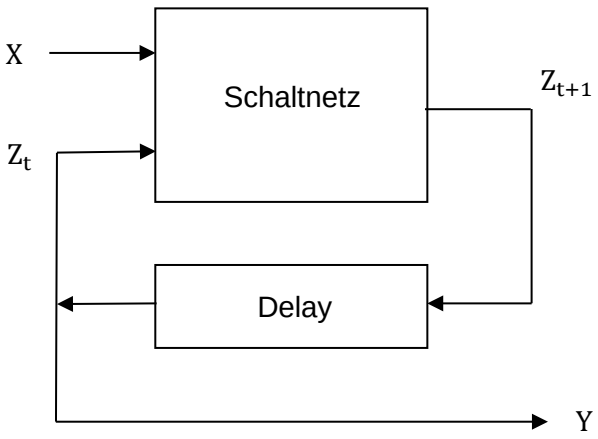
Schaltwerke

Inhalt

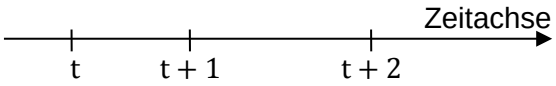
1. Asynchrone Schaltwerke

Situation *):



X	eine bis mehrere Eingangsvariablen
Z_t	zum Zeitpunkt t ins Schaltnetz geführte Variable/n
Z_{t+1}	vom Schaltnetz bereitgestellte Variable/n, die zum Zeitpunkt t+1 (nach Delay) zu neuen Eingangsvariable/n Z_t werden.
Y	Ausgangsvariablen des Schaltwerks

Delay: Verzögerung in der Rückkopplung, Entkoppelung



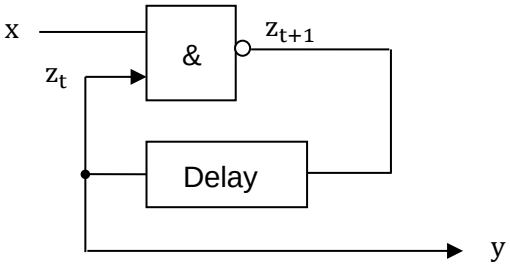
Ein *Schaltwerk* ist eine digitale Schaltung, welche Zustandscharakter aufweist. Der Zustand zum Zeitpunkt $t + i$ ist abhängig vom Vorzustand (von den rückgekoppelten Werten und den angelegten Werten der Eingangsvariablen), etc.

Asynchron heisst, dass kein äusseres Synchronisier-Signal die Schaltung beeinflusst. Bei einer Änderung von Eingangsvariablen zur Zeit t ist genügend lange zu warten, bis ein neuer Zustand zur Zeit $t + i$ eingenommen wird. Es wird vorausgesetzt, dass die Schaltung sich in kurzer Zeit stabilisiert.

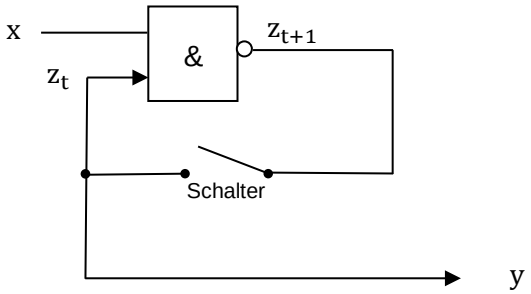
Zustandsmaschinen (wie z.B. obige Konstellation) spielen eine wichtige Rolle - auch in der Informatik. Mehr darüber kann z.B. unter den Begriffen *Finite State Machines* (FSM) und *Endliche Automaten* gefunden werden.

*) Situation für nachfolgend eingeführtes RS-Flipflop.

Frage:
Ist die untenstehende Schaltung asynchron? Handelt es sich um eine stabile Schaltung?



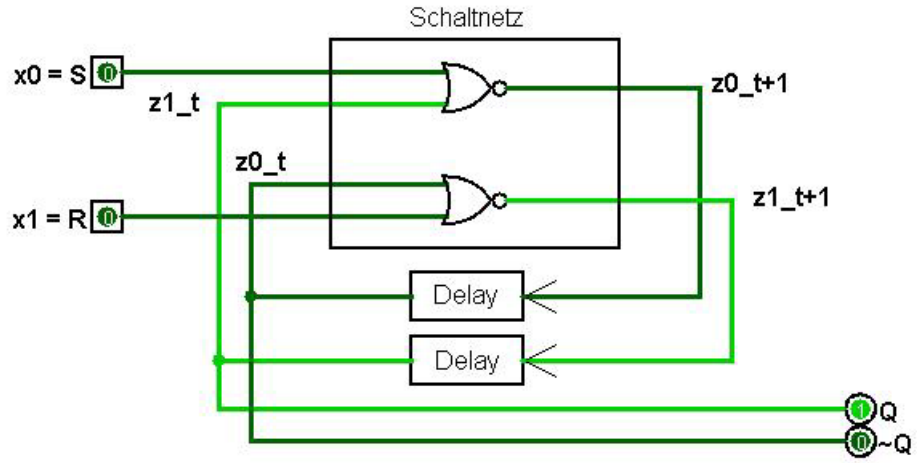
Vorgehen bei der Stabilitätsbetrachtung:
Schaltung unten bei geöffnetem Schalter (Rückführung unterbrochen) betrachten. Danach Schalter schliessen (Rückführung geschlossen). Testen, ob Situation stabil ist ($z_t = z_{t+1}$).



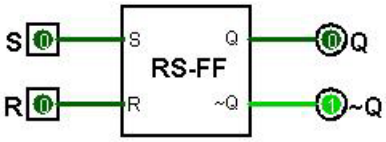
2. Flipflops (FF)

RS-Flipflop

Beim RS-FF handelt es sich um einen einfachen 1-Bit-Speicher, der als asynchrones Schaltwerk verstanden werden kann, bestehend aus Schaltnetz und Rückführungen (vgl. auch *flipflops.circ*, Schaltung RS-FF).



Das RS-FF als Black Box gezeichnet:



Für das RS-FF kann folgende sog. Zustandsfolgetabelle formuliert werden. In dieser wird auch gerne der inverse Ausgang von Q weggelassen.

