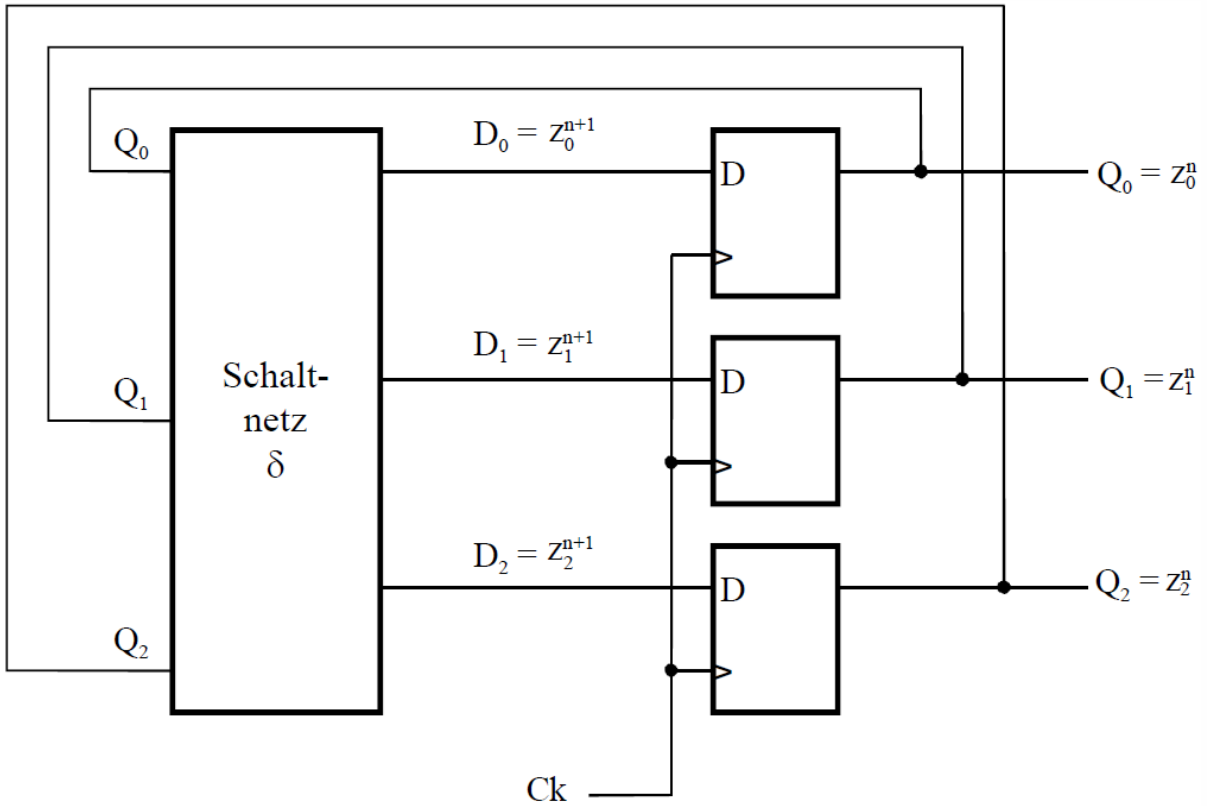
Aus dieser Zustandsübergangstabelle kann man direkt die Bool'schen Gleichungen des Schaltnetzes ablesen. Bei einem n-stelligen Binär-Zähler enthält die Flusstabelle 2^n verschiedene Eingangs-kombinationen.

Pro Spalte in der Flusstabelle ergeben sich $½ \cdot 2^n$ zu realisierende Einsen; das bedeutet, dass bei n Spalten $n \cdot 2^{n-1}$ UND-Schaltungen mit je n Eingängen und n ODER-Schaltungen mit je 2^{n-1} Eingängen für die Realisierung notwendig sind (Realisierung in disjunktiver Normalform).

Für die tatsächliche Realisierung wird die Logik natürlich optimiert und es wird nach regelmäßigen Strukturen gesucht. Dies soll am Beispiel eines 3-Bit-binär Zählers (Modulo-8-Zähler) hier demonstriert werden.

Vorwärtszählender 3-Bit Binärzähler

Das Bild 3-24 zeigt das Schaltwerk eines 3-Bit-Zählers mit dem Rückkoppelschaltnetz δ und dem 3-Bit Register aus D-Flipflops.



