3. ENTWURF SEQUENTIELLER LOGIK 3-2

3.1 EINFÜHRUNG IN DETERMINIERTE AUTOMATEN 3-2

3.2 BESCHREIBUNGSFORMEN FÜR AUTOMATEN 3-6

3.3 ELEMENTARE SEQUENTIELLE SCHALTUNGEN 3-10

3.3.1 RS-Basis-Flipflop 3-11
3.3.2 Latches 3-13
3.3.3 System-Flipflops 3-14
3.3.3.1 2-Taktpegel-Steuerung (Master-Slave-Flipflops) 3-15
3.3.3.2 Taktflanken-Steuerung 3-17

3.4 ZÄHLER (SYNCHRONE AUTONOME SYSTEME) 3-20

3.5 BEISPIELIMPLEMENTIERUNGEN: RECHENWERK UND STEUERWERK 3-26

3.5.1 Binäres Multiplizierwerk 3-26
3.5.2 Steuerwerk (Multiplikation) 3-29

3.6 IMPLEMENTIERUNGSASPEKTE 3-31

3. Entwurf sequentieller Logik

3.1 Einführung in determinierte Automaten

Für kombinatorische Logik hatten wir in Kap. 2 postuliert, dass sie keine Rückkopplungen aufweisen darf. In diesem Kapitel befassen wir uns mit der zweiten Kategorie von Logikschaltungen, die als Erweiterung auch Rückkopplungen enthalten. Diese werden als Schaltwerke oder allgemein als sequentielle Schaltungen bezeichnet. Erfolgt die Rückkopplung über so genannte Systemflipflops (vgl. Kapitel 3.3.3), dann spricht man vom synchronen Schaltwerk. An folgendem Beispiel einer simplen Aufzugssteuerung soll zunächst gezeigt werden, dass viele Problemstellungen allein mit kombinatorischen Schaltungen nicht lösbar sind.

Beispiel: Antriebssteuerung eines einfachen Aufzuges.

Es sei die Steuerung für den Antrieb eines Aufzuges zu entwerfen. Der Aufzug pendelt zwischen zwei Stockwerken. 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" □ aus
"1" □ an

y_1 Antriebsrichtung
"0" □ nach unten
"1" □ nach oben

Für eine Schaltnetzrealisierung ergäbe sich folgende Wahrheitstabelle:

Situation	x_1	x_0	y_1	y_0
0	0	0	?	1
1	0	1	1	1
2	1	0	0	1
3	1	1	x	0

Im Falle dass beide Stockwerkkontakte aktiv sind (Fehlerfall, Situation 3) soll der Antrieb abgeschaltet sein. In den Situationen, in denen die Kabine die eindeutige Endposition erreicht hat (Situation 1 oder 2) soll die Kabine in Richtung des anderen Stockwerkes fahren (nach hier nicht berücksichtigter Zeitverzögerung zum Ein-/Aussteigen). Die Antriebsrichtung ist somit klar definiert.

Wenn die Kabine zwischen zwei Stockwerken ist ($x_1 = x_0 = 0$), hängt die Antriebsrichtung von der "Vorgeschichte" ab, d. h. ob die Kabine gerade nach oben oder nach untern fährt. Dies ist mit einem reinem Schaltnetz nicht zu realisieren. Man könnte aber z. B. die aktuelle Antriebsrichtung y_1 auch als Eingang für das Schaltnetz nutzen, d. h. eine Rückkopplung vom Ausgang auf den Eingang durchführen.

Dies würde in diesem Beispiel möglich sein, da sich durch die einfache Betriebsweise folgender Ablauf ergibt:

Bewegungszustand:

		$y_{1(alt)}$	x_1	x_0	$y_{1(neu)}$	y_0
1.	Aufzug in Fahrt nach unten	0	0	0	0	1
2.	Aufzug unten angekommen, Antrieb umgeschaltet, Aufzug nach oben	0	0	1	1	1
3.	Aufzug nach oben, gerade unten losgefahren	1	0	1	1	1
4.	Aufzug in Fahrt nach oben	1	0	0	1	1
5.	Aufzug oben angekommen, Antrieb umgeschaltet, Aufzug nach unten	1	1	0	0	1
6.	Aufzug nach unten, gerade oben losgefahren	0	1	0	0	1

$\rightarrow$ weiter bei 1. (Kombinationen 011 und 111 nur im Fehlerfall)

Die Zuordnung der Antriebsumschaltung ist richtungseindeutig, daher könnte direkt rückgekoppelt werden. Dies ist jedoch nicht bei allen Problemstellungen der Fall.

Um Oszillationen zu vermeiden ist es sicherer die "Vorgeschichte" des Systems in eigenen Variablen, den Zustandsvariablen, zu beschreiben und zu realisieren.

Für das Beispiel benötigt man eine Variable z_0 für die zwei Bewegungszustände

$z_0 =$ "0" Fahrt nach unten
"1" Fahrt nach oben

Tabelle 3-1 : Automatenbeschreibung der Aufzugsteuerung

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

Tabelle 3-1 beschreibt dann vollständig den Automaten.
Die Steuerung lässt sich mit folgenden Elementen realisieren:

Ausgabefunktion: $Y = \lambda(X, Z^n)$

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

Bild 3-1 zeigt das allgemeine Blockschaltbild dazu.

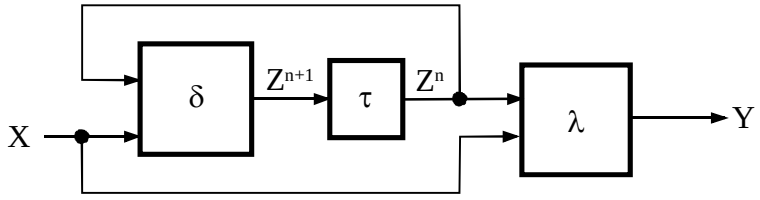


Bild 3-1: Blockschaltbild der Steuerung (Mealy-Automat)

Gegenüber der ersten Version ergibt sich eine Struktur aus zwei Schaltnetzen wobei das Schaltnetz für die Zustandsfortschaltung die Rückkopplung aufweist. Um hier sicher zu stellen, dass die Zustände auch zeitlich voneinander getrennt wechseln ist eine Zeitverzögerung (τ) in die Rückkopplung eingebaut. Ohne die Zeitverzögerung wäre die Zustandswechselzeit gleich der Durchlaufzeit durch das Schaltnetz δ (idealer Weise = 0).

Die Darstellung in Bild 3-1 hat den Vorteil, dass die Ausgangsfunktion λ unabhängig von der Zustandsfortschaltung definiert werden kann.

Diese Form eines Schaltnetzes wird als determinierter Automat A des Typs "Mealy-Automat" beschrieben:

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

Realisiert man den mit Tabelle 3-1 beschriebenen Automaten als Mealy-Automat, dann kommt man in optimierter Form zu folgenden Gleichungen.

Für das Schaltnetz δ zur Fortschaltung:

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

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

und für das Schaltnetz λ der Ausgangsfunktion

$$Y = \lambda(X, Z)$$

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

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

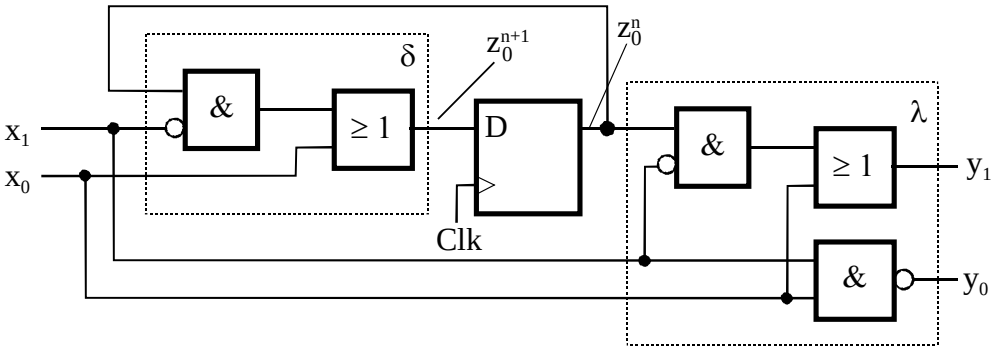


Bild 3-2: Realisierung der Aufzugssteuerung als Mealy-Automat

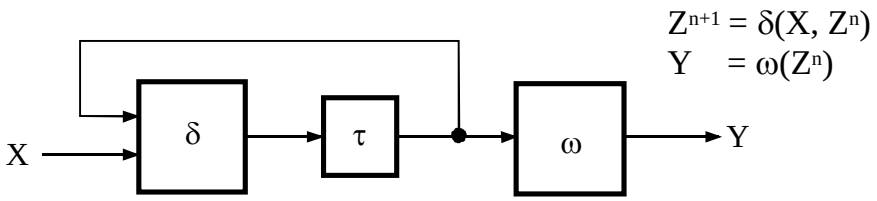
Die Verzögerung τ kann zum Beispiel mit einem flankengesteuerten D-Flipflop (siehe Kapitel 3.3.3.2) realisiert werden.

Bild 3-2 zeigt die vollständige Realisierung der beschriebenen Aufzugssteuerung als Mealy-Automat.

Neben dem in Bild 3-1 beschriebenen Mealy-Automaten unterscheidet man noch die in Bild 3-3 beschriebenen weiteren Typen.

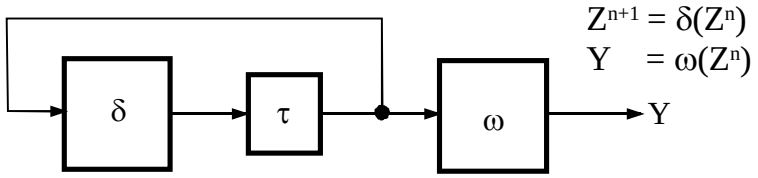
"Moore Automat"

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



"Autonomer Automat"

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



"Medwedjew -Automat"

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

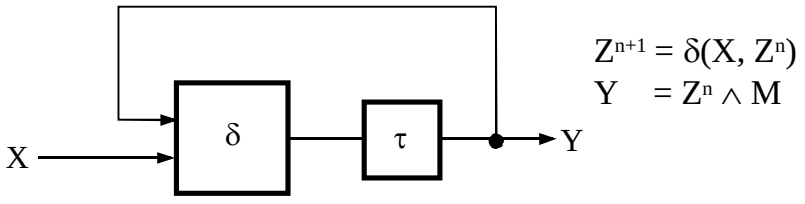


Bild 3-3: Gebräuchliche Automatentypen

Die gebräuchlichen Typen sind Mealy- und Moore-Automat. Beide lassen sich ineinander überführen. Eine zusätzliche Unterscheidung ergibt sich in der Berücksichtigung des Zustandes in der Ausgabefunktion. Die Ausgabefunktion kann auf dem aktuellen Zustand Z^n oder dem Folgezustand Z^{n+1} basieren (siehe Bild 3-4, Beispiel Moore-Automat). Hier werden nur Automaten verwendet, deren Ausgangsfunktion sich auf den aktuellen Zustand Z^n bezieht.

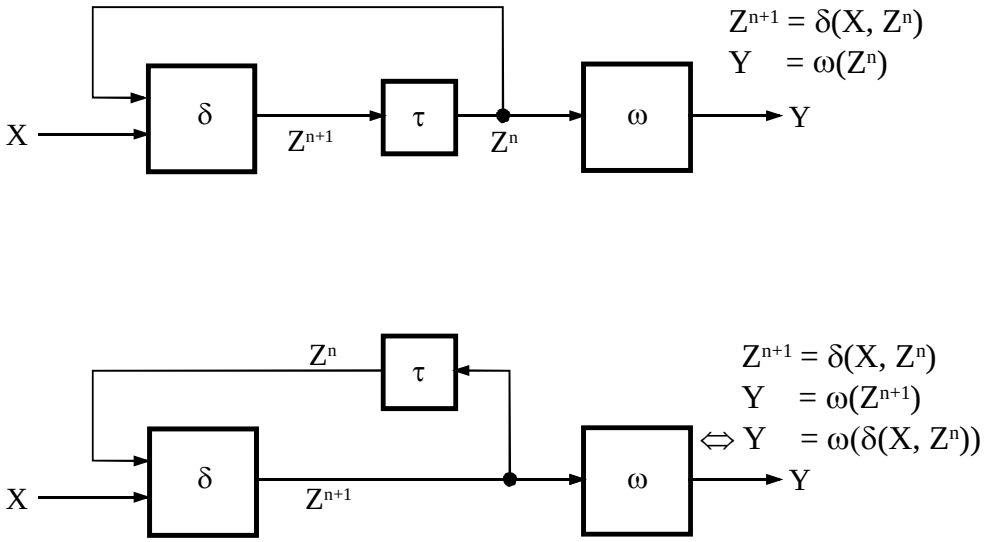


Bild 3-4: Moore-Automat mit Ausgang basierend a) auf aktuellem Zustand b) auf Folgezustand

3.2 Beschreibungsformen für Automaten

Ähnlich wie in Kap. 2 gibt es auch für die Schaltwerke bzw. für die sequentielle Logik verschiedene Beschreibungsformen:

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

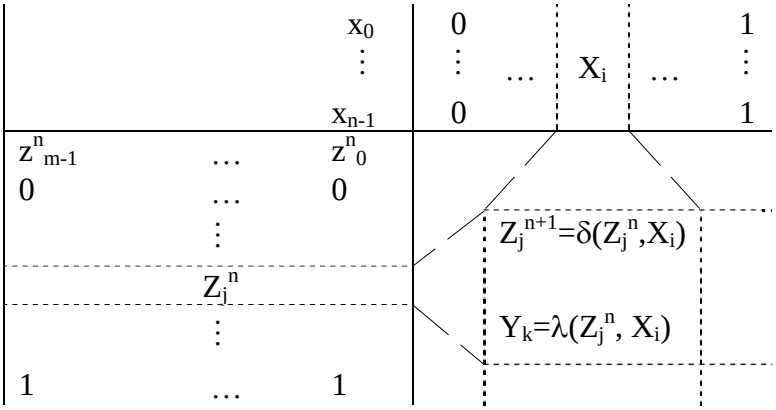


Bild 3-5: Beispiel für eine Automatentabelle (Mealy)

Diese Darstellung wird schnell komplex, wenn mehrere Eingänge und Zustände zu berücksichtigen sind.