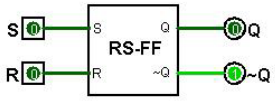
S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	verboten

Mit S als Set-Eingang und R als Reset-Eingang wird den Eingängen Bedeutung gegeben. Die Zeilen obiger Tabelle von unten nach oben erklärt:

- Es ist verboten, dass das RS-FF gleichzeitig gesetzt ($S = 1$) und zurückgesetzt ($R = 1$) wird. Das gleichzeitige Zurücksetzen ergäbe auch sprachlich-logisch keinen Sinn.
- Das Setzen des FF mit $S = 1$ hat zur Folge, dass der Ausgang $Q_{t+1} = 1$ ist oder wird.
- Das Rücksetzen des FF mit $R = 1$ hat zur Folge, dass der Ausgang $Q_{t+1} = 0$ ist oder wird.
- Wird weder gesetzt ($S = 0$) noch zurückgesetzt ($R = 0$), so behält das FF seinen alten Zustand bei (bzw. der neue Zustand entspricht dem alten, d.h. $Q_{t+1} = Q_t$).

Beim RS-FF muss es sich um einen Speicher handeln, denn es ist möglich, dass für S=R=0 am Ausgang Q=0, dann aber auch Q=1 ist.

Symbol



Zustandsfolgetabelle

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	verboten

Anregungstabelle

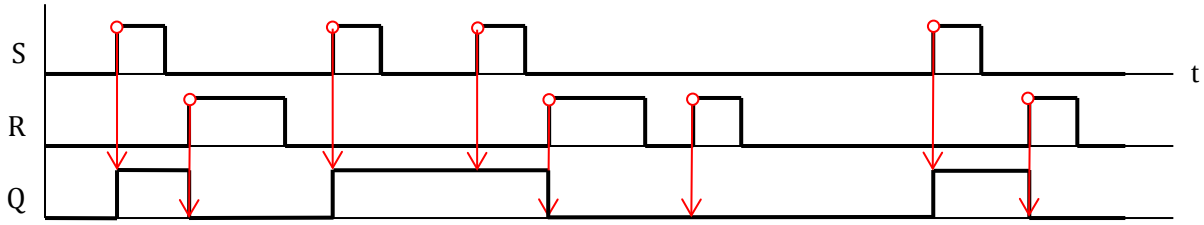
S	R	Q_t	Q_{t+1}
0	x	0	0
1	0	0	1
0	1	1	0
x	0	1	1

Die Zustandsfolgetabelle wird wie folgt gelesen (oberste Zeile als Beispiel):
Gilt $S=0$ und $R=0$, dann ist der neue Zustand Q_{t+1} gleich dem alten Zustand Q_t.

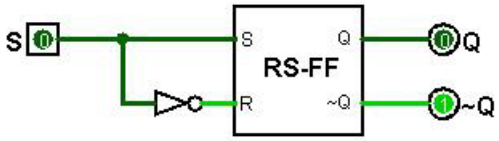
Die Anregungstabelle wird wie folgt gelesen (oberste Zeile als Beispiel):
Soll, ausgehend vom alten Zustand $Q_t = 0$, der neue Zustand $Q_{t+1} = 0$ sein, dann kann dies erreicht werden, wenn $S=0$ ist oder wird. R darf den Wert 0 oder 1 aufweisen.

Zustandsfolgefunktion oder -gleichung: $Q_{t+1} = \overline{R + \overline{S + Q_t}}$

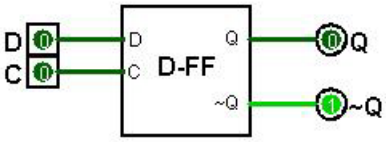
Signal-Zeit-Diagramm (willkürlich gewählte Signale S und R. Q zu Beginn auf Tief initialisiert):



D-Flipflop
Die verbotene Situation des RS-FF kann verhindert werden, indem der R- und der S-Eingang wie folgt verbunden werden:



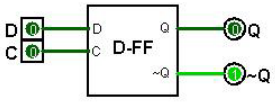
Dies führt zum D-FF, auch als Delay-FF bezeichnet. Hier wurde zusätzlich der Eingang C (als Enable-Eingang) ergänzt.



Damit ergibt sich das

D-FF mit Zustands-Steuerung

Symbol



Zustandsfolgetabelle

C	D	Q_{t+1}
0	x	Q_t
1	0	0
1	1	1

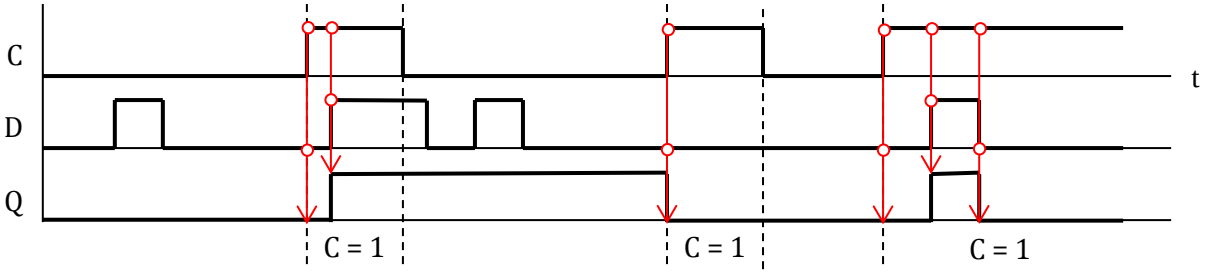
Anregungstabelle

C	D	Q_t	Q_{t+1}
0	x	0	0
0	x	1	1
1	0	0	0
1	1	0	1
1	0	1	0
1	1	1	1

Zustandsfolgefunktion

$$Q_{t+1} = DC + \bar{C}Q_t$$

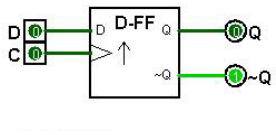
Signal-Zeit-Diagramm (willkürlich gewählte Signale C und D. Q zu Beginn auf Tief initialisiert):



Zur Funktionsweise vgl. auch *flipflops.circ*, Schaltung D-FF (zustandsgesteuert)).