Vorwärtszählender 3-Bit Binärzähler

Hier soll nun das Rückkoppelschaltnetz hergeleitet. Dazu stellt man zweckmäßigerweise die zugehörige Flusstabelle auf:

Zustand	Q_2 z_2^n	Q_1 z_1^n	Q_0 z_0^n	Folge-zustand	D_2 z_2^{n+1}	D_1 z_1^{n+1}	D_0 z_0^{n+1}
Z_0^n	0	0	0	Z_0^{n+1}	0	0	1
Z_1^n	0	0	1	Z_1^{n+1}	0	1	0
Z_2^n	0	1	0	Z_2^{n+1}	0	1	1
Z_3^n	0	1	1	Z_3^{n+1}	1	0	0
Z_4^n	1	0	0	Z_4^{n+1}	1	0	1
Z_5^n	1	0	1	Z_5^{n+1}	1	1	0
Z_6^n	1	1	0	Z_6^{n+1}	1	1	1
Z_7^n	1	1	1	Z_7^{n+1}	0	0	0

Vorwärtszählender 3-Bit Binärzähler

Der Zustand Z^n aus der Zustandstabelle (z_2^n , z_1^n , z_0^n) ist dabei das was im als Ausgang (Q_2, Q_1, Q_0) sichtbar ist und der Folgezustand Z^{n+1} = (z_2^{n+1}, z_1^{n+1}, z_0^{n+1}) liegt am Ausgang des Schaltnetzes an den D-Eingängen (D_2, D_1, D_0) der Flipflops an.

Dieser Folgezustand wird mit dem nächsten Takt am Clk-Eingang in die Flipflops übernommen und ist dann der aktuelle Zustand Z^n.

Die notwendigen logischen Gleichungen können also direkt aus der Tabelle abgelesen werden:

$$D_0 = \bar{Q}_2\bar{Q}_1\bar{Q}_0 \vee \bar{Q}_2Q_1\bar{Q}_0 \vee Q_2\bar{Q}_1\bar{Q}_0 \vee Q_2Q_1\bar{Q}_0$$

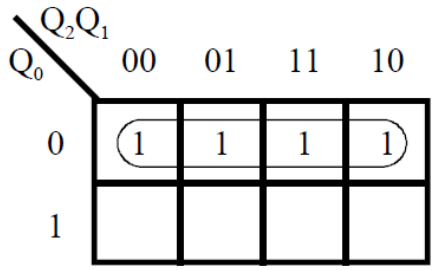
$$D_1 = \bar{Q}_2\bar{Q}_1Q_0 \vee \bar{Q}_2Q_1\bar{Q}_0 \vee Q_2\bar{Q}_1Q_0 \vee Q_2Q_1\bar{Q}_0$$

$$D_2 = \bar{Q}_2Q_1Q_0 \vee Q_2\bar{Q}_1\bar{Q}_0 \vee Q_2\bar{Q}_1Q_0 \vee Q_2Q_1\bar{Q}_0$$

Karnaugh-Veith Minimierung

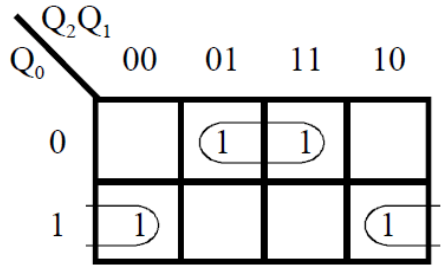
Die Minimierung ergibt:

D_0:



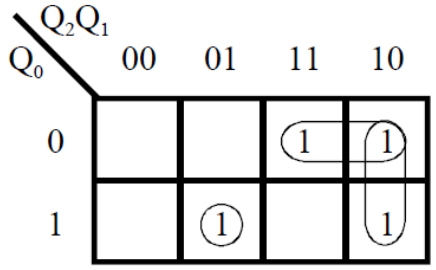
$$D_0 = \bar{Q}_0$$

D_1:



$$D_1 = \bar{Q}_0 Q_1 \vee Q_0 \bar{Q}_1 = Q_0 \oplus Q_1$$

D_2:



$$D_2 = Q_2 \bar{Q}_0 \vee Q_2 \bar{Q}_1 \vee \bar{Q}_2 Q_1 Q_0$$

Alternative Implementierung

Diese logischen Gleichungen könnte man so implementieren allerdings lässt sich D_2 noch systematisieren:

$$
\begin{aligned}
D_2 &= Q_2\bar{Q}_0 \vee Q_2\bar{Q}_1 \vee \bar{Q}_2 Q_1 Q_0 \\
&= Q_2 \wedge \left(\bar{Q}_0 \vee \bar{Q}_1\right) \vee \left(\bar{Q}_2 \wedge Q_1 \wedge Q_0\right) \text{ (Satz von DeMorgan)} \\
&= Q_2 \wedge \left(\overline{Q_0 \wedge Q_1}\right) \vee \bar{Q}_2 \wedge \left(Q_1 \wedge Q_0\right) \\
&= Q_2 \oplus \left(Q_0 \wedge Q_1\right)
\end{aligned}
$$

Somit kann man D_2 und D_1 als XOR-Verknüpfung mit sich selbst und dem jeweils niederwertigeren Bit auffassen und D_0 ergibt sich als XOR-Verknüpfung mit 1. Es ergibt sich:

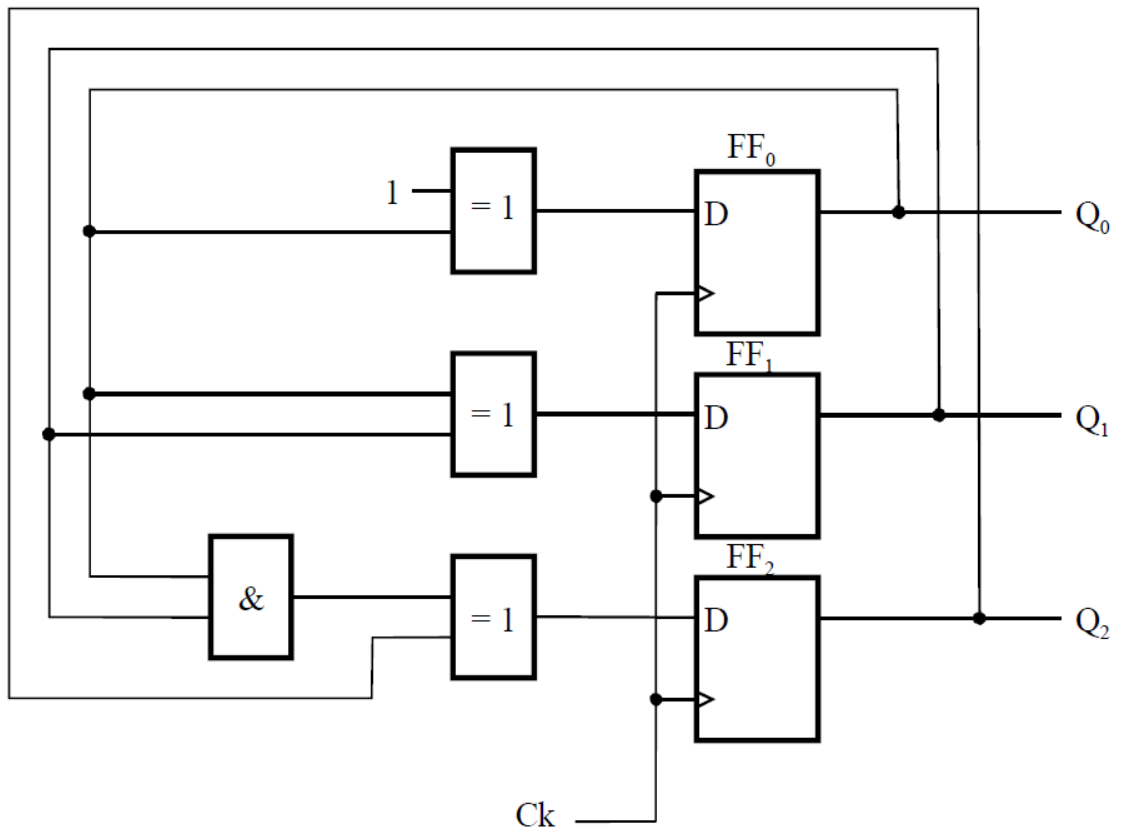
Und als mögliche Erweiterung:

$$
\begin{aligned}
D_0 &= Q_0 \oplus 1 \\
D_1 &= Q_1 \oplus Q_0 \\
D_2 &= Q_2 \oplus \left(Q_0 Q_1\right) \\
D_3 &= Q_3 \oplus \left(Q_0 Q_1 Q_2\right) \\
D_4 &= Q_4 \oplus \left(Q_0 Q_1 Q_2 Q_3\right)
\end{aligned}
$$

usw.