Tabelle 3-2 Automatentabelle für die Aufzugssteuerung

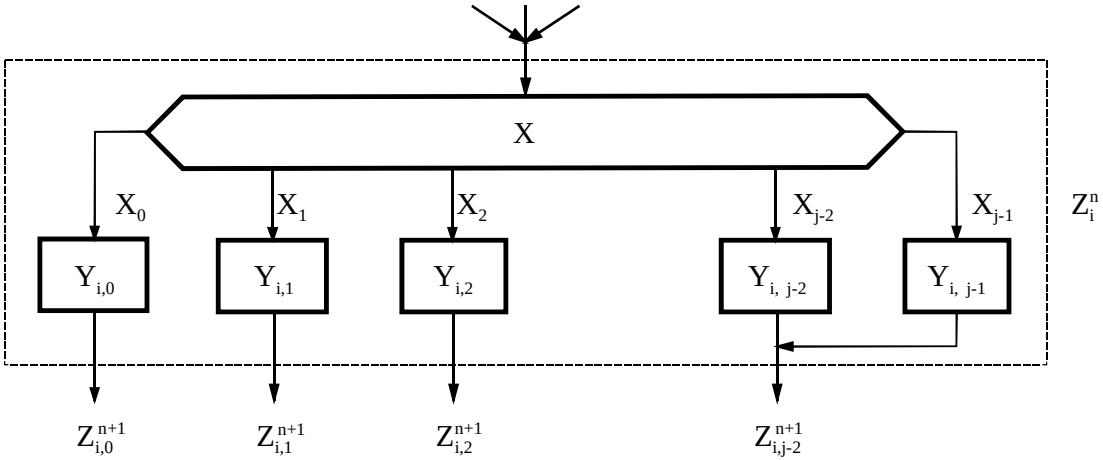
Zustand	z_0	x_0 x_1	0 0	1 0	0 1	1 1
Z_0	0		$Z^{n+1} = (0)$ $Y = (0,1)$	$Z^{n+1} = (1)$ $Y = (1,1)$	$Z^{n+1} = (0)$ $Y = (0,1)$	$Z^{n+1} = (-)$ $Y = (-,0)$
Z_1	1		$Z^{n+1} = (1)$ $Y = (1,1)$	$Z^{n+1} = (1)$ $Y = (1,1)$	$Z^{n+1} = (0)$ $Y = (0,1)$	$Z^{n+1} = (-)$ $Y = (-,0)$

Tabelle 3-2 zeigt für das Beispiel der Aufzugssteuerung die vollständige Automatentabelle.

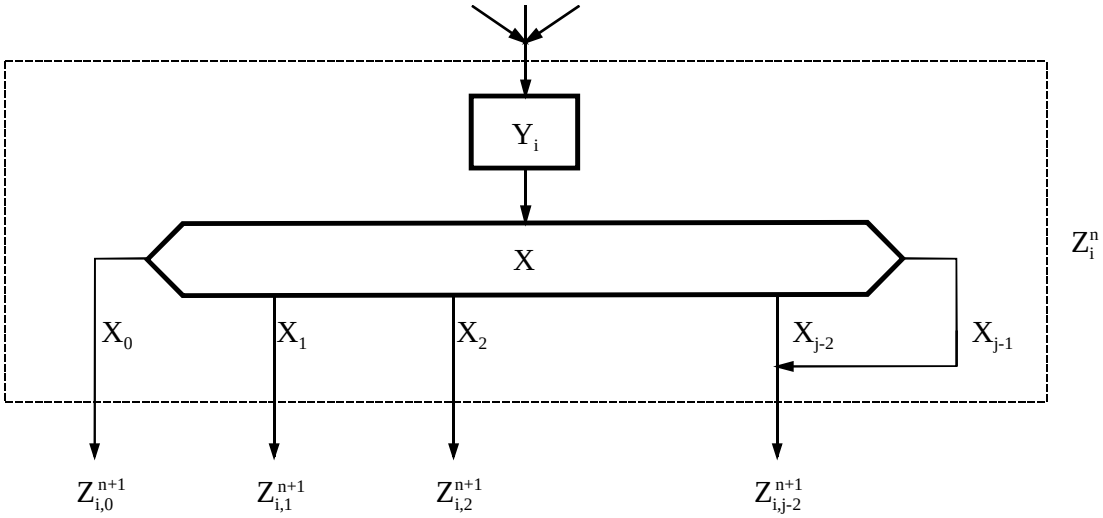
b) Ablaufdiagramm (Flussdiagramm)
Der Automat wird über den logischen Ablauf von seinen Zuständen definiert. Das folgende Bild zeigt die Beispiele für verschiedene Automaten s.

Bild 3-6.

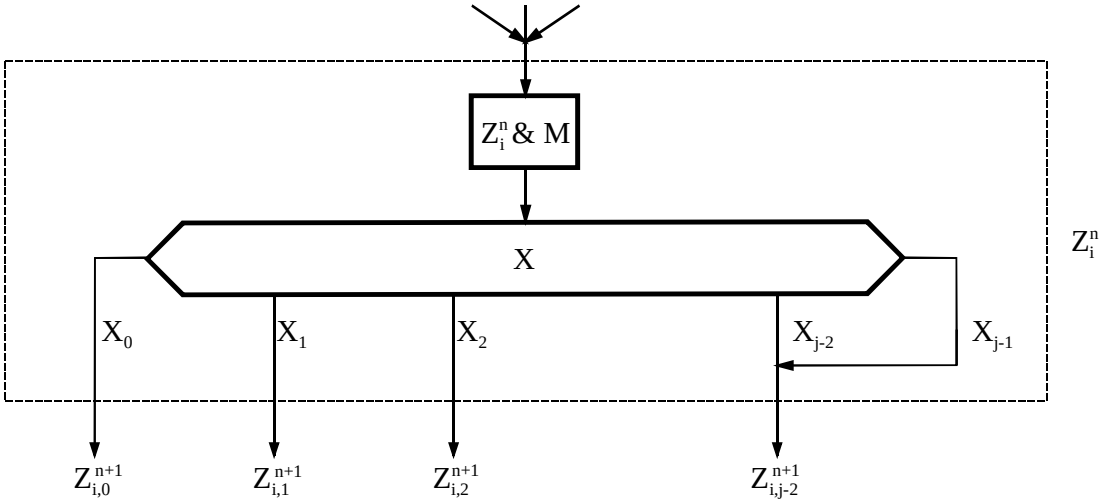
Damit lässt sich ein Automat ähnlich wie ein Programmablauf formal beschreiben. Allerdings führt u. U. eine hohe Anzahl von Verzeigungen und Einsprungstellen oft zu unübersichtlicher Darstellung.



a) Mealy-Automat



b) Moore-Automat



c) Medwedjew Automat

Bild 3-6**: Zustandsdarstellungen im Flussdiagramm**

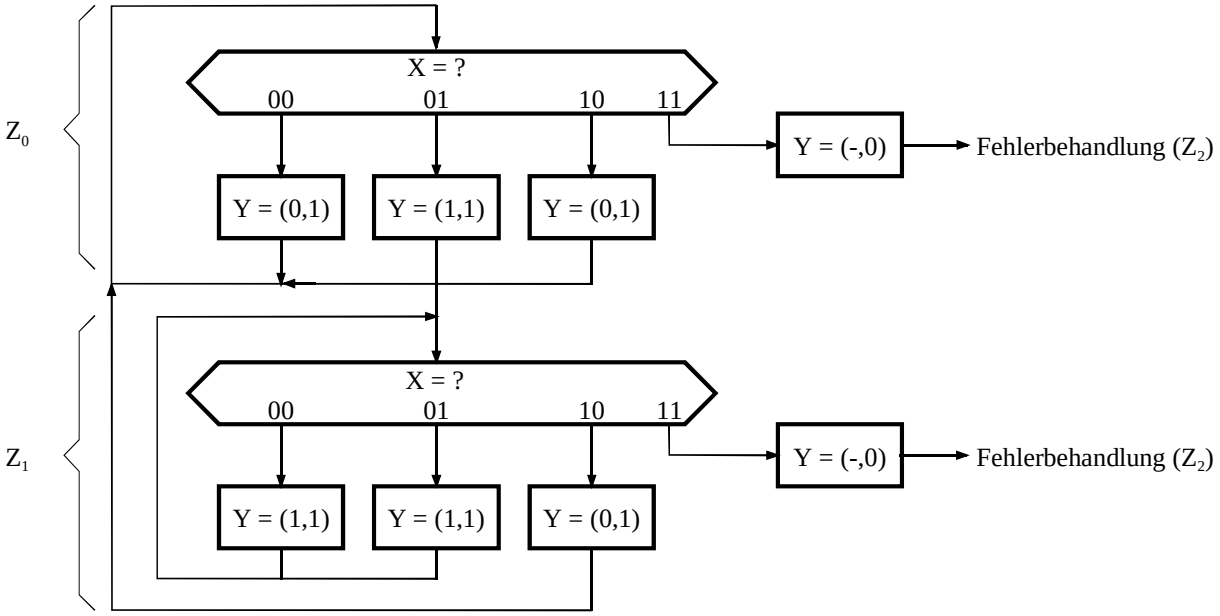


Bild 3-7: Flussdiagramm der Aufzugssteuerung

Bild 3-7 zeigt als Beispiel das Flussdiagramm der Aufzugssteuerung. Die Abfragen bzw. Aktionen wurden bereits mit Vektoren beschrieben. Im Flussdiagramm kann auch eine weniger formale Darstellung in Textform verwendet werden.

c) Zustands- oder Automatengraph
Ähnlich wie beim Flussdiagramm wird hier der Automat durch seine Zustände charakterisiert. Der Zustandsgraph ist ein gerichteter Graph mit den Zuständen Z als Knoten und den Eingangsbedingungen X mit den zugehörigen Ausgabewerten Y an den Kanten (s. Bild 3-8).

Bild 3-9 zeigt wiederum die Aufzugssteuerung als Beispiel .
Der Zustandsgraph hat eine Zustandsvariable z_0, die die Zustände Z_0, Z_1 binär in Z codiert.

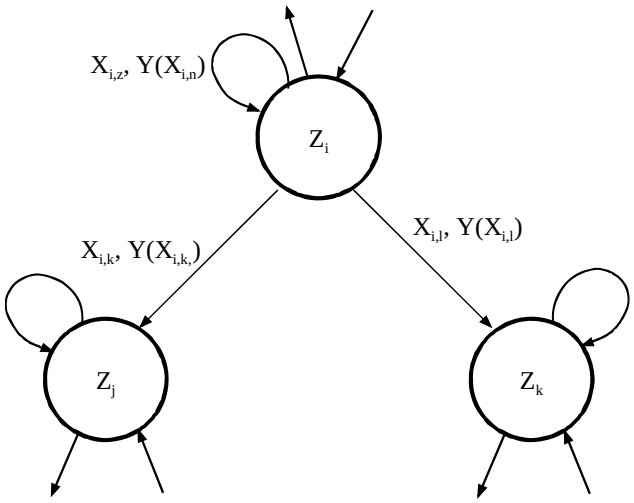


Bild 3-8: Beispiel für einen Zustandsgraph (Mealy)

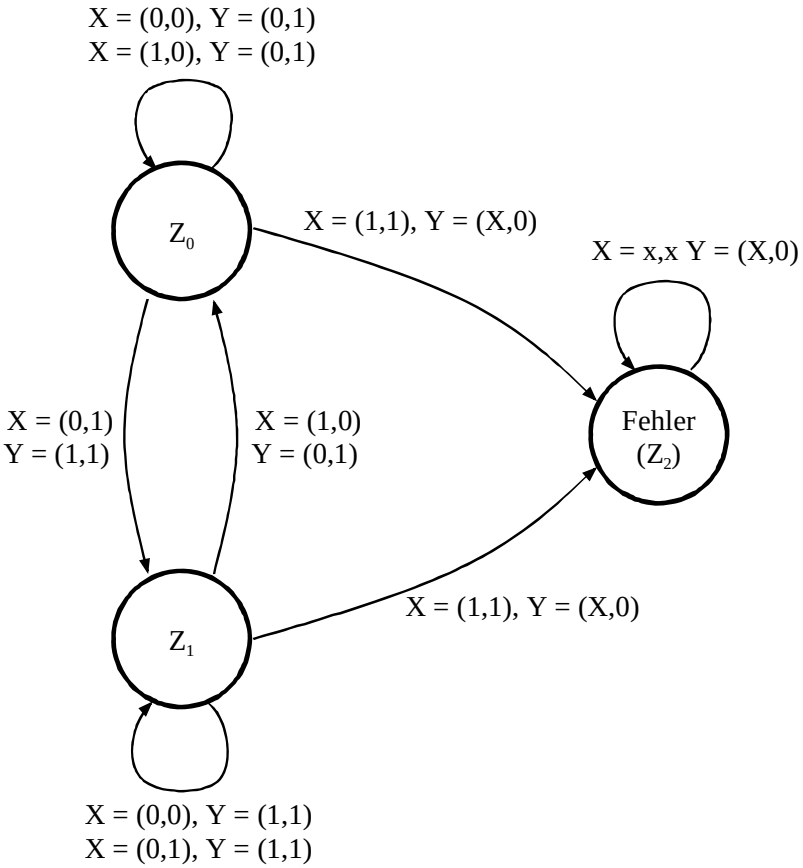


Bild 3-9: Zustandsgraph der Aufzugssteuerung

Wir wollen im folgenden die Automaten hauptsächlich durch Zustandsgraphen 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.

3.3 Elementare sequentielle Schaltungen

Eine elementare sequentielle Struktur enthält mindestens zwei Zustände. Diese Gruppe von sequentiellen Elementarstrukturen nennt man Flipflops.