D-FF mit positiver 1-Flanken-Steuerung

Positive Flanke: Eingangsdetektion von D
mit sofortigem Ausgangseinfluss auf Q_{t+1}

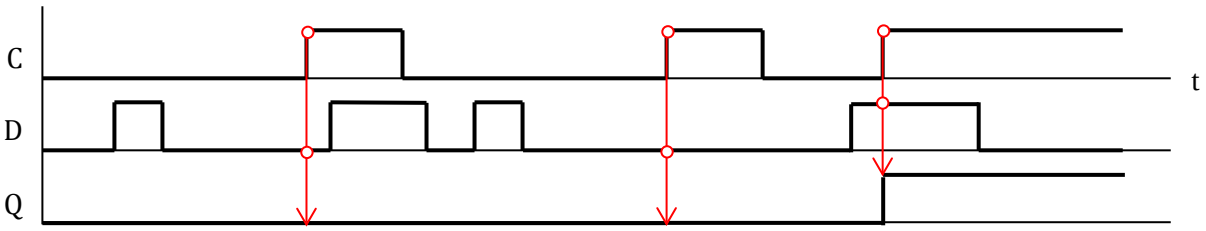


$$Q_{t+1} = DC + \bar{C}Q_t$$

C	D	Q_{t+1}
0,1,↓	x	Q_t
↑	0	0
↑	1	1

C	D	Q_t	Q_{t+1}
0,1,↓	x	0	0
0,1,↓	x	1	1
↑	0	0	0
↑	1	0	1
↑	0	1	0
↑	1	1	1

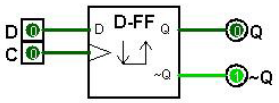
Signal-Zeit-Diagramm (willkürlich gewählte Signale C und D. Q zu Beginn auf Tief initialisiert):



D-FF mit 2-Flanken-Steuerung

1. Negative Vorderflanke: Eingangsdetektion von D
2. Positive Hinterflanke: Ausgangseinfluss auf Q_{t+1}

Symbol



$$Q_{t+1} = DC + \bar{C}Q_t$$

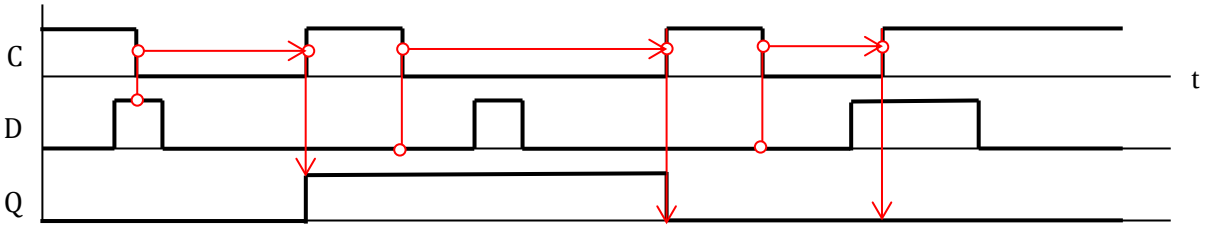
Zustandsfolgetabelle

C	D	Q_{t+1}
0, 1	x	Q_t
↓↑	0	0
↓↑	1	1

Anregungstabelle

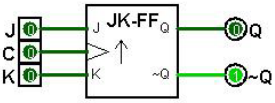
C	D	Q_t	Q_{t+1}
0,1	x	0	0
0,1	x	1	1
↓↑	0	0	0
↓↑	1	0	1
↓↑	0	1	0
↓↑	1	1	1

Signal-Zeit-Diagramm (willkürlich gewählte Signale C und D. Q zu Beginn auf Tief initialisiert):



JK-FF mit positiver 1-Flanken-Steuerung

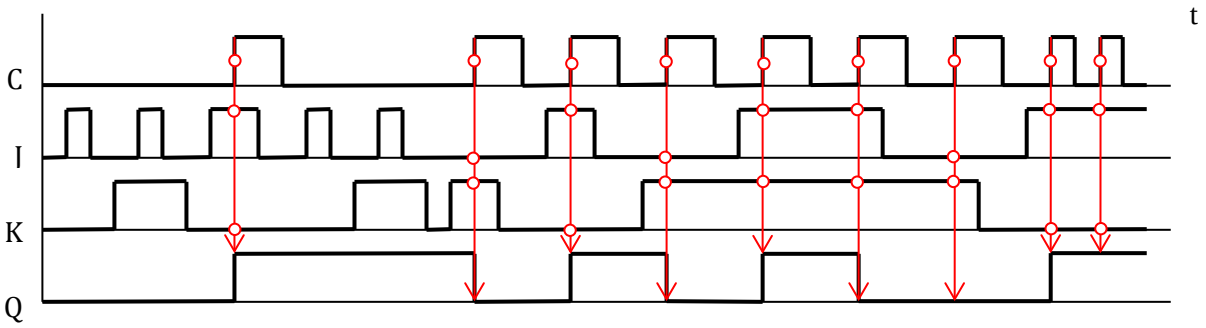
Positive Flanke: Eingangsdetektion von J und K
mit sofortigem Ausgangseinfluss auf Q_{t+1}



$$Q_{t+1} = C\,J\overline{Q_t} + \bar{K}Q_t + \bar{C}Q_t$$

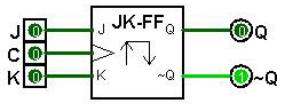
C	J	K	Q_{t+1}
0	x	x	Q_t
1	x	x	Q_t
↓	x	x	Q_t
↑	0	0	Q_t
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_t}$

C	J	K	Q_t	Q_{t+1}
0,1,↓	x	x	0	0
0,1,↓	x	x	1	1
↑	0	x	0	0
↑	1	x	0	1
↑	x	1	1	0
↑	x	0	1	1