Vorwärtszählender mit D-FF

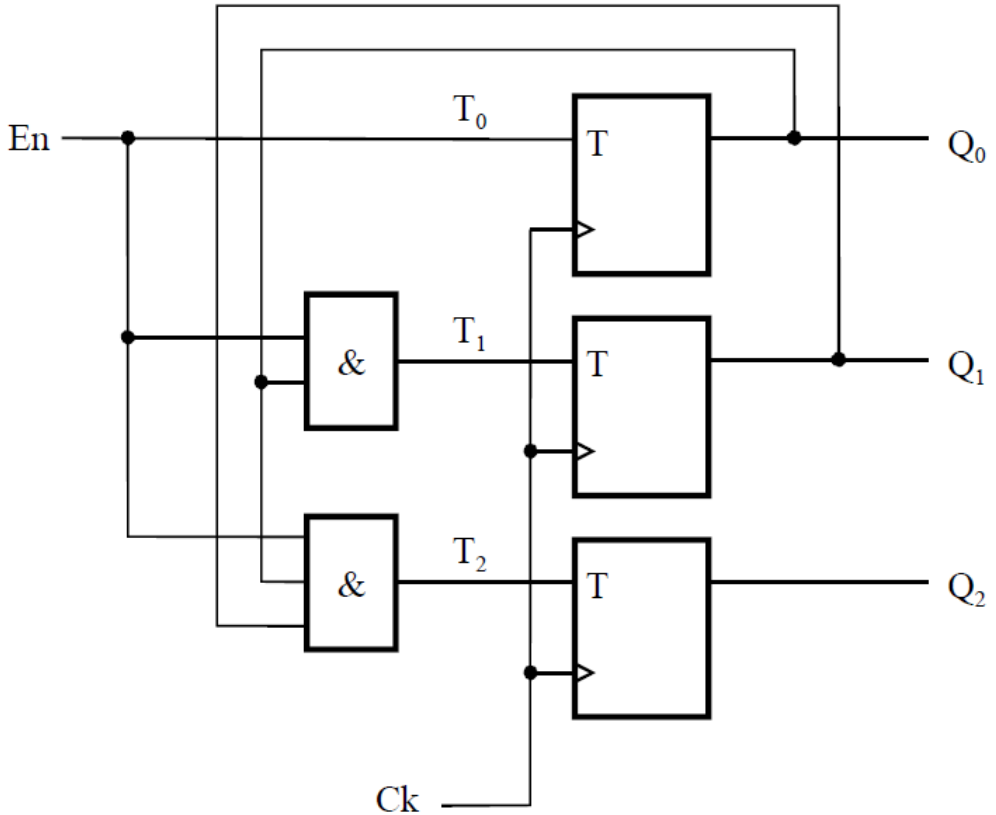
Mit den aufgestellten Gleichungen kann jetzt das Schaltnetz für den vorwärtszählenden 3-Bit Binärzähler vervollständigt werden:



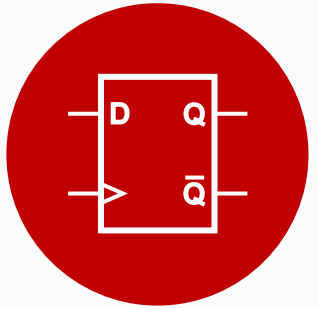
Vorwärtszählender mit T-FF

Ein D-Flipflop mit XOR-Gatter, bei dem ein Eingang des XOR-Gatter am Ausgang des Flipflops angeschlossen ist, ergibt ein Toggle-Flipflop.