Jedes Flipflop besitzt daher genau eine Zustandsvariable, hier Q genannt, die zwei Zustände annehmen kann (Q = 0 oder Q = 1). Der Ausgangswert ist dabei fest mit dem Zustandswert

gekoppelt und hängt nicht mehr von den Eingangswerten ab. Der Zustand kann durch eine Setz- und durch eine Rücksetzbedingung beeinflusst werden. Das Flipflop (FF) ist neben den Gattern das wichtigste Grundelement digitaler Schaltungen. Es wird je nach Typ als Grundbaustein von Registern oder Speichern, Zählern, Schieberegistern und Frequenzteilern und in synchronen Schaltwerken eingesetzt. Tabelle 3-3 zeigt eine Klassifikation der wichtigsten Flipfloptypen. Ausgehend von dem Basis-Flipflop werden die wichtigsten Flipflops im folgenden beschrieben.

Tabelle 3-3 Flipflop-Klassifikation

Flipflop				
ungetaktet	getaktet			
	pegelgesteuert			flankengesteuert
	Ein-Pegel-gesteuert (transparent)	Zwei-Pegel-gesteuert (Master-Slave)		
		irreversibel	reversibel	
RS	RS	-	RS	RS
-	D	-	D	D
-	-	JK	-	JK
Basis-Flipflop	Latch	System-Flipflop		

3.3.1 RS-Basis-Flipflop

Wenn man den Ausgang eines Gatters oder eines Schaltnetzes wieder auf den eigenen Eingang zurückführt, dann erhält man eine Abhängigkeit der Stufe zum eigenen aktuellen Zustand. Diese Abhängigkeit kann sich in einem stabilen Verhalten (Speichern) oder in einem instabilen Verhalten (Oszillationen) auswirken.

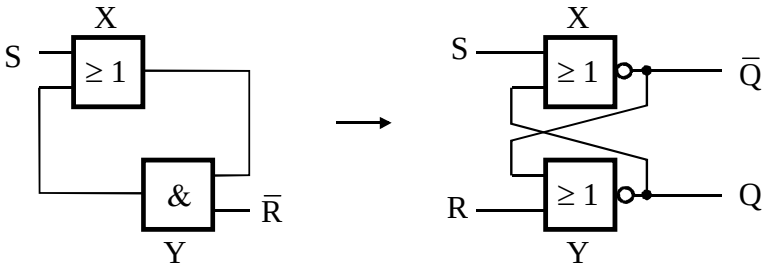
Setzt man in die Rückkopplungsschleife des ODER-Gatters in Bild 5.2a noch ein UND-Gatter mit einem Steuereingang, so ergibt sich ein Element mit zwei einstellbaren stabilen Zuständen. Man bezeichnet einen Punkt der Ringstruktur als den Ausgang mit dem Namen "Q" und spricht von einem "gesetzten Flipflop", wenn Q = l ist, und von einem "rückgesetzten Flipflop", wenn Q = 0 ist.

In Abhängigkeit von den beiden Steuereingängen S (set) und R (reset) können die Funktionen, Setzen, Rücksetzen und Speichern ausgeführt werden.

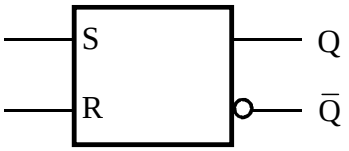
Dieses Element ist das RS-Basis-Flipflop. Eine Umformung nach De Morgan führt zu der bekannten Struktur aus zwei kreuzgekoppelten NOR-Gattern (Bild 3-10b). Das Schaltsymbol und die zu dieser Schaltung gehörende Wahrheitstabelle sind in Bild 3-10c und Bild 3-10.d dargestellt.

Wenn beide Eingänge im Ruhezustand sind (S,R) = (0,0), dann wird der aktuelle Zustand gespeichert, er ist gleich dem "alten Zustand" ($Q = Q_{old}$). Ist nur S aktiv, (S,R) = (1,0), dann wird das

Flipflop gesetzt ((Q, $\overline{Q}$) = (1,0)). Ist nur R aktiv, (S,R) = (0,1), dann wird es zurückgesetzt ((Q, $\overline{Q}$) = (0,1)).



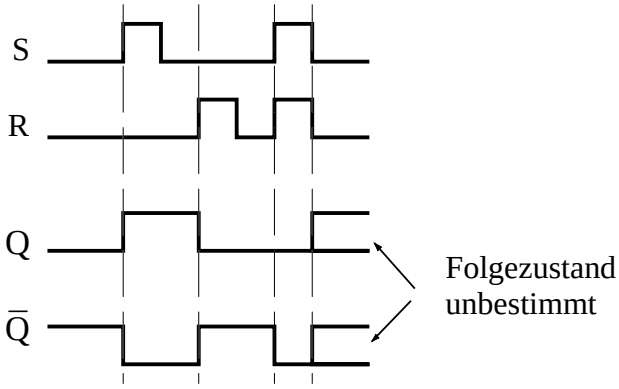
(a) (b)



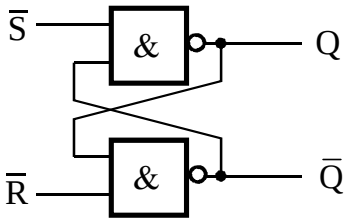
(c)

S	R	Q	$\overline{Q}$	
0	0	Q_{old}	$\overline{Q}_{old}$	Speichern
0	1	0	1	Rücksetzen
1	0	1	0	Setzen
1	1	0	0	nicht erlaubt

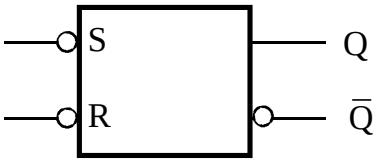
(d)



(e)



(f)



(g)

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
1	1	Q_{old}	$\overline{Q}_{old}$	Speichern
1	0	0	1	Rücksetzen
0	1	1	0	Setzen
0	0	1	1	nicht erlaubt

(h)

Bild 3-10: RS-Basis-Flipflop

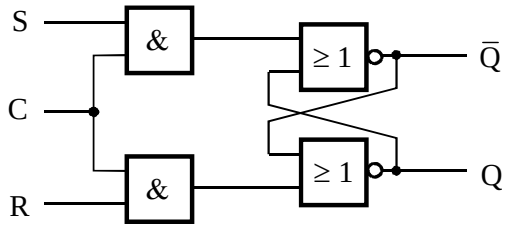
Wenn beide Eingänge gleichzeitig aktiv sind, (S,R) = (1,1), dann sind beide Ausgänge auf 0 ((Q, $\overline{Q}$)= (0,0)). Dies ist ein besonderer Zustand, bei dem die beiden Ausgänge nicht invers zueinander sind. Schon aus diesem Grunde könnte man diesen Sonderzustand als unzulässig betrachten. Problematischer ist allerdings der Übergang in die Speicherfunktion, (S,R) = (1,1) → (0,0) (vgl. Bild 3-10e, Timing Diagramm). Theoretisch geht das Element in den sog. "metastabilen" Zustand. Aufgrund kleiner Unsymmetrien in der Ansteuerung oder in der Schaltung stellt sich aber praktisch einer der beiden möglichen stabilen Zustände ein (entweder (Q, $\overline{Q}$) = (0,1) oder (Q, $\overline{Q}$) = (1,0)). Da der Folgezustand in jedem Falle unbestimmt ist, wird die Ansteuerung mit (S,R) = (1,1) als nicht erlaubt angesehen. Bild 3-10f, Bild 3-10g und Bild 3-10h zeigen Schaltung, Symbol und Wahrheitstabelle für die Version des RS-Basis-Flipflops aus zwei kreuzgekoppelten NAND-Gattern.

3.3.2 Latches

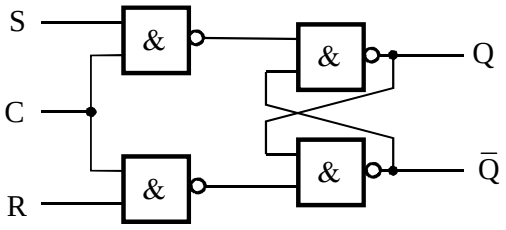
RS-Latch (Ein-Taktpegel-gesteuertes RS-Flipflop)

Um zu erreichen, dass Zeitbedingungen für die Eingangssignale erfüllt werden können, d.h. die Informationen nur zu bestimmten, durch ein „Takt“-Signal (C) definierten Zeiten übernommen werden können, werden zwei UND-Gatter vor die beiden Eingänge des Basis-Flipflops geschaltet (Bild 3-11a). Damit kann dieses Flipflop in Systemen zum Halten von Daten eingesetzt werden während vorgelagerte Schaltnetze die Kombination ändern. Man nennt dieses taktpegelgesteuerte RS-Flipflop auch "RS-Auffangflipflop" oder üblicherweise RS-Latch.

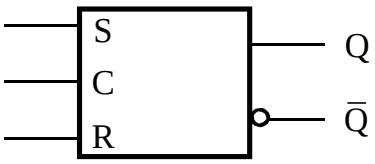
Solange C auf 0 ist, können R und S beliebige Pegel annehmen, ohne den Zustand des Flipflops zu beeinflussen. Für C = 1 verhält sich das Latch wie ein Basis-Flipflop und es gilt die entsprechende Wahrheitstabelle (für die NAND-Version (b)). Da also das „Takt“-Signal nicht das Umschalten des Ausgangs zu genau definierten Zeitpunkten ermöglicht, sondern im Gegenteil solange C =1 ist, das Flipflop direkt durch S und R gesteuert wird, ist dieses Flipflop transparent. Aus diesem Grund wird das „Takt“-Signal hier besser als Steuersignal (abgekürz C für Control) bezeichnet.



(a)



(b)

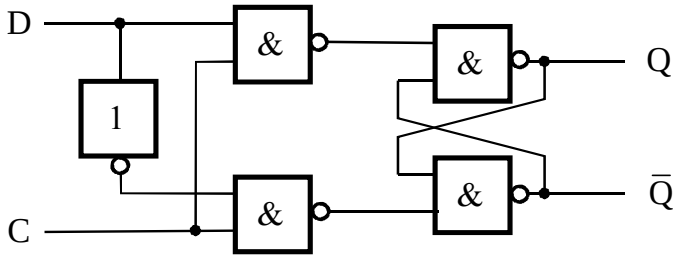


(c)

R	S	C	Q	$\overline{Q}$	
X	X	0	Q_{old}	$\overline{Q}_{old}$	} Speichern
0	0	1	Q_{old}	$\overline{Q}_{old}$	
0	1	1	1	0	Setzen
1	0	1	0	1	Rücksetzen
1	1	1	1	1	nicht erlaubt

Bild 3-11: RS-Latch (Ein-Taktpegel-gesteuertes RS-Flipflop)

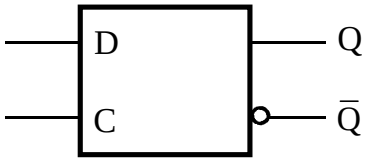
D-Latch (Ein-Taktpegel-gesteuertes D-Flipflop)
Wenn man den S- und den R-Eingang des RS-Latches über einen Inverter miteinander verbindet, dann erhält man das in Bild 3-12a dargestellte D-Latch. Durch diese Beschaltung ist die metastabile Ansteuerung (S,R) = (1,1) ausgeschlossen. Allerdings gibt es nun auch keinen Ruhezustand des Eingangs mehr ((S,R) = (0,0)), so dass solange das Freigabesignal C = 1 ist, das anliegende Eingangssignal zum Ausgang durchgegeben wird. Der Ausgang ist also ein zeitlich verzögertes Abbild des Eingangs, daher der Name "D" wie "Delay".



(a)

D	C	Q	$\overline{Q}$	
X	0	Q_{old}	$\overline{Q}_{old}$	Speichern
0	1	0	1	Rücksetzen
1	1	1	0	Setzen

(b)



(c)

Bild 3-12: D-Latch

3.3.3 System-Flipflops

Um einwandfreies Arbeiten eines Schaltwerkes zu gewährleisten sollten sowohl die Eingänge als auch die Zustandsfortschaltung synchronisiert werden. Dazu werden so genannte System-Flipflops benötigt. Sie zeichnen sich dadurch aus, dass der Ausgang sich zu genau definierten, möglichst kurzen Zeitpunkten eines Taktsignals ändert und die Änderung der Ausgangskombination zeitlich von der Änderung der Eingangskombination entkoppelt ist.

Ein einfaches Latch (nach Bild 3-11 oder Bild 3-12) genügt dieser Bedingung nicht, weil nach Aktivierung des Eingangs durch den Freigabeimpuls bereits nach zwei minimalen Gatterlaufzeiten eine Zustandsänderung am Ausgang (Q) erscheinen kann.

Es gibt generell zwei Möglichkeiten der Realisierung dieser Bedingung:

1. Statisch durch 2-Taktpegel-Steuerung (Master-Slave)

2. Dynamisch durch Taktflanken-Steuerung

Nach DIN EN 60617 Teil 12 wird zwischen diesen beiden Ansteuerungsarten auch in den Symbolen unterschieden. Die hier mit Takt*pegel*steuerung bezeichneten Flipflops werden in der Literatur (incl. DIN) als takt*zustands*gesteuerte Flipflops bezeichnet. Diese Bezeichnungsweise hat den Nachteil das sie mit dem Zustand von Flipflops (gesetzt, nicht gesetzt) verwechselt werden kann. Andererseits ist der Begriff Pegel für die physikalische Repräsentierung reserviert. Wir werden hier

trotzdem diesen Begriff verwenden. Mit 2-taktpegelgesteuerten als auch mit flankengesteuerten Typen kann man RS-, JK und D-Flipflops realisieren (Tabelle 3-3). Im folgenden werden die einzelnen Flipflop-Typen detailliert beschrieben.

3.3.3.1 2-Taktpegel-Steuerung (Master-Slave-Flipflops)

Die 2-Taktpegel-Steuerung wird unter Verwendung eines statischen Zwischenspeichers realisiert, d.h. man schaltet zwei Latches (mit versetzten Takten) in Reihe. Eine solche Struktur wird als "Master-Slave-Flipflop" bezeichnet. Sie bietet den Vorteil, dass in ihr einfache taktpegelgesteuerte Flipflops (Latches) verwendet werden können, die für sich genommen nicht die Systembedingung erfüllen. Diese Art der Taktansteuerung wird in integrierten Schaltungen als Zweiphasen-Takt verwendet, d.h. man setzt generell zwei getrennte Taktnetze ein.