JK-FF mit 2-Flanken-Steuerung

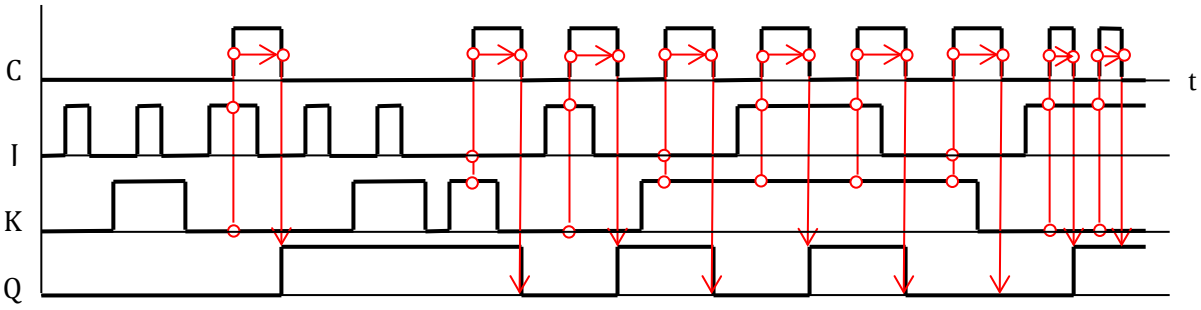
1. Positive Vorderflanke: Eingangsdetektion von J und K
2. Negative Hinterflanke: Ausgangseinfluss auf Q_{t+1}



$$Q_{t+1} = C\,J\overline{Q_t} + \overline{K}Q_t + \overline{C}Q_t$$

C	J	K	Q_{t+1}
0, 1	x	x	Q_t
↓	0	0	Q_t
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q_t}$

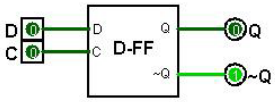
C	J	K	Q_t	Q_{t+1}
0,1	x	x	0	0
0,1	x	x	1	1
↓	0	x	0	0
↓	1	x	0	1
↓	x	1	1	0
↓	x	0	1	1

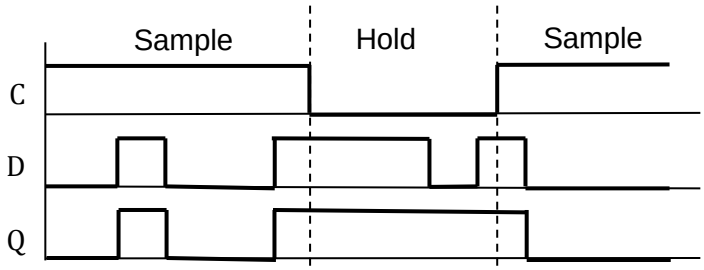


Weitere Flipflops s. Datei *flipflops.circ*, insbesondere mit Hilfseingängen Enable, Set und Reset. Set und Reset geniessen die höchsten Prioritäten und wirken asynchron, d.h. unabhängig von C bezüglich Niveau und Flanken und unabhängig von Enable. Enable muss aktiviert sein, soll das FF bei seiner spezifizierten Triggerung (Niveau, Flanke/n) entsprechende Werte speichern.

Beispiele von asynchronen Schaltungen

Sample-Hold



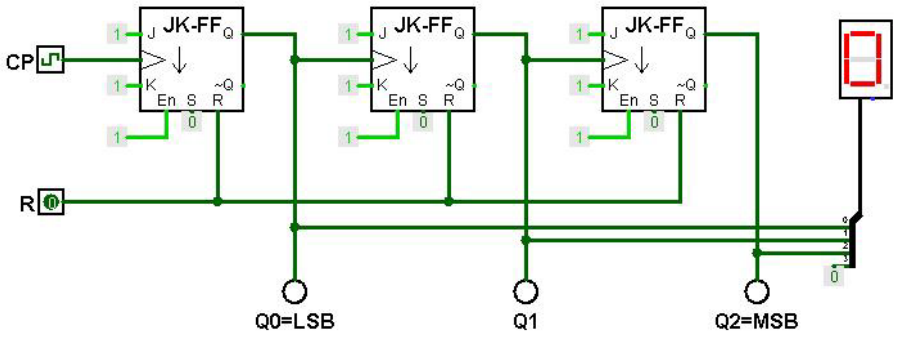


8 D-FF in parallel können z.B. an einem 8-Bit-Bus mithören. Der Bustakt ist mit allen C verbunden, so dass je eine Bus-Halbphase gesamplet wird und eine Bus-Halbphase die Daten gespeichert bleiben, noch wenn während dieser Zeit weitere Änderungen auf dem Bus geschehen. Z.B. eingesetzt, um auf multiplexten Bussen (Daten und Adressen auf gleichen Leitungen) gezielt die Daten herauszufiltern.

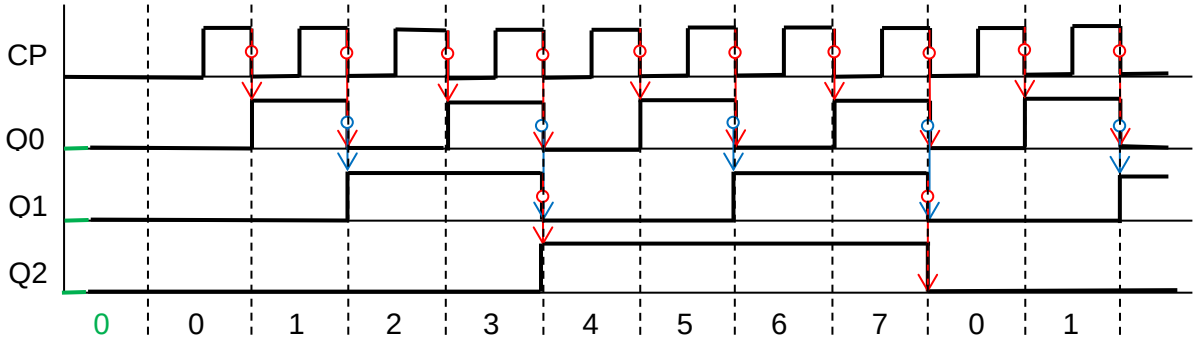
D-FF sind einfach als Bit-Speicher einsetzbar.
D-FF werden auch als Delay-FF bezeichnet: Ein Signalwert wird gespeichert und kann zu einem späterem Zeitpunkt weiter verwendet werden.