Hier ist der Zähler ergänzt mit einen Eingabeeingang En (Enable = Zählfreigabe)



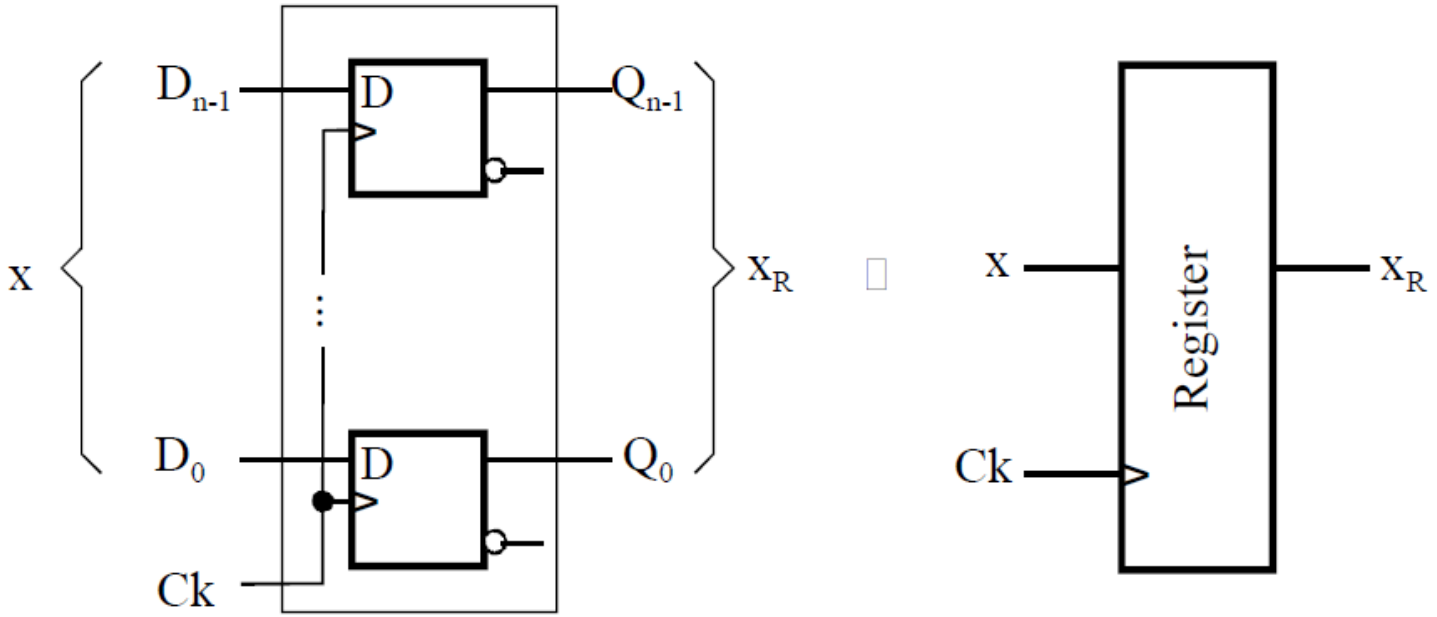
Synchrone Schaltungen



Register

Die Systemflipflops lassen sich auch als Speicher für mehrere Variablen verwenden. Ein Eingangsvektor X mit n Variablen (x_{n-1} … x_0) lässt sich dann in einem sogenannten Register aus z.B. D-Flipflops speichern.