RS-Master-Slave-Flipflop

Bild 3-13 zeigt ein RS-Master-Slave-Flipflop in der Darstellung aus zwei taktpegelgesteuerte RS-Latches mit invertierter Taktzuführung (a) und das zugehörige Schaltsymbol (b). Seine Funktionsweise kann im allgemeinen folgendermaßen beschriebenwerden:

Solange Ck auf 1 liegt, ist der Eingang des Master-Flipflops freigegeben und der des Slave-Flipflops gesperrt. Am Ausgang liegt also der im Slave gespeicherte Zustand unabhängig vom Zustand des Masters. Solange Ck auf 1 liegt kann das Flipflop mit den S- und R-Eingängen beliebig gesetzt und zurückgesetzt werden, d.h. das Flipflop ist reversibel. Wenn Ck auf 0 geht, wird der Eingang des Masters gesperrt und der zuletzt bestehende Zustand gespeichert. Außerdem wird der Slave geöffnet, so dass der Ausgang des Masters den Slave-Zustand und damit den Ausgang bestimmt.

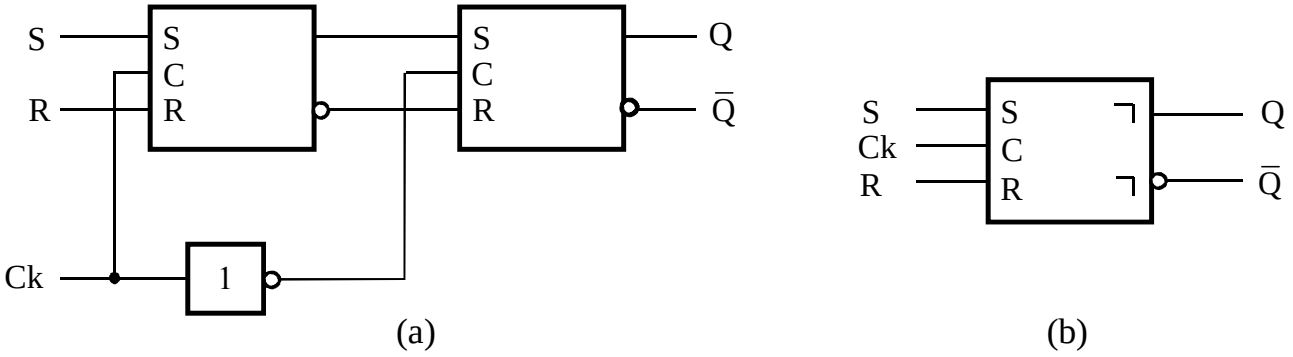


Bild 3-13: RS-MS-Flipflop

Es scheint also nach außen im Wesentlichen nur eine Taktflanke wirksam zu sein (in diesem Fall die negative), obwohl das nicht ganz den Tatsachen entspricht. Das Timing Diagramm in Bild 3-14 zeigt warum. Wird der Setz-Eingang irgendwann während der Takt auf 1 liegt aktiviert, dann wird nach der fallenden Flanke der Slave gesetzt und Q = 1. Allerdings reicht ein kurzer Impuls, wie hier beim Rücksetzen skizziert, aus, um das Flipflop mit der nächsten fallenden Flanke zurückzusetzen. Man spricht von einem so genannten retardierten Ausgang, d. h. der Ausgang nimmt den inneren Zustand erst an wenn der Takt den Ruhepegel wieder erreicht.

Dies ist durch den Aufbau dieses Flipflops begründet, und unterscheidet sich wesentlich vom flankengesteuerten Flipflop. Wenn wie weiter skizziert während desselben Taktimpulses das Flipflop gesetzt und wieder zurückgesetzt werden kann, dann spricht man von reversiblem Flipflop. Das RS-MS-FF gehört zu dieser Gruppe.

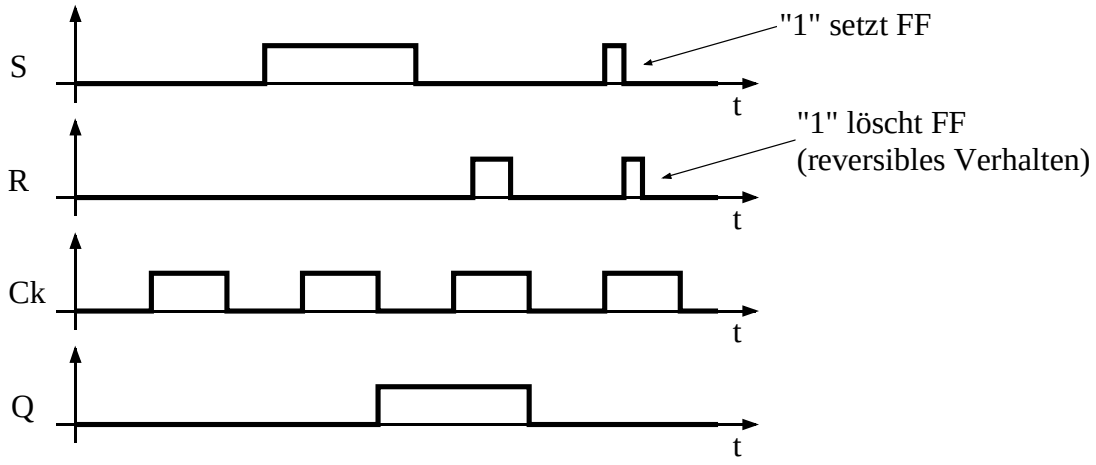
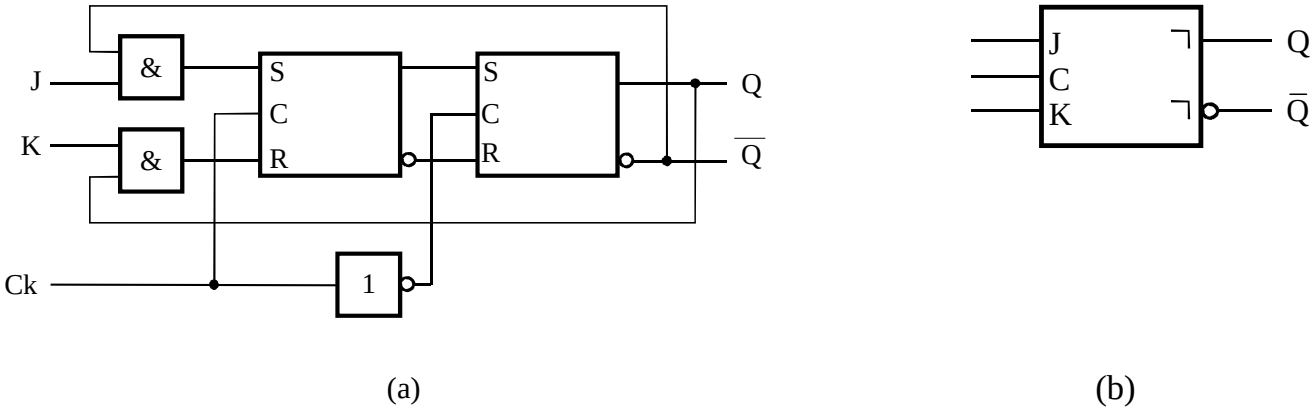


Bild 3-14: **Timing-Diagramm des RS-MS-Flipflops**

JK-Master-Slave-Flipflop

Nachdem – per Definition – ein Systemflipflop zum Eingang rückgekoppelt werden kann, ist es möglich, die Ausgänge über Kreuz zurückzukoppeln und sie als zusätzliche Bedingung für das Setzen und Rücksetzen zu benutzen (Bild 3-15 zeigt dies in der Master-Slave-Version). Die Eingänge nennt man J und K. Damit wird erreicht, dass der Fall der Eingangskonfiguration ((J, K) = (1, 1)) nicht verboten werden muss, sondern die Funktion des "Wechselns" bewirkt. Dadurch wird das Flipflop aber irreversibel, denn wenn am J-Eingang ein kurzer 1-Impuls anliegt, um das Flipflop zu setzen, dann ist es aufgrund der kreuzweisen Verriegelung nicht möglich, das Flipflop wieder zurückzusetzen (Bild 3-16). Das geht erst nach dem nächsten Taktwechsel.



(a) (b)

Bild 3-15: JK-Master-Slave-Flipflop

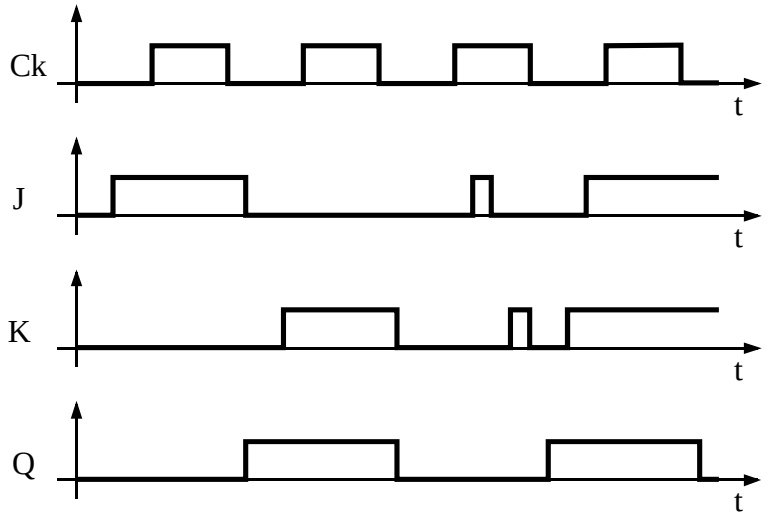


Bild 3-16: Timing-Diagramm JK-Master-Slave-Flipflop

D-Master-Slave-Flipflop

Verbindet man wie beim D-Latch die beiden Eingänge eines RS-Master-Slave-Flipflops über einen Inverter, so entsteht ein D-Master-Slave-Flipflop (Bild 3-17). Da wie beim D-Lastch der Zustand des Masters dem Eingangspegel am D-Eingang direkt folgt, kann sich der Eingang bis direkt vor die fallende Flanke des Taktes ändern, d.h. das D-Master-Slave-Flipflop lässt sich vom taktflankengesteuerten D-Flipflop in der Funktion nicht unterscheiden.

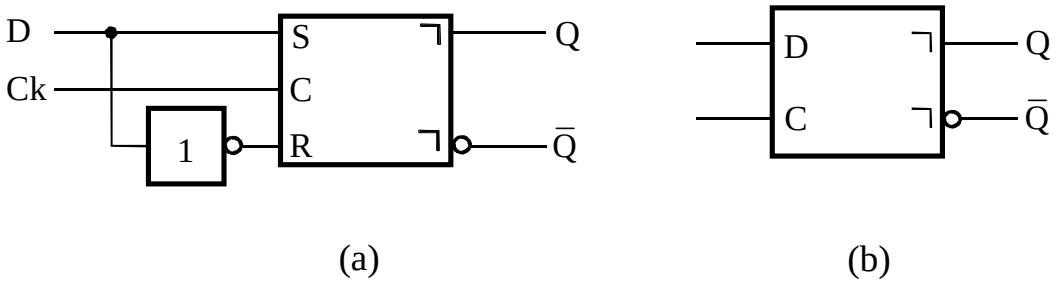


Bild 3-17: D-Master-Slave-Flipflop

3.3.3.2 Taktflanken-Steuerung

Die Taktflankensteuerung ist das heute häufiger verwendete Verfahren zur Entkopplung der Eingänge von den Ausgängen. Dazu wird die Taktflanke differenziert, d.h. man formt aus der Taktflanke einen kurzen Impuls.

Bild 3.15(a) zeigt ein so genanntes Impulsgatter, das dieses Verhalten realisiert, wie das Timing-Diagramm in Bild 3.15(b) deutlich macht. Immer wenn der Eingang V auf 1 liegt dann erscheint bei einer positiven Taktflanke ein kurzer Impuls am Ausgang. Dieses Verhalten kann mit so genannten Laufzeitdifferenzierern erzeugt werden.

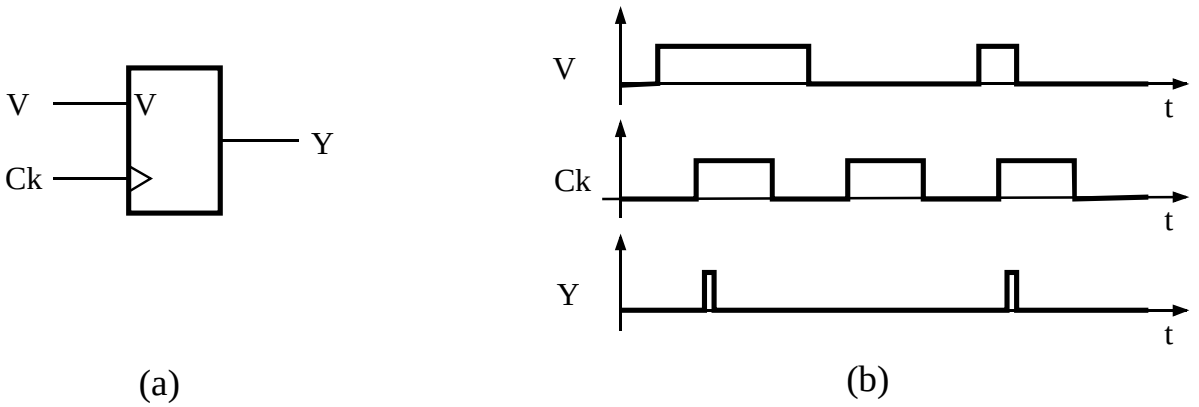


Bild 3-18: Impulsgatter

Flankengesteuertes RS-Flipflop

Setzt man bei einem RS-Basisflipflop vor den S- und den R-Eingang jeweils ein Impulsgatter, dann erhält man ein flankengesteuertes RS-Flipflop (Bild 3-19).