Modulo-8-Vorwärtszähler

Realisiert mit negativ 1-Flanken-getriggerten JK-FF mit asynchronem hoch-aktivem R. $R = 0$ bewirkt in jedem Fall (prioritär) $Qi = 0$.

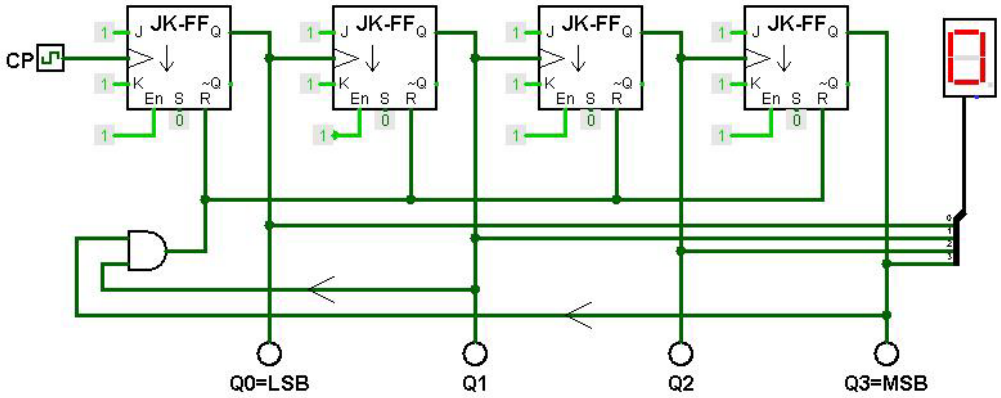


Bei gegebenem CP und Initialzuständen Q0, Q1 und Q2 ergeben sich folgende Ausgangssignale Qi:



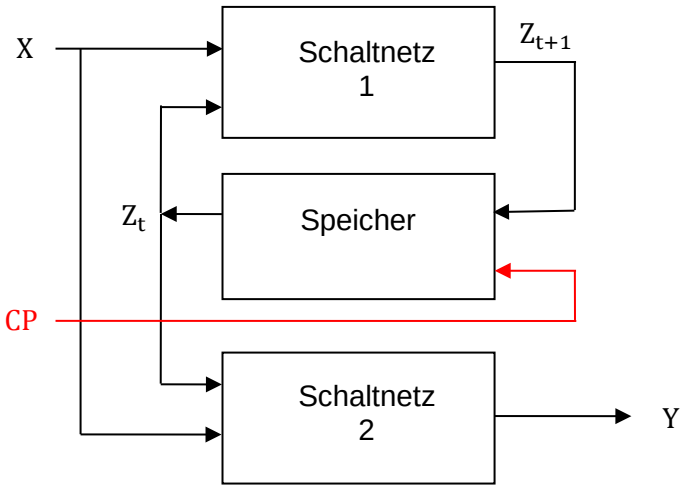
Obige Konfiguration kann auch als Frequenzteiler eingesetzt werden. Q1 weist die halbe Frequenz von Q0 auf etc.

Modulo-10-Vorwärtszähler unter Ausnützung der asynchronen Reset



Der Zähler erreicht kurzzeitig den Zählerstand 10dez., was anschliessend sofort das Zurücksetzen aller Ausgänge Qi bewirkt. Das Rücksetzen geschieht mit dem Und-Gatter für Q1=Q3=1.

3. Synchrone Schaltwerke



X	eine bis mehrere Eingangsvariablen
CP	Synchronisiersignal (Clock Pulse)
Z_t	ins Schaltnetz zurückgeführte Variable/n
Z_{t+1}	Aus dem Schaltnetz herausgeführte, Variable/n, zum Zeitpunkt t+1 neue Eingangsvariable/n in den Speicher
Y	Decodierte Ausgänge

Speicher:
synchronisierte Verzögerung in der Rückkopplung mit Speicherelementen;
zeitlich kontrollierte Entkoppelung

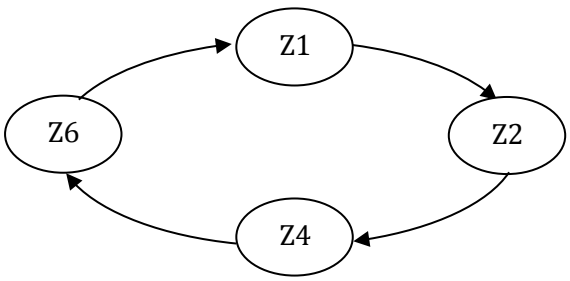
Synchrone Schaltwerke werden für die Realisierung von Zustandsmaschinen eingesetzt.

Der Clock Pulse (auch: Clock, Takt) synchronisiert die Schaltung zu gegebenen Zeitpunkten, die so zu wählen sind, dass die Ausgangssignale (Z_{t+1}) sicher stabil vorliegen. D.h. erst wenn die Signale stabil vorliegen, werden die Signalwerte mit Hilfe des CP in den Speicher übernommen.

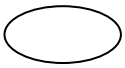
Einfachere Schaltwerke können auf das Schaltnetz 1 und/oder das Schaltnetzt 2 verzichten (bzw. man kann sich diese auf einfache Drahtverbindungen reduziert vorstellen). Das Schaltnetz 1 realisiert die Rückkopplung von den Speicherausgängen zu deren Eingängen. Das Schaltnetz 2 ist eine Dekodierschaltung (Codewandler).
Schaltwerke, die von aussen nicht beeinflusst zu werden brauchen, haben keine Eingangsvariablen X nötig.

Beispiel:
Es soll eine Verkehrsampel mit den Leuchten rot, gelb und grün angesteuert werden. Die Ampel kennt folgende vier Zustände: grün – gelb – rot – rot/gelb. Diese Zustände werden wiederholt in der notierten Reihenfolge durchlaufen. Mit jedem Takt findet ein Zustandswechsel statt. Einen Eingangsvektor X gibt es nicht, da von aussen keine Beeinflussung vorgesehen ist.