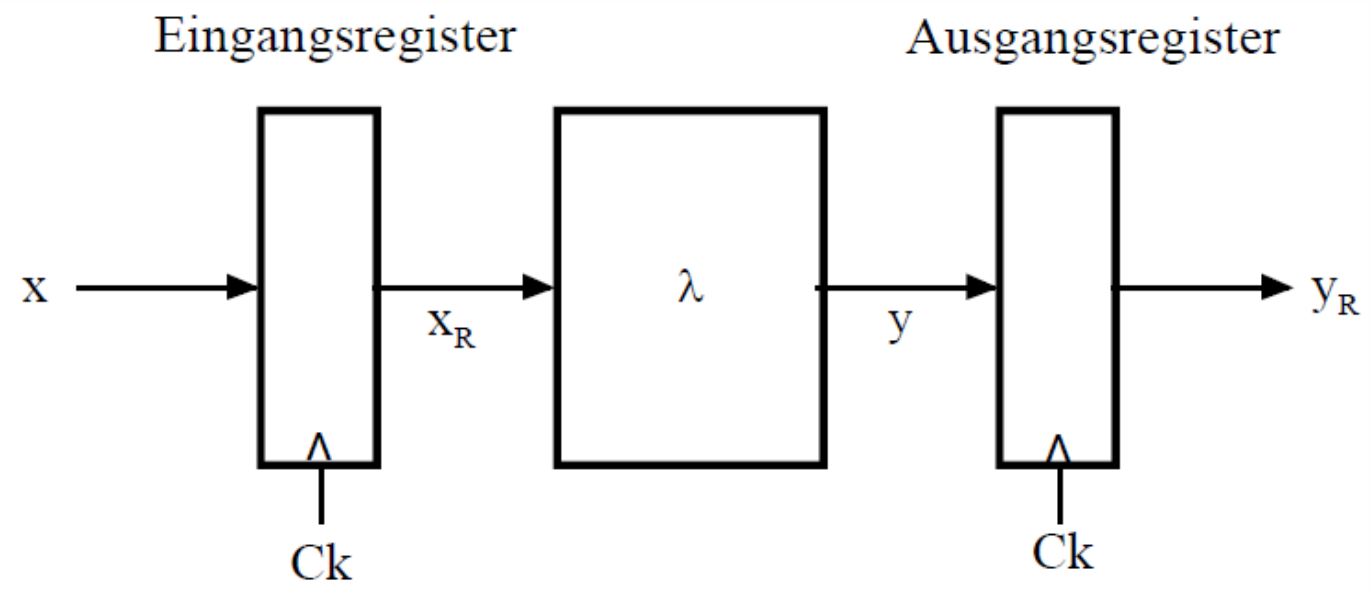
Der Registerausgang X_R ändert sich nur mit jeder positiven Taktflanke auf den dann gültigen Eingangswert X, zwischen den Taktflanken ist der Ausgang stabil.



Register

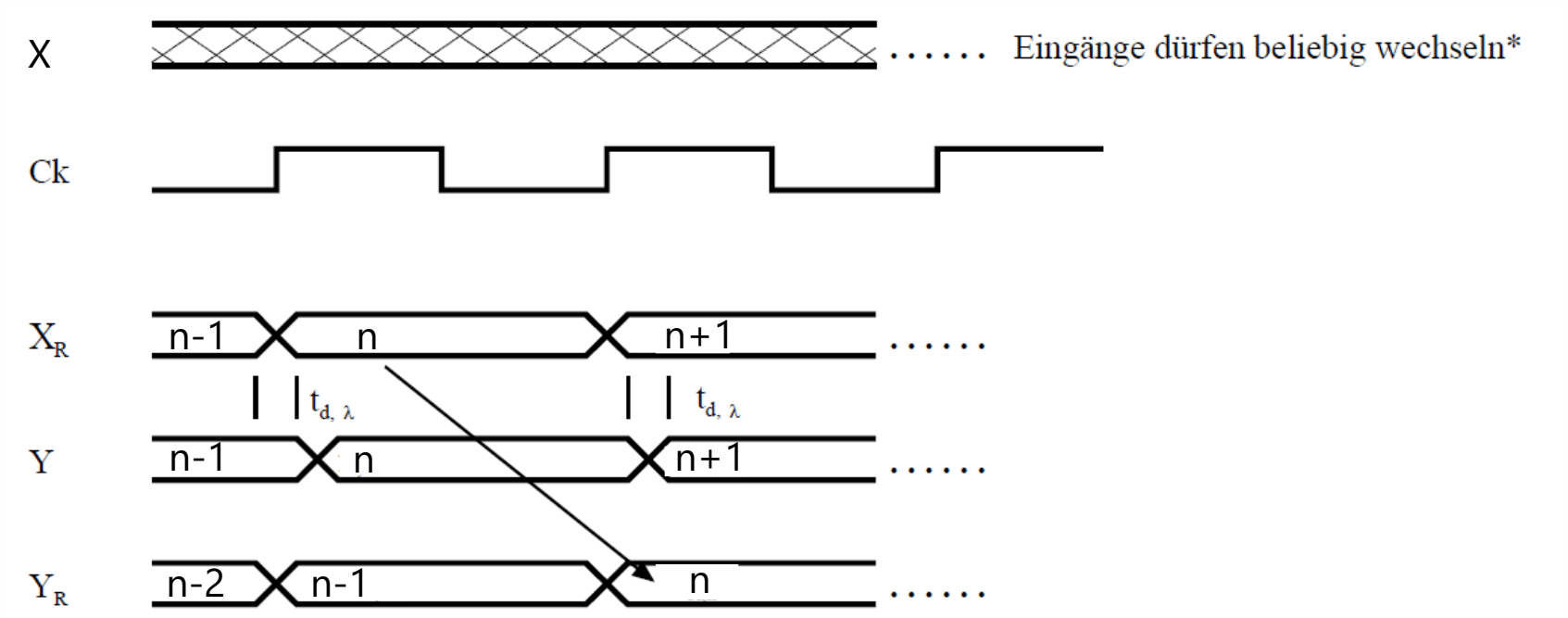
Diese "Registrierung" von Signalen kann zu der in Kap. 2 angesprochenen Vermeidung von Auswirkungen durch Hazards in einem System verwendet werden.