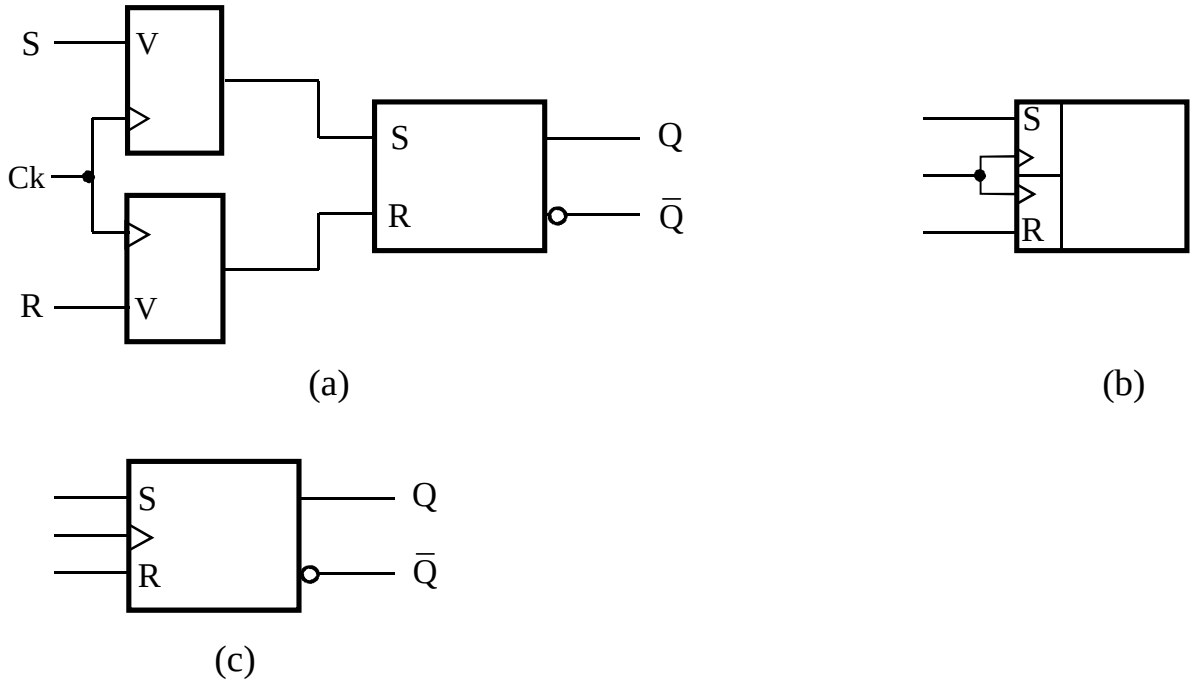


Bild 3-19 Flankengesteuertes RS-Flipflop

Immer, wenn während der positiven Flanke des Taktes der Setzeingang aktiv ist, (S,R) = (1,0), dann wird das Flipflop gesetzt und entsprechend wenn der Rücksetzeingang aktiv ist (S,R) = (0,1), dann wird das Flipflop zurückgesetzt. Solange beide Eingänge inaktiv sind wird wie beim RS-Basisflipflop die Information gespeichert. Dass beide Eingänge aktiv sind, (S,R) = (1,1), ist ebenfalls wie beim Basisflipflop nicht zulässig. Bild 3-20 zeigt das Timingdiagramm dazu. Im Vergleich zum RS-Master-Slave-Flipflop fällt auf, dass kurze Impulse auf den Eingängen wirkungslos sind, und nur noch die Eingangskombination während der Taktflanke von Interesse ist.

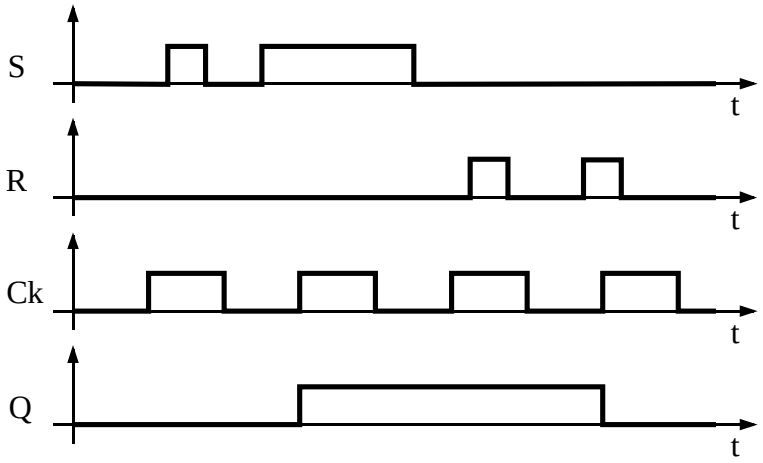


Bild 3-20: **Timing-Diagramm für flankengesteuertes RS-Flipflops**

Flankengesteuertes JK-Flipflop

Genau wie beim JK-Master-Slave-Flipflop ist es möglich, durch kreuzweises Rückführen der Ausgänge die unzulässige Eingangskombination (S,R) = (1,1) beim RS-Flipflop zu vermeiden und in die zusätzliche Funktion des Wechsels umzuwandeln (Bild 3-21). Die logische Funktion des flankengesteuerten JK-Flipflops entspricht dabei seinem Gegenstück, dem JK-Master-Slave-Flipflop, mit der Ausnahme, dass das Flipflop mit Flankensteuerung jetzt reversibel ist. Ein flankengesteuertes Flipflop ist immer reversibel, da es nur während der Taktflanke die Information an den Eingängen übernimmt.

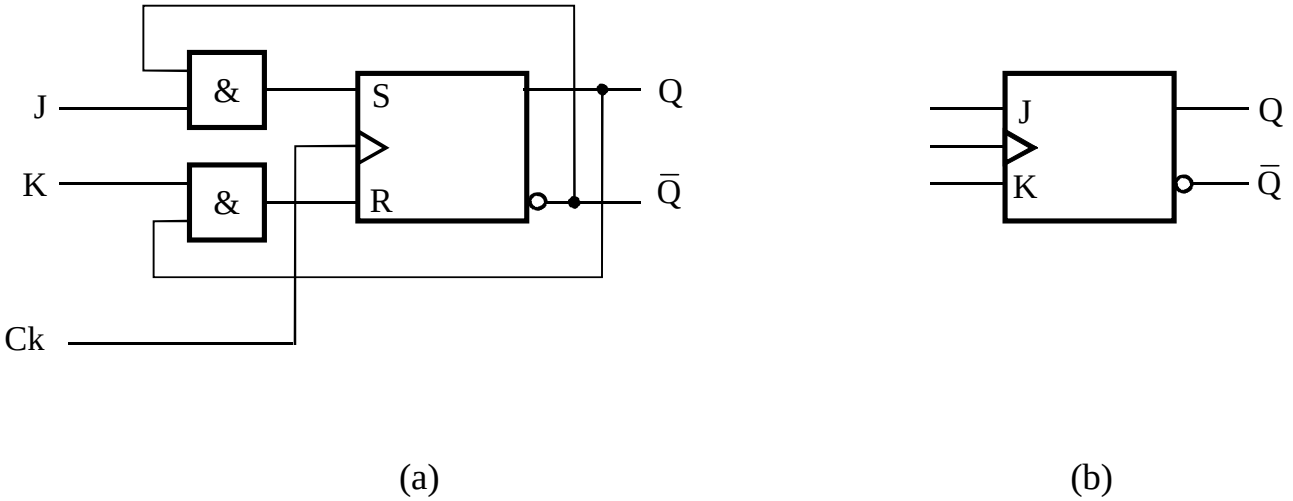


Bild 3-21: **Flankengesteuertes JK-Flipflop**

Flankengesteuertes D-Flipflop

Verbindet man wie beim D-Latch und wie beim D-Master-Slave-Flipflop die beiden Eingänge eines flankengesteuerten RS-Flipflops über einen Inverter, so entsteht ein flankengesteuertes D-Flipflop (Bild 3-22). Da wie beim D-Latch der Zustand des Masters dem Eingangspegel am D-Eingang direkt folgt, kann sich der Eingang bis direkt vor die fallende Flanke des Taktes ändern, d.h. das D Master-Slave Flipflop lässt sich vom taktflankengesteuerten D-Flipflop in der Funktion nicht unterscheiden.

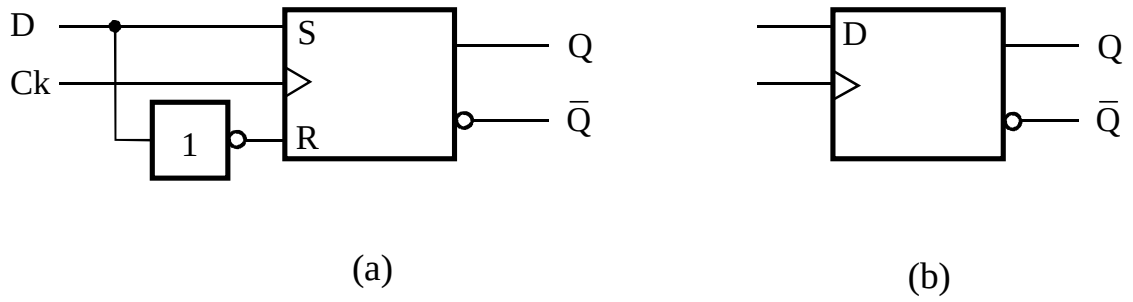


Bild 3-22: Flankengesteuertes D-Flipflop

3.4 Zähler (synchrone autonome Systeme)

Die nächst komplexere Gruppe von sequentiellen Schaltungen sind die Zähler. Zähler können als autonome Automaten mit rückgekoppelten Schaltnetzen realisiert werden (Bild 3-23). 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 (Tabelle 3-4). Andere Zähler können 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 Tabelle 3-4 ist als Beispiel die Übergangstabelle eines vorwärtszählenden Binärzählers angegeben.

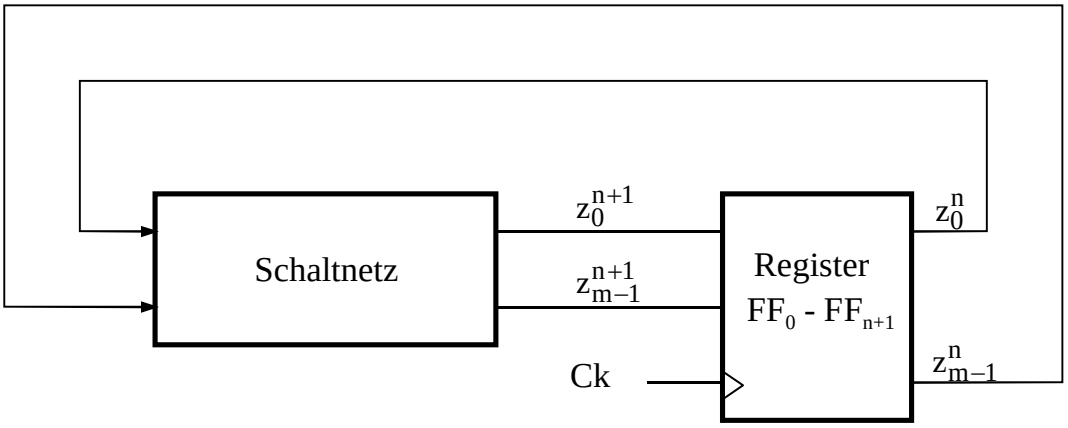


Bild 3-23: Realisierung von Zählern

Tabelle 3-4 Flusstabelle für einen vorwärtszählenden Binärzähler

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

Aus dieser Übergangstabelle kann man direkt die Bool'schen Gleichungen des Schaltnetzes ablesen.

Bei einem n-stelligen Binär-Zähler enthält die Flusstabelle 2^n verschiedene Eingangskombinationen. 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.

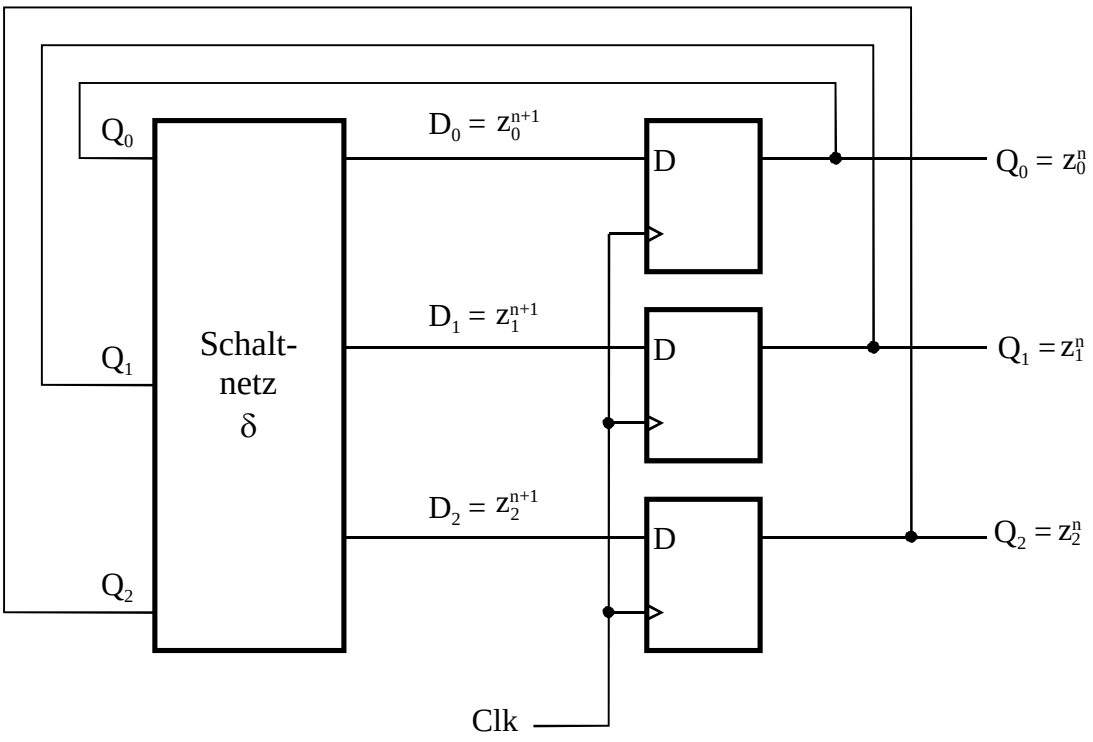


Bild 3-24: Allgemeines Schaltwerk eines 3-Bit-Zählers

Bild 3-24 zeigt das Schaltwerk eines 3-Bit-Zählers mit dem Rückkoppelschaltnetz δ und dem 3-Bit Register aus D-Flipflops. Hier soll nun das Rückkoppelschaltnetz hergeleitet. Dazu stellt man zweckmäßigerweise die zugehörige Flusstabelle auf (Tabelle 3-5).