Zugehöriges Zustandsdiagramm



Legende



Zustand

Zustandsname
Z1 Die Wahl der *i* von Z*i* erklärt sich weiter unten.

Zustandsübergang, erfolgt auf Clock Pulse.

Vereinbarungen:
Z1: Zustand grün
Z2: Zustand gelb
Z4: Zustand rot
Z6: Zustand rot/gelb

Für jede Leuchte sehen wir eine1-Bit Leitung vor. Demnach ist ein Zustand durch die Werte z von 3 Signalen beschreibbar: $Z = (z2(=MSB), z1, z0(= LSB))$. Die Zustandsfolge kann aus dem Zustandsdiagramm in eine Tabelle übertragen werden. Dies führt zur Zustandsfolgetabelle, welche anschliessend codiert werden kann (Zustandsfolgetabelle mit den einzelnen z):

Zustandsfolgetabelle

Z_t	Z_{t+1}
Z1	Z2
Z2	Z4
Z4	Z6
Z6	Z1

Codierte Zustandsfolgetabelle

$z2_t$	$z1_t$	$z0_t$	$z2_{t+1}$	$z1_{t+1}$	$z0_{t+1}$
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	1	1	0
1	1	0	0	0	1

Hinweis:
Die Nummerierung der Zustände Z erfolgte so, dass diese gleich den Code $(z2, z1, z0)$ als Dezimalzahl widergeben. Jede andere Bezeichnung wäre auch möglich.

Die Vervollständigung der Tabelle oben rechts führt zu

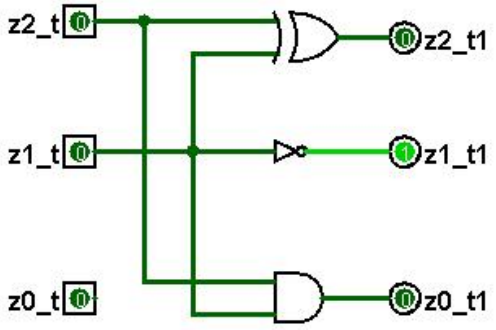
$z2_t$	$z1_t$	$z0_t$	$z2_{t+1}$	$z1_{t+1}$	$z0_{t+1}$
0	0	0	x	x	x
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	x	x	x
1	0	0	1	1	0
1	0	1	x	x	x
1	1	0	0	0	1
1	1	1	x	x	x

In der codierten Zustandsfolgetabelle sind bei 3 Eingangsvariablen als weitere vier Permutationen möglich. Diese interessieren vorerst nicht.

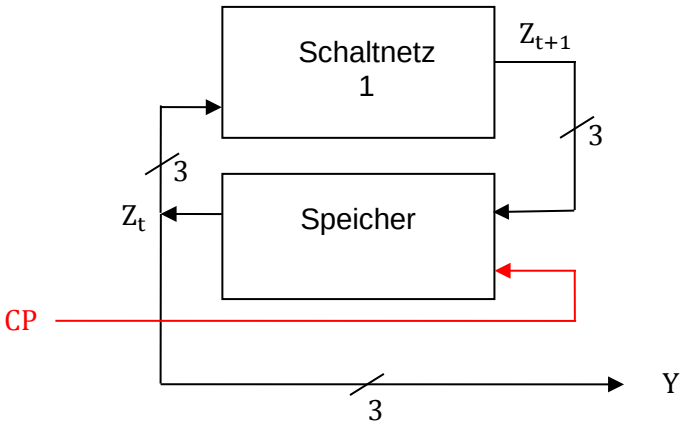
Suchen des minimierten Schaltnetzes (SN1) unter Berücksichtigung der Don't Cares liefert:

$z2_{t+1} = \overline{z2_t}\ z1_t + z2_t\ \overline{z1_t} \quad [= z2 \oplus z1]$
$z1_{t+1} = \overline{z1_t}$
$z0_{t+1} = z2_t\ z1_t$

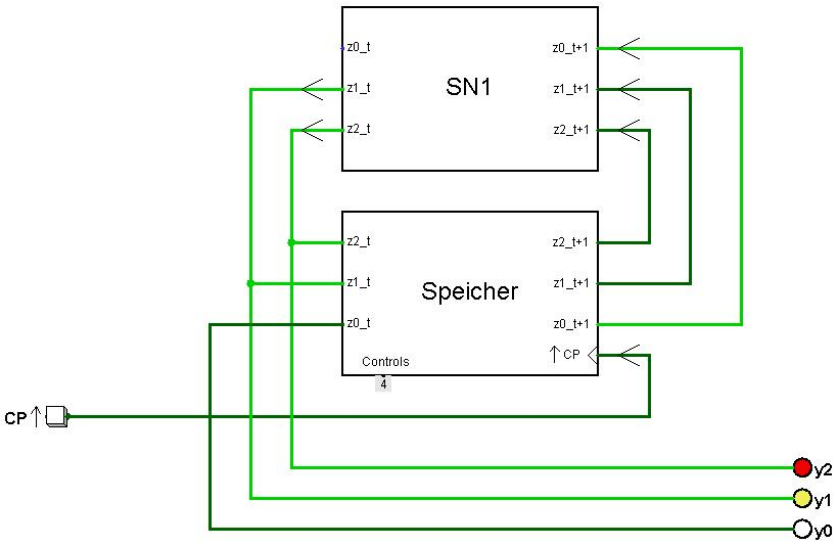
Oder als Schaltung:



Mit der gewählten Codierung oben verkümmert das Schaltnetz 2 zu reinen Drahtverbindungen. Damit reduziert sich die Zustandsmaschine auf folgendes Aussehen:



Mit den 3 (oben gebündelten) Leitungen an die und von der Speicher-Blackbox sind 3 Flipflop nötig. Damit kann jetzt die vollständige Zustandsmaschine in Logisim realisiert werden