Wie im Bild dargestellt lassen sich sowohl Eingangs- wie auch Ausgangssignale auf den Takt synchronisieren und somit Funktions-/Strukturhazards in ihren Auswirkungen vermeiden.



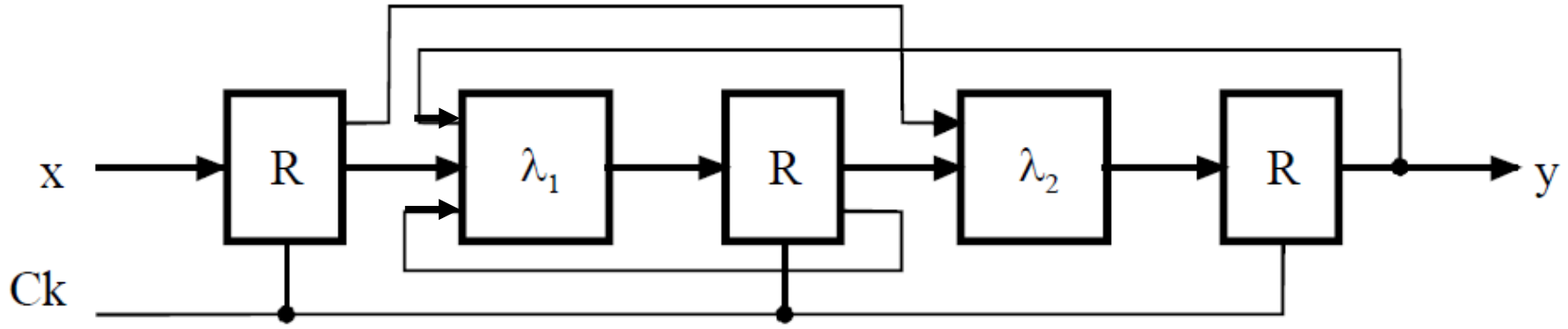
Register

Der Ausgang X_R des Eingangsregisters ändert sich nur mit jeder positiven Taktflanke auf den dann gültigen Eingangswert X, zwischen den Taktflanken ist der Ausgang stabil. Das gleiche gilt auch für das Ausgangsregister Y, das sich um eine Taktperiode verzögert gegenüber dem Eingangssignal ändert.



Register-Transfer-Logik

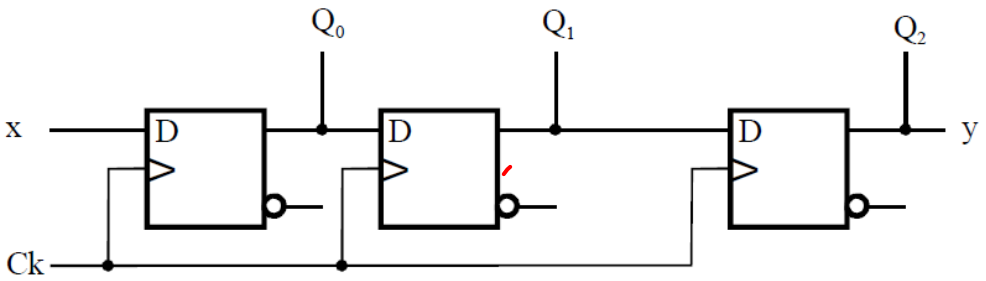
Taktsynchrone Systeme dürfen rückgekoppelt werden. Ein vollsynchronisiertes System aus Schaltnetzen / Schaltwerken wird dann als sog. **Register-Transfer-Logik** bezeichnet, wobei sich das Ausgangssignal mit jeder Registerstufe um eine Taktperiode verzögert:

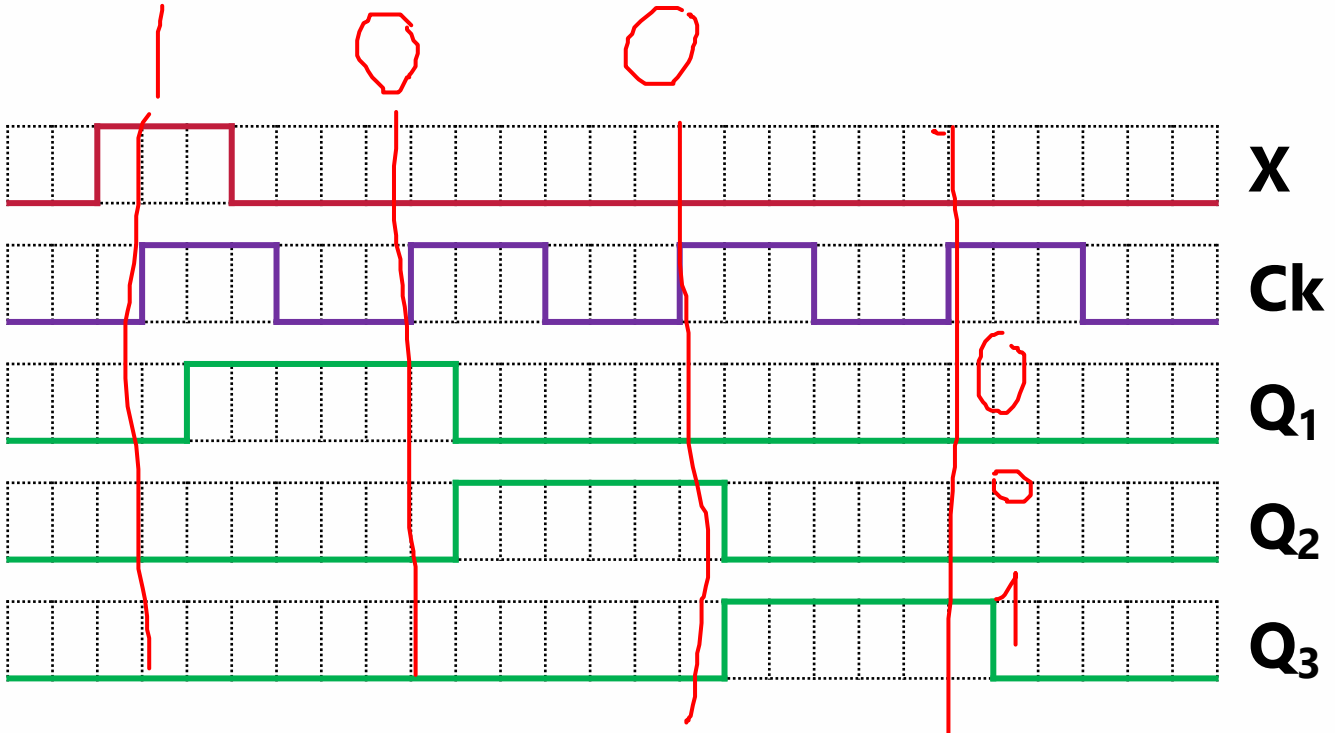


Bedingung hierfür ist, dass die Verzögerungszeiten durch die Schaltnetze kleiner sind als die Taktperiode.

Schieberegister

Eine Kette von n Flipflops kann daher als digitale Verzögerungseinheit um n-1 Takte verwendet werden





Annahme: Flipflop hat 1 Zeiteinheit Verzügerung

Konfigurierbare Register

Mit entsprechenden Schaltnetzen an Ein-/Ausgängen der Flip-Flops können aus einem einfachen Schieberegister auch parallel ladbare und parallel auslesbare Schieberegister oder andere Abwandlungen realisiert werden.

Das Bild zeigt ein Schieberegister mit parallelen (Px) und seriellen (Sx) Ein-/Ausgängen, das z. B. zur Umwandlung eines seriellen (bit für bit) Datenstroms in parallele Datenworte von n bits und umgekehrt genutzt werden kann.