Tabelle 3-5 Flusstabelle eines 3-Bit-Binärzählers

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

Der Zustand Z^n aus der Zustandstabelle (z_2^n , z_1^n , z_0^n) ist dabei das was im Bild 3-24 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 Tabelle 3-5 abgelesen werden:

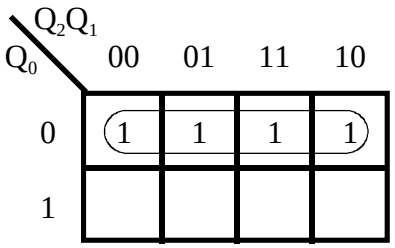
$$D_0 = \bar{Q}_2\bar{Q}_1\bar{Q}_0 \vee \bar{Q}_2Q_1\bar{Q}_0 \vee Q_2\bar{Q}_1\bar{Q}_0 \vee Q_2Q_1\bar{Q}_0$$

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

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

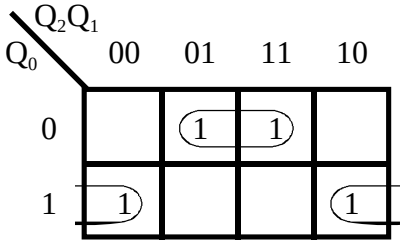
Zweckmäßigerweise wird ein solches Schaltnetz vor der Implementierung optimiert, z.B. nach Karnaugh:

D_0:



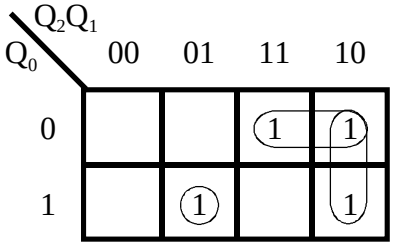
$D_0 = \bar{Q}_0$

D_1:



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

D_2:



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

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

$$D_0 = Q_0 \oplus 1$$

$$D_1 = Q_1 \oplus Q_0$$

$$D_2 = Q_2 \oplus \left(Q_0 Q_1\right)$$

Diese Reihe ließe sich fortsetzen:

$$D_3 = Q_3 \oplus \left(Q_0 Q_1 Q_2\right)$$

$$D_4 = Q_4 \oplus \left(Q_0 Q_1 Q_2 Q_3\right)$$

usw.

Mit den aufgestellten Gleichungen kann jetzt das Schaltnetz in Bild 3-24 vervollständigt werden (Bild 3-25).

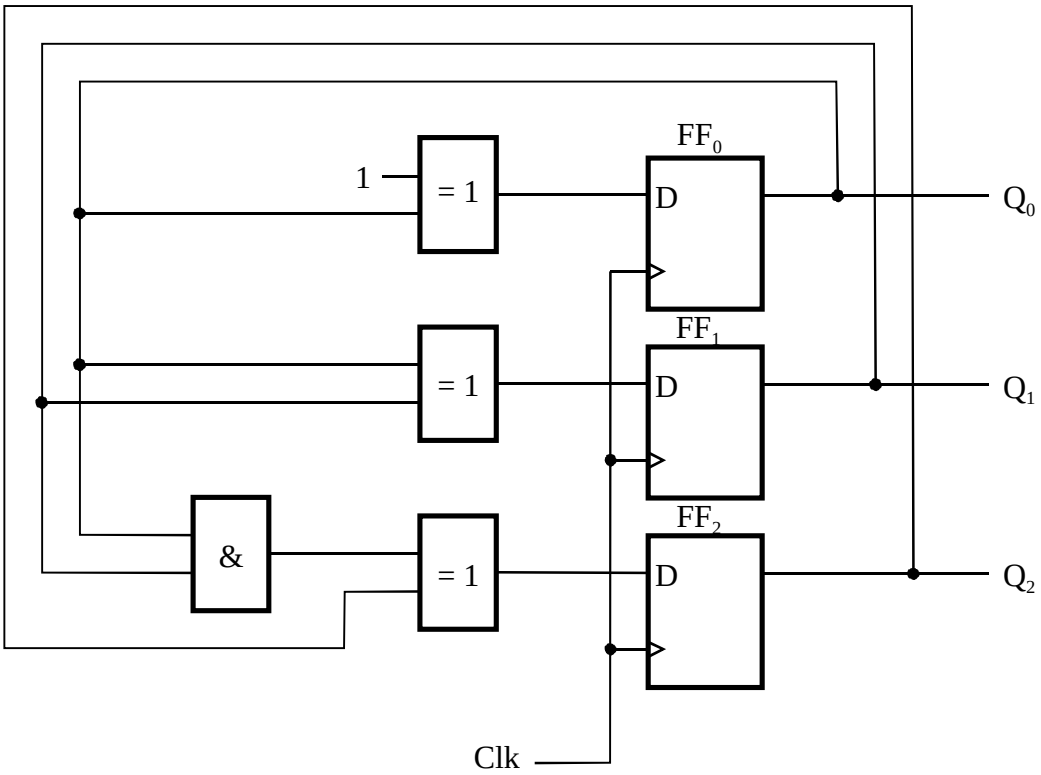


Bild 3-25: Synchroner binärer 3-bit-Zähler (Modulo-8-Zähler)

Die Darstellung kann weiter vereinfacht werden.

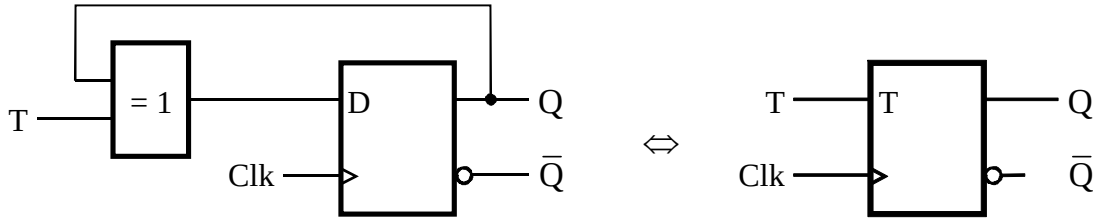


Bild 3-26: Toggle-Flipflop

Ein D-Flipflop mit XOR-Gatter, bei dem ein Eingang des XOR-Gatter am Ausgang des Flipflops angeschlossen ist, ergibt ein so genanntes Toggle-Flipflop (Bild 3-26).

Dieser Flipflop-Typ wechselt also mit jedem Takt den Zustand solange der T-Eingang auf 1 liegt. Es ergab sich klassisch aus dem JK-Flipflop (J = K = T) und wurde gerne für Zähler eingesetzt.

Verwendet man diese T-Flipflops und ergänzt den Zähler noch um einen Eingabeeingang E_n dann ergibt sich die Schaltung nach Bild 3-27.

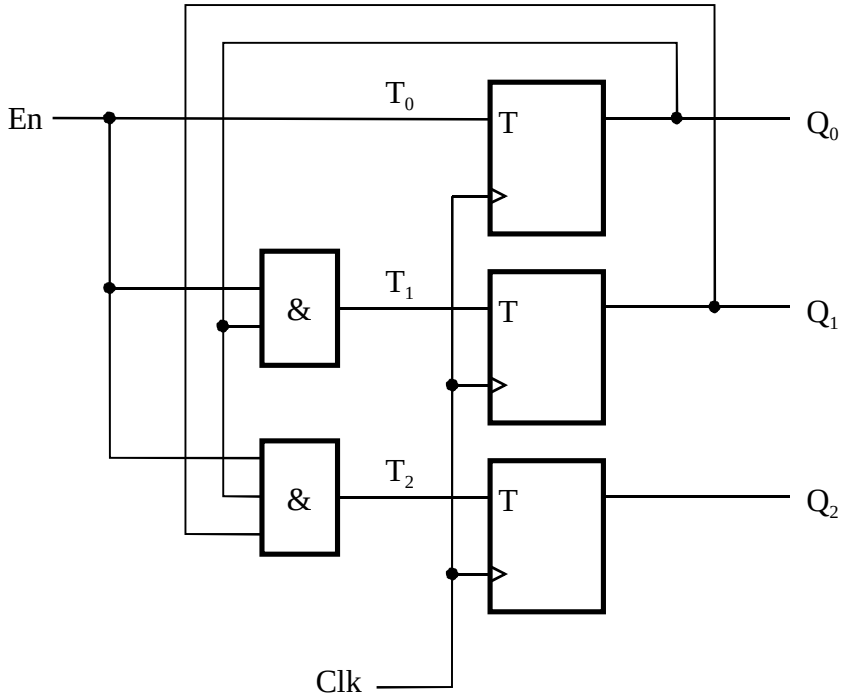


Bild 3-27: Synchroner binärer 3-bit-Zähler mit Freigabe-Eingang En

Es wird deutlich, dass für jede weitere Flipflop-Stufe das UND-Gatter einen Eingang mehr benötigt. Um dieses zu vermeiden kann man das Prinzip der Kaskadierung anwenden (Bild 3-28).

Dabei wird Gatter a in die Kaskade der Gatter a" und a' aufgespalten (A = A"). Diese Aufspaltung ist sinnvoll, wenn auch der Ausgang A' gebraucht wird. Es entsteht ein vermaschtes Netz.

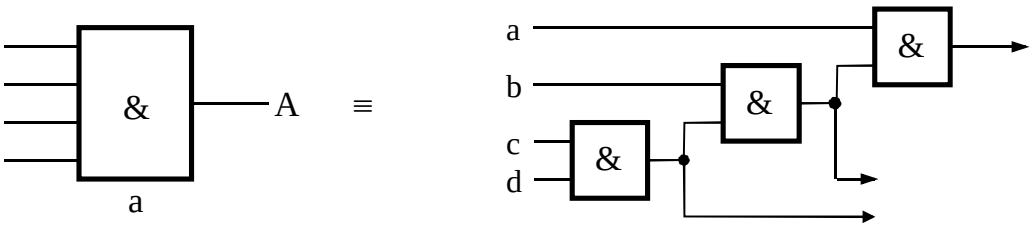


Bild 3-28: **Prinzip der Kaskadierung**

Wendet man dieses Prinzip auf den Modulo 8-Zähler an, ergibt sich mit der Tabelle in Bild 3-29 die in Bild 3-30 angegebene kaskadierte Realisierung.

mit Wechsel-FF:	Kaskade:
$T_0 = E_n$	$T_0 = E_n$
$T_1 = Q_0 \wedge E_n$	$T_1 = Q_0\ T_0\ E_n$
$T_2 = Q_1 \wedge Q_0 \wedge E_n$	$T_2 = Q_1\ T_1$

Bild 3-29: **Kaskadierung der Toggle-Flipflop-Ansteuerung für einen Modulo 8-Zähler**

Der Nachteil der Kaskadierung besteht darin, dass die maximale Verzögerungszeit des Schaltnetzes bei zunehmender Zählergröße steigt. Dadurch ergibt sich eine einzuhaltende obere Grenze für die Taktfrequenz. Daraus folgt, dass aufgrund der Kaskadierung das Gesamtsystem langsamer ist, aber Gatter mit weniger Eingängen benötigt. Dies kann bei er Realisierung von Vorteil sein.

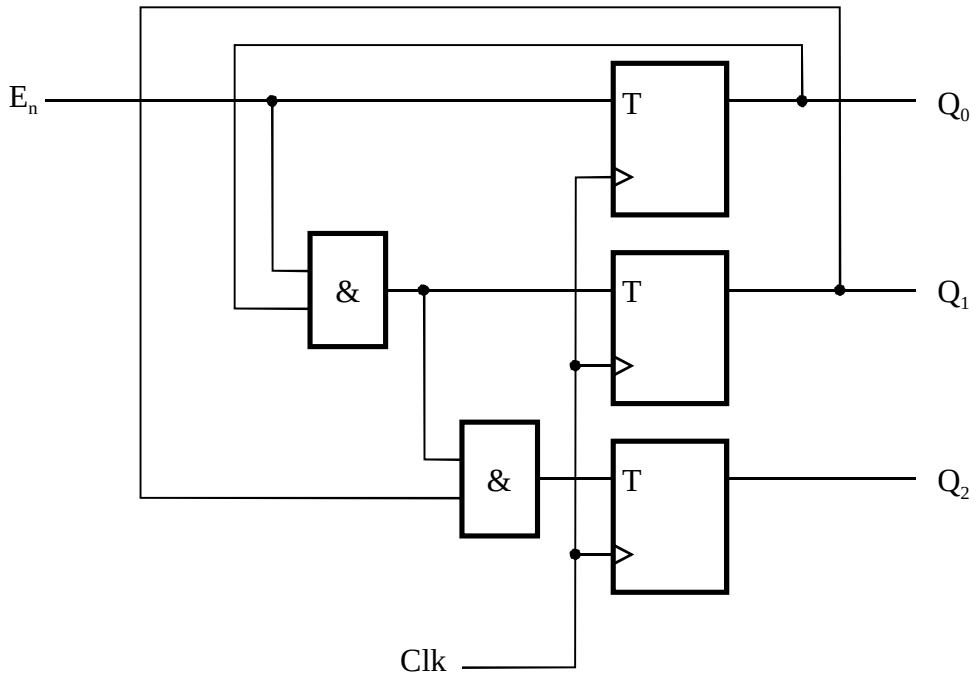


Bild 3-30: Realisierung eines 3-Bit-Zählers mit Toggle-Flipflops, Eingabeeingang E_n und kaskadierter Zähllogik

3.5 Beispielimplementierungen: Rechenwerk und Steuerwerk

Als Beispiel für ein einfaches Rechenwerk wird ein einfaches Multiplizierwerk für positive ganze Zahlen angegeben. Am dazugehörigen Steuerwerk soll der Schaltungsentwurf verdeutlicht werden.