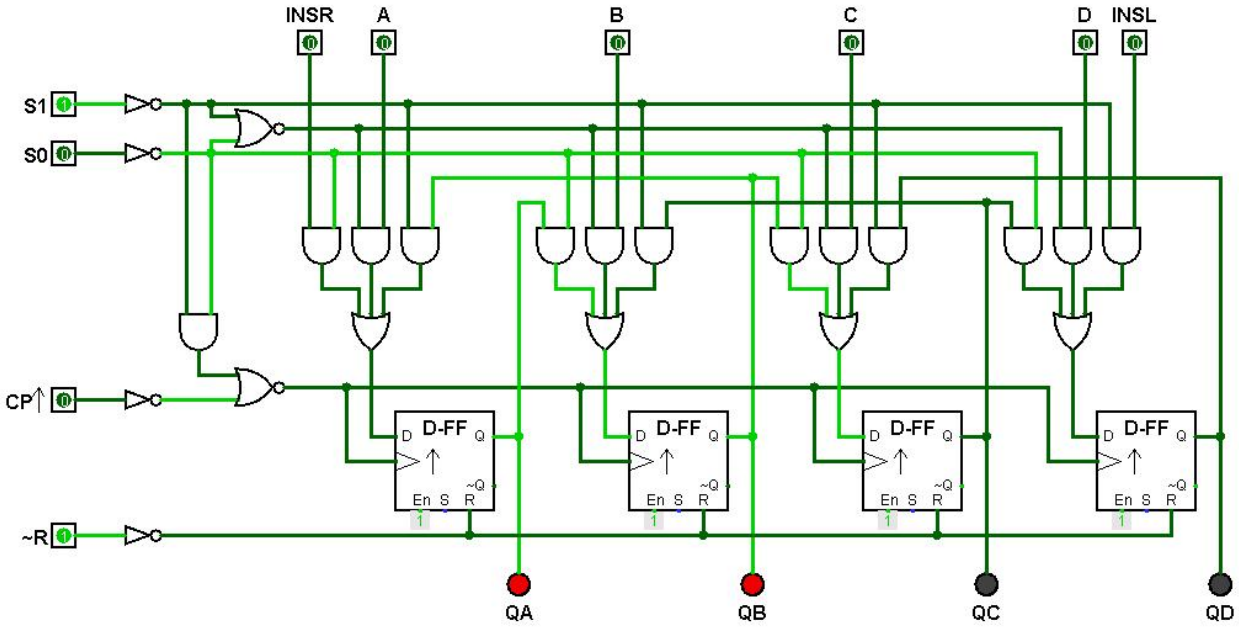


Aufgabe:
Jetzt kann die Zustandsfolgetabelle vervollständigt werden. Bestimmen Sie die konkreten Werte der vormaligen Don't Cares.
Notieren Sie jetzt das vollständige Zustandsdiagramm

Beispiele von synchronen Schaltungen

Schieberegister, Serie-Parallelregister

Mehrere 1-Bit-Speicher nebeneinander für denselben Zweck verwendet werden als Register bezeichnet. Unten ein 4-Bit-Register.



Die Wahrheitstabelle zeigt, dass Reset (tief aktiv) die höchste Priorität geniesst.

Mode	S1	S0	$\overline{R}$	CP	INSR	INSL	A	B	C	D	QA	QB	QC	QD
load parallel	1	1	1	↑	x	x	A	B	C	D	A	B	C	D
shift right	1	0	1	↑	1	x	x	x	x	x	1	QA	QB	QC
	1	0	1	↑	0	x	x	x	x	x	0	QA	QB	QC
shift left	0	1	1	↑	x	1	x	x	x	x	QB	QC	QD	1
	0	1	1	↑	x	0	x	x	x	x	QB	QC	QD	0
inhibit clock	0	0	1	x	x	x	x	x	x	x	QA	QB	QC	QD
reset	x	x	0	x	x	x	x	x	x	x	0	0	0	0

Aufgabe:
Laden Sie ein Nibble parallel ins Register und schieben Sie es anschliessen zweimal nach rechts. Wie lautet die Befehlsfolge?

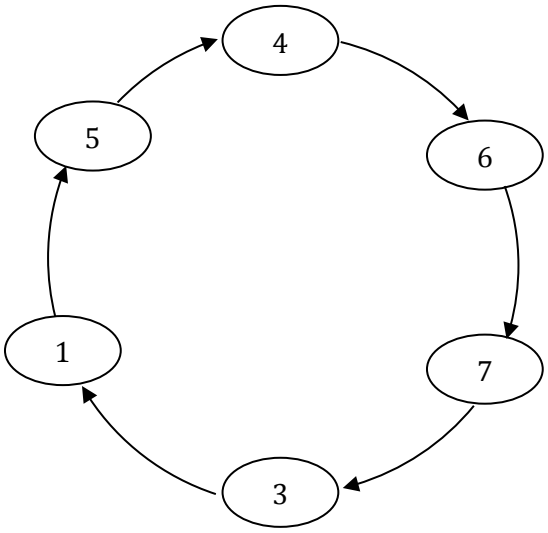
Zählerschaltungen

Beliebige Zählerschaltungen (Modulozähler, vorwärts, rückwärts, hoch aktiv, tief aktiv) lassen sich als synchrone Schaltungen unter Zuhilfenahme von FF realisieren. Mittels der vorgestellten Theorie lassen sich solche leicht synthetisieren.

Beispielhaft wird hier ein synchroner „Zähler“ entworfen. Aus der Aufgabenstellung und der vorgestellten Lösung geht hervor, dass sogar Werte (in Modulo-Manier, d.h. repetitiv) durchlaufen werden, die keine Reihenfolge mit auf- oder absteigenden Werten aufweisen.

Aufgabe (mit Lösung):
Es ist ein synchroner Modulo-6-"Zähler" zu entwerfen, wie dies die nachfolgende Tabelle zeigt. Einer der bevorstehenden Übergänge von einem zum nächsten Zählerstand werde mit dem Ausgang $\overline{W} = 0$ gekennzeichnet (W für Wrap over).

t			t	t+1		
Q2	Q1	Q0	$\overline{W}$	D2	D1	D0
0	1	1	1	0	0	1
0	0	1	1	1	0	1
1	0	1	1	1	0	0
1	0	0	1	1	1	0
1	1	0	1	1	1	1
1	1	1	0	0	1	1
0	0	0	1	x	x	x
0	1	0	1	x	x	x



$Q2_t = z2_t$ etc. $\quad D2_{t+1} = z2_{t+1}$ etc.

Aufgabe (mit Lösung):
Stellen Sie die minimierten Übergangsfunktionen auf. Notieren Sie auch die Gleichung für $\overline{W}$.

Minimierung der KKNF ($Q2_t = x2$, $Q1_t = x1$, $Q0_t = x0$)

$D2_{t+1}$

$\overline{x_0}$ \ $\overline{x_2}$				
	0: x	2: x	6	4
	1	3: 0	7: 0	5

$\overline{x_1}$

$$D2_{t+1} = \overline{Q1_t} + \overline{Q0_t}$$

$D1_{t+1}$

$\overline{x_0}$ \ $\overline{x_2}$				
	0: x	2: x	6	4
	1: 0	3: 0	7	5: 0

$\overline{x_1}$

$$D1_{t+1} = Q2_t(Q1_t + \overline{Q0_t})$$

$D0_{t+1}$

$\overline{x_0}$ \ $\overline{x_2}$				
	0: x	2: x	6	4: 0
	1	3	7	5: 0

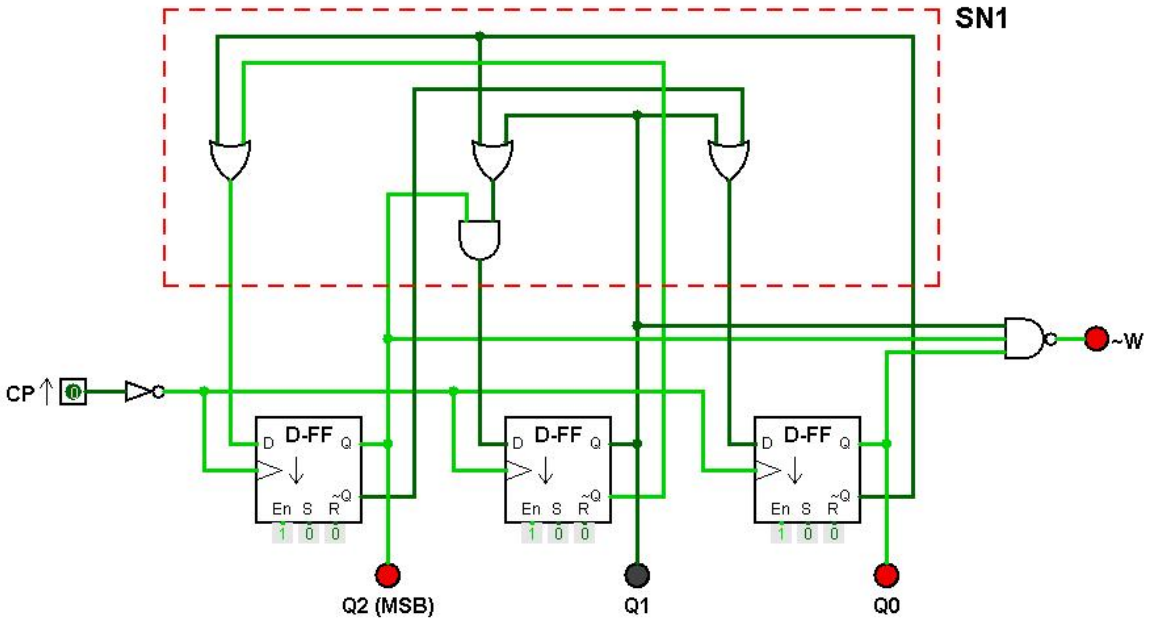
$\overline{x_1}$

$$D0_{t+1} = \overline{Q2_t} + Q1_t$$

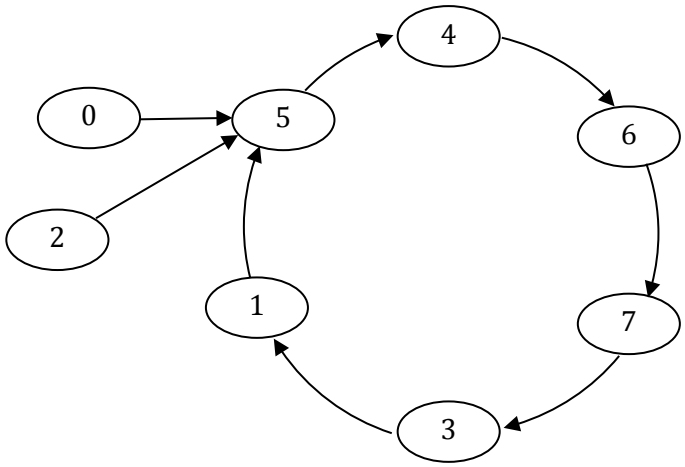
$$\overline{W} = \overline{Q2_t} + \overline{Q1_t} + \overline{Q0_t}$$

$$\overline{W} = \overline{Q2_t\, Q1_t\, Q0_t}$$

Aufgabe (mit Lösung):
Zeichnen Sie die Schaltung des Zählers unter Verwendung von negativ flankengetriggerten D-Flipflop.



Aufgabe (mit Lösung):
Zeichnen Sie das vollständige Zustandsdiagramm.



t			t+1		
Q2	Q1	Q0	D2	D1	D0
0	1	1	0	0	1
0	0	1	1	0	1
1	0	1	1	0	0
1	0	0	1	1	0
1	1	0	1	1	1
1	1	1	0	1	1
0	0	0	x=1	x=0	x=1
0	1	0	x=1	x=0	x=1

Pseudozufallsgeneratoren
Schieberegister können so beschaltet werden, dass Pseudozufallsgeneratoren entstehen. Für die Realisierung wird auf die Literatur verwiesen: z.B. [1].

Quellen

[1] Digitaltechnik, Klaus Fricke, Vieweg, 2005,
ISBN 3-528-33861-X

[2] Rechnerarchitektur, Helmut Malz, Vieweg, 2001,
ISBN 3-528-03379-7