3.5.1 Binäres Multiplizierwerk

Bild 3-31 zeigt die binäre Multiplikation durch explizite Berechnung anhand eines Beispiels. Die beiden Dualzahlen können miteinander multipliziert werden, indem die Teilprodukte P_i der einzelnen Dualzahlen des Multiplikators MQ stellenrichtig addiert werden. Die Teilprodukte sind bei der Binärmultiplikation sehr einfach, nämlich für $q_i = 0$, $P_i = 0$ und für $q_i = l$, $P_i = 2^i$ x MD (d. h. MD wird um i Stellen nach links verschoben).

Prinzipiell ist es möglich, im Rechner wie bei der Handrechnung alle Teilprodukte P_i nacheinander zu errechnen, zu speichern und am Schluss stellenrichtig zu addieren.

11 x 13 = 143 dezimal
MD MQ

q_0

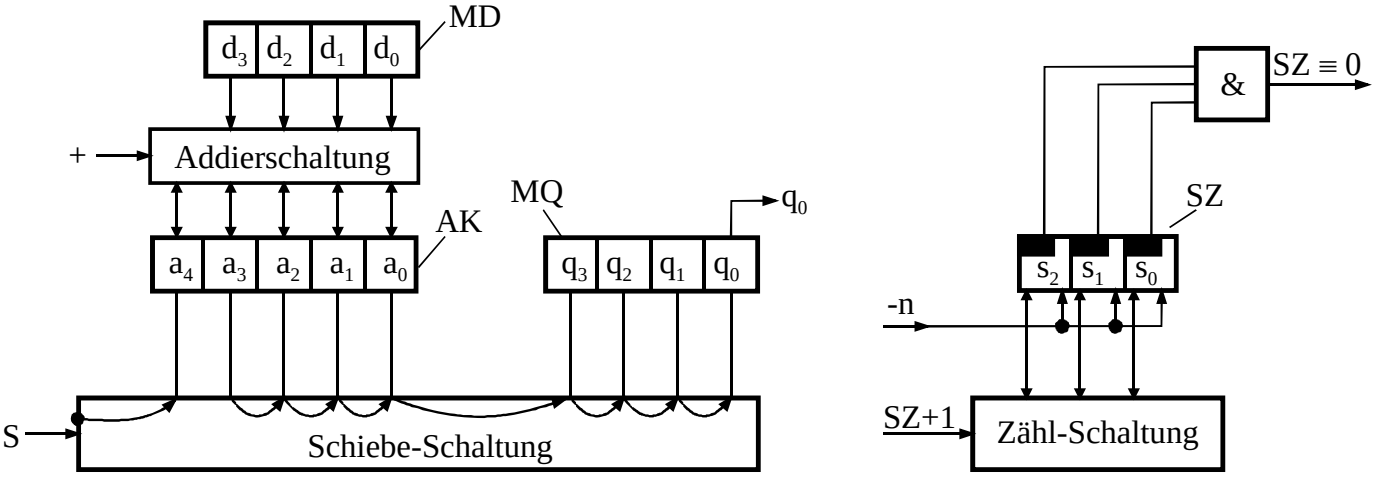
1011 x 1101	binär
1011	$P_0 = MD \cdot q_0 \cdot 2^0$
0000	$P_1 = MD \cdot q_1 \cdot 2^1$
1011	$P_2 = MD \cdot q_2 \cdot 2^2$
1011	$P_3 = MD \cdot q_3 \cdot 2^3$
10001111	$P = P_0 + P_1 + P_2 + P_3$

Bild 3-31: **Beispiel für eine binäre Multiplikation**

Dieses Verfahren ist jedoch für eine Maschine ungünstig, da in speziellen, sehr schnellen und aufwendigen Multiplizierwerken alle Teilprodukte gleichzeitig parallel addiert werden müssen.
Bild 3-34 zeigt ein einfaches Multiplizierwerk für vorzeichenlose (d.h. positive) Festkommazahlen und Bild 3-35 gibt das dazugehörige Verfahren an.

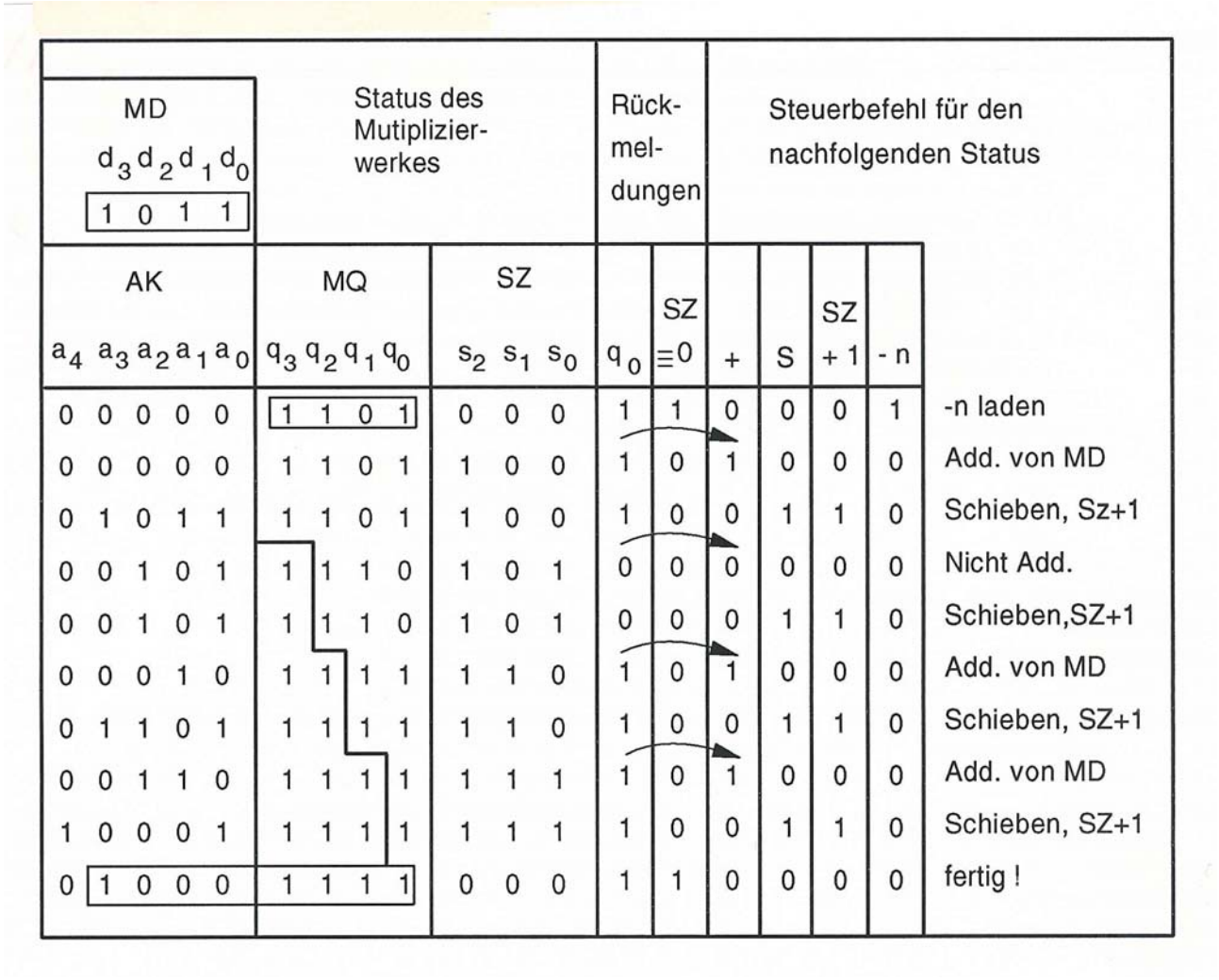
Verfahren:
MD wird mit dem Multiplikanden und MQ mit dem Multiplikator geladen, AK wird gelöscht und SZ wird mit -n geladen. MD wird zum Akkumulatorinhalt AK aufaddiert, wenn die niederwertigste Stelle in MQ (q_0) gleich 1 ist. Dann werden die Register AK und MQ um 1 Stelle nach rechts verschoben, a_0 kommt in die höchste Stelle von MQ. Die niedrigste Stelle von MQ geht verloren. Der Inhalt von MD ändert sich nicht! Dieser Vorgang wird wiederholt, bis alle Stellen abgearbeitet sind (Zählschaltung, siehe Bild 3-32). Das Ergebnis steht dann in AK und MQ.

Werden die einzelnen Operationen parallel Zeile für Zeile durchgeführt, sind 2n Takte notwendig.



MD: Multiplikanden-Register
MQ: Multiplikator-(Quotienten)-Register
n: Zahl der Stellen in MD bzw. MQ
AK: Akkumulator-Register
SZ: Stellenzähler

Bild 3-32: Multiplizierwerk für vorzeichenlose Festkommazahlen



MD $d_3 d_2 d_1 d_0$	Status des Mutiplizier-werkes			Rück-mel-dungen		Steuerbefehl für den nachfolgenden Status				
	AK	MQ	SZ							
	$a_4 a_3 a_2 a_1 a_0$	$q_3 q_2 q_1 q_0$	$s_2 s_1 s_0$	q_0	SZ ≡0	+	S	SZ + 1	- n	
1 0 1 1	0 0 0 0 0	1 1 0 1	0 0 0	1	1	0	0	0	1	-n laden
	0 0 0 0 0	1 1 0 1	1 0 0	1	0	1	0	0	0	Add. von MD
	0 1 0 1 1	1 1 0 1	1 0 0	1	0	0	1	1	0	Schieben, Sz+1
	0 0 1 0 1	1 1 1 0	1 0 1	0	0	0	0	0	0	Nicht Add.
	0 0 1 0 1	1 1 1 0	1 0 1	0	0	0	1	1	0	Schieben,SZ+1
	0 0 0 1 0	1 1 1 1	1 1 0	1	0	1	0	0	0	Add. von MD
	0 1 1 0 1	1 1 1 1	1 1 0	1	0	0	1	1	0	Schieben, SZ+1
	0 0 1 1 0	1 1 1 1	1 1 1	1	0	1	0	0	0	Add. von MD
	1 0 0 0 1	1 1 1 1	1 1 1	1	0	0	1	1	0	Schieben, SZ+1
	0 1 0 0 0	1 1 1 1	0 0 0	1	1	0	0	0	0	fertig !

Bild 3-33: Funktionsbeschreibung der binären Multiplikation

Die Multiplikation kann durch die folgenden Maßnahmen noch beschleunigt werden:

a) Shift um mehrere Stellen, wenn mehrere Stellen = 0 im MQ-Register nacheinander stehen (Shifting = Over-Zeros)

b) Multiplikation mit Übertragregister und Carry-Save-Addierern (CSA)

c) Einsatz von mehreren kaskadischen Addierern (Array-Multiplizierer)

3.5.2 Steuerwerk (Multiplikation)

In diesem Abschnitt sollen die Grundprinzipien der Steuerung von digitalen Prozessen mit synchronen Schaltwerken bzw. "Mikrosteuerwerken" erläutert werden.

Codierung der Steuerkommandos

Ein Steuerwerk versorgt das zu steuernde Werk mit einer Sequenz von Kommandos (Aktionen, Y, Ausgangsfunktionen). Im Beispiel des Multiplizierwerkes (Bild 3-34) sind dies die Menge der Mikrooperationen (einschließlich Leerbefehl, "0")

μ = [+, S, SZ+1, -n, f, 0]

+:	Addieren
S:	Schieben
SZ+l :	Stellenzähler inkrementieren
-n:	-n in Stellenzähler laden
f:	fertig
0:	"NOP"= No Operation

Definitionen:

Mikrooperation = verdrahtete Elementaroperation, die i.a. in einem elementaren Taktintervall ausgelöst werden kann. Jeder Mikrooperation ist ein elementares, auslösendes Steuersignal zugeordnet. Elementaroperationen sind unteilbare Operationen in der Hardware.

Mikrobefehl = Menge der in einer Taktzeit aktivierten simultan ablaufenden Mikrooperationen. Ein Mikrobefehl benötigt also ebenfalls i.a. ein elementares Taktintervall zur Ausführung.

Mikroprogramm = Folge von nacheinander auszuführenden Mikrobefehlen

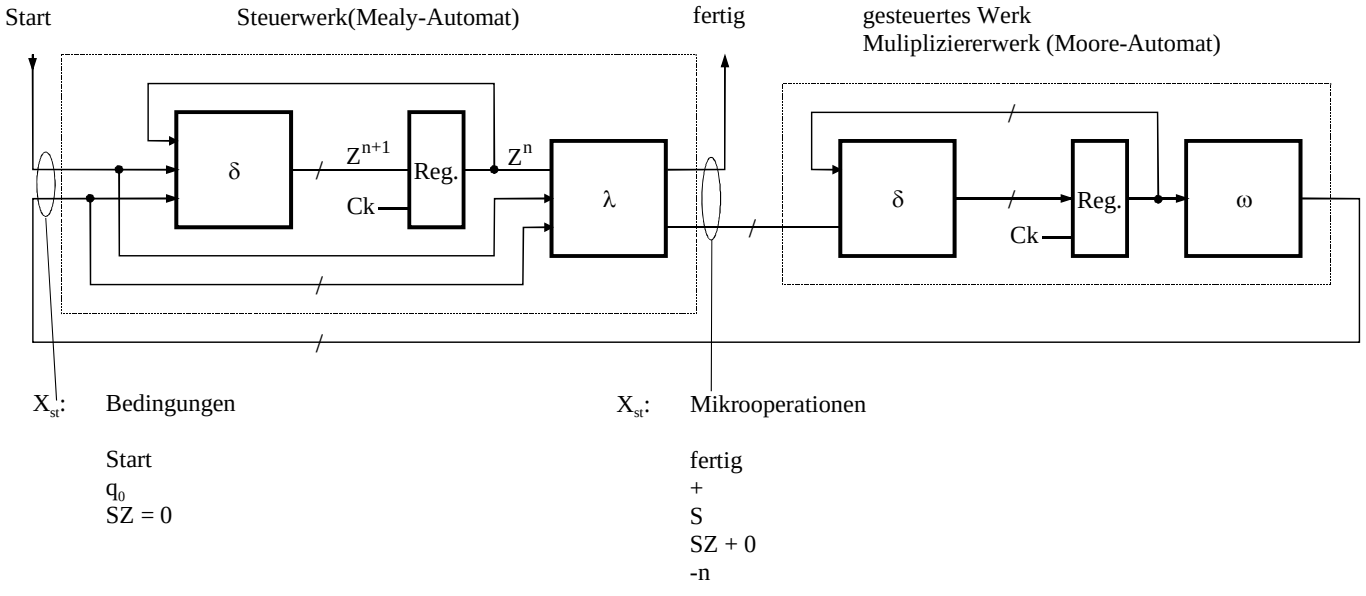


Bild 3-34: **Beispiel Multiplizierautomat**

Folge der Steuerkommandos

Die geforderte Folge der Steuerkommandos kann man als Zustandsgraph darstellen. Die "Bedingungen" (X) werden z.T. von einem übergeordneten Steuerprozess (z.B. "Start", Bild 3-34) gegeben und zum anderen Teil aus den Rückmeldungen des gesteuerten Werkes nach Ausführung der letzten Aktion (z.B. "q_0","SZ=0") abgeleitet werden. In unserem Beispiel besteht der Bedingungsvektor aus den Komponenten

$$X_{st} = (q_0, SZ=0, Start)$$

Ein im Aufwand reduzierter Ansatz ergibt sich in der Darstellung als Zustandsgraph. Bild 3-35a zeigt den Zustandsgraph und Bild 3-35b die Realisierung als PLA.

Tabelle 3-6 zeigt die Wahrheitstabelle des Steuerwerkes und den dazugehörigen Zustandsgraphen.

Tabelle 3-6 Wahrheitstabelle des Steuerwerkes

Start	q_0	SZ = 0	Z_1^n	Z_0^n	Z_1^{n+1}	Z_0^{n+1}	S	SZ+1	+	-n	f
0	X	X	0	0	0	0	0	0	0	0	1
1	X	X	0	0	0	1	0	0	0	1	0
X	0	X	0	1	1	0	0	0	0	0	0
X	1	X	0	1	1	0	0	0	1	0	0
X	X	X	1	0	1	1	1	1	0	0	0
X	X	0	1	1	0	1	0	0	0	0	0
X	X	1	1	1	0	0	0	0	0	0	0

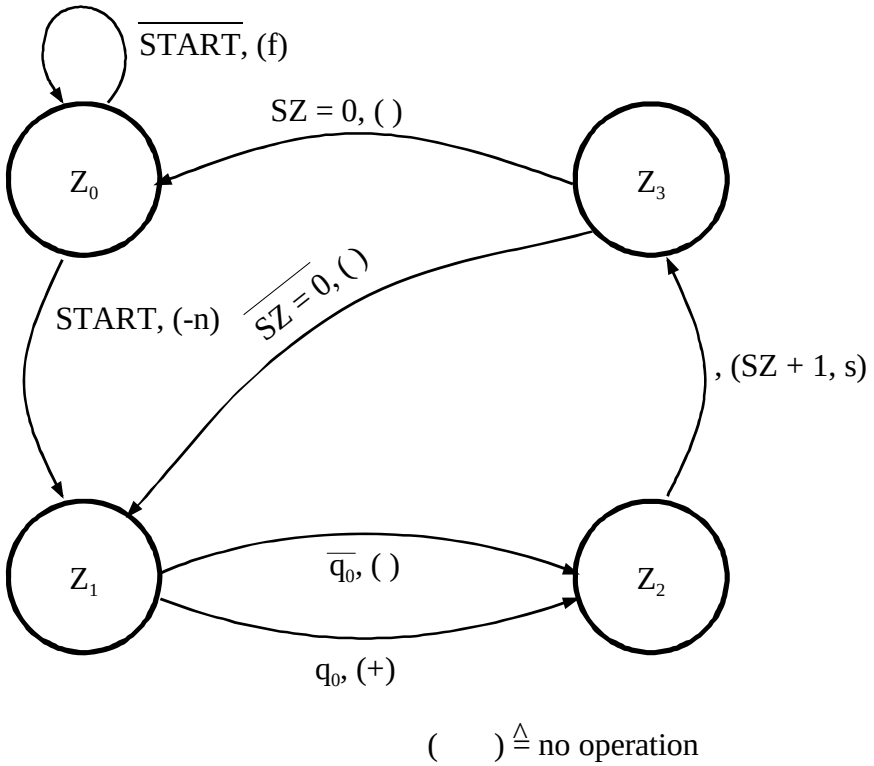


Bild 3-35: Zustandsgraph des optimierten Steuerwerks

Bild 3-35 gibt eine nicht optimierte Realisierung für PLA an.

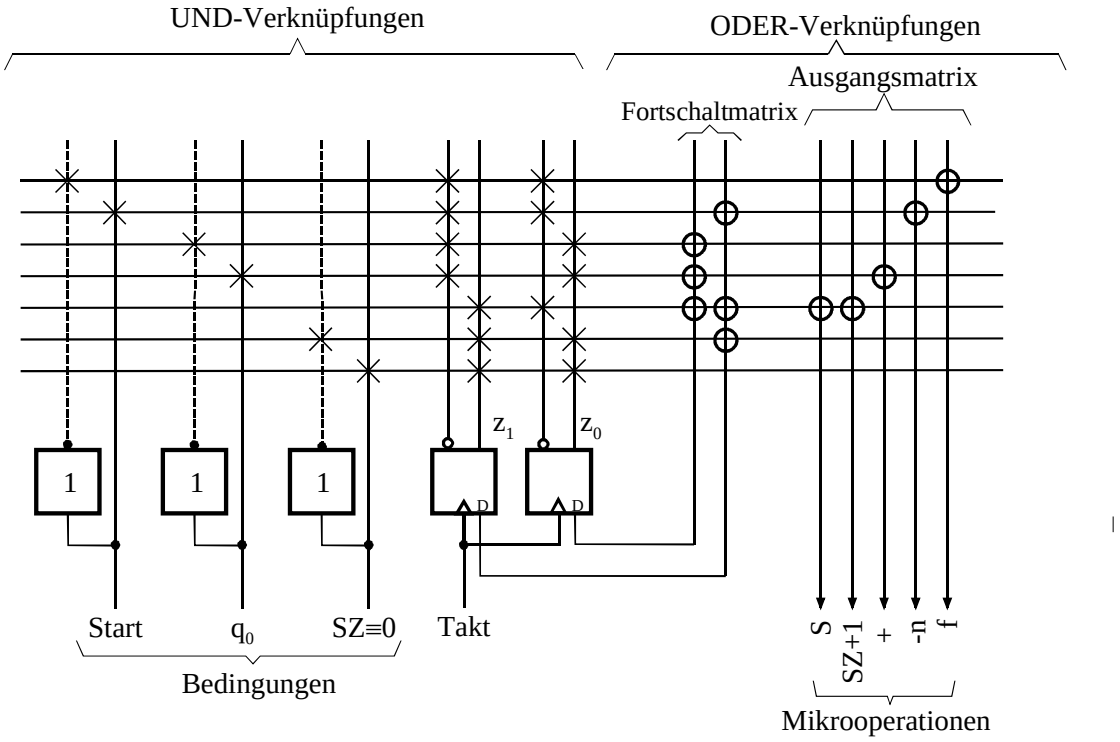


Bild 3-36: **Steuerwerk für die Multiplikation**

3.6 Implementierungsaspekte

Synchrone Schaltwerke sind, wie im vorigen Abschnitt gezeigt, rückgekoppelte Schaltnetze, wobei Register in der Rückkopplung eine geordnete Zustandsfortschaltung gewährleisten.

Dabei muss auf eine vollständig synchrone Implementierung geachtet werden. Ein taktgesteuertes oder 2-Taktpegel-gesteuertes Zustandsregister sorgt für eine stabile Zustandsfortschaltung, bei der der jeweilige Zustand mindestens eine Taktperiode gültig ist.

Zusätzlich **müssen** auch alle Eingangssignale auf den Takt synchronisiert werden die nicht schon synchron aus einem Schaltwerk kommen, dass auf dem selben Takt läuft. Zur Taktsynchronisation können z. B. D-Flipflops verwendet werden. Werden die Ausgangssignale eines Schaltwerkes wiederum als Taktsignale verwendet, so sind für Automaten vom Typ Moore und Mealy zusätzlich Ausgangsregister vorzusehen. Beim Medwedjew-Automaten ist dies nicht notwendig, da alle Ausgangssignale direkt aus Flipflops kommen, was für bestimmte Anwendungen von Vorteil sein kann. In allen Fällen sind die Laufzeiten der Signale durch die Schaltnetze in Hinblick auf die verwendete Taktfrequenz zu beachten.

Das System soll getaktet sein, d. h. e muss eine "leere Eingabe" KE (keine Eingabe) und eine "leere Ausgabe" KA (keine Ausgabe) geben. Der Mealy-Automat A = (X, Y, Z, d, g) hat folgende Ein-/Ausgabe und Zustandsvektoren:

X = { 0.50, 1.-, KE, Rück }

Y = { Cola, 0.50, 1.-, KA }

Z = { KS, 0.50, 1.- }

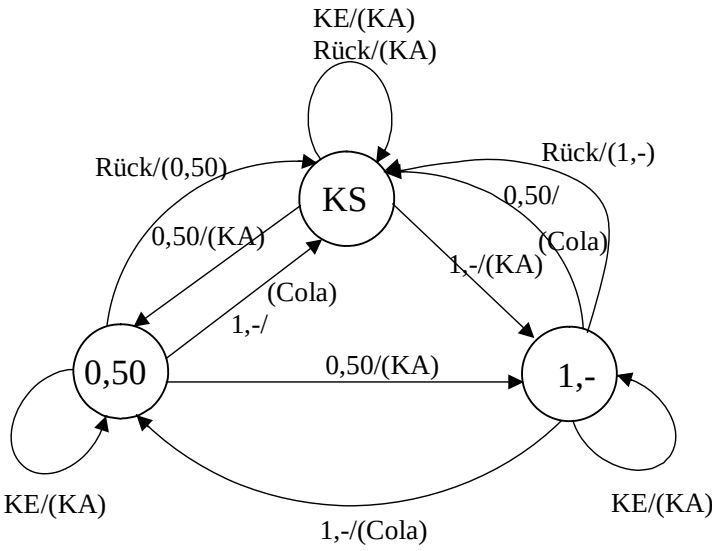


Bild 3-37: Zustandsgraph des Cola-Automaten

In Bild 3-37 ist der Zustands- oder Automatengraph dargestellt. Es werden drei Zustände zur Realisierung benötigt:

KS: Geldspeicher leer

0,50: Geldspeicher enthält 0,50 €

1,-: Geldspeicher enthält 1,00 €

es sind die Schaltnetze δ für die Zustandsfortschaltung λ für die Ausgangstransformation zu entwerfen. Um die Zahl der Ein- und Ausgänge für diesen Schaltnetzentwurf klein zu halten, werden die Eingänge X, der Zustandsvektor und die Ausgänge Y codiert:

X	KE	0.50	1.-	Rück
x_1x_0	00	01	10	11

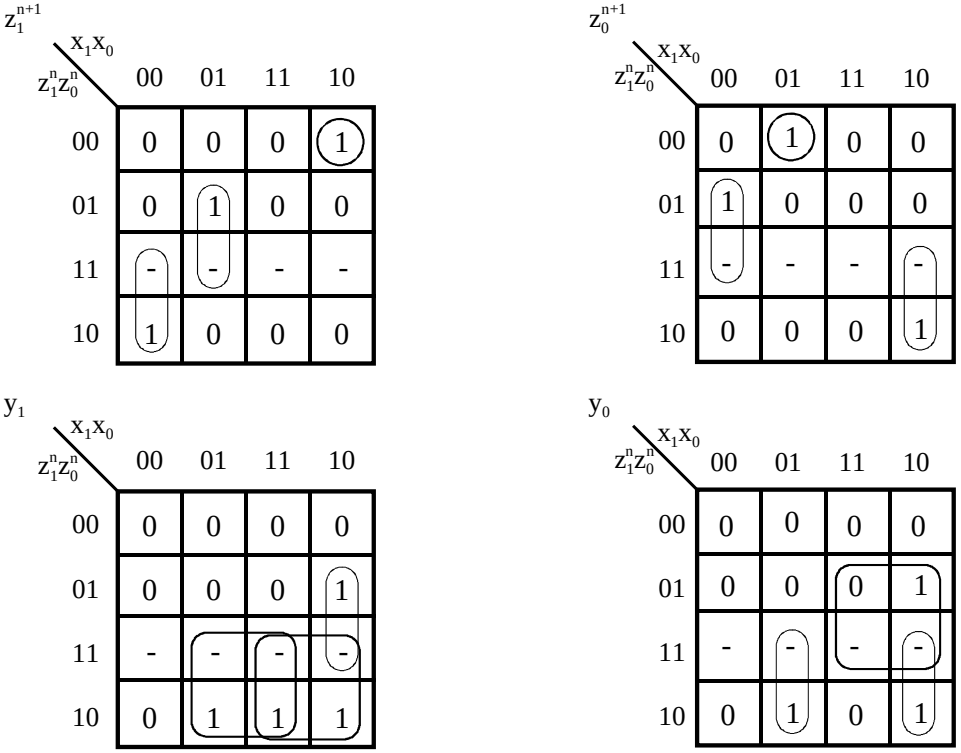
Y	KA	0.50	1.-	Cola
y_1y_0	00	01	10	11

Z	KS	0.50	1.-
z_1z_0	00	01	10

Tabelle 3-7: Funktionstabelle für λ und δ

Eingänge δ, λ				Ausgänge δ		Ausgänge λ	
x_1	x_0	z_1^n	z_0^n	z_1^{n+1}	z_0^{n+1}	y_1	y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0
0	0	1	0	1	0	0	0
0	0	1	1	-	-	-	-
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	0	0	1	1
0	1	1	1	-	-	-	-
1	0	0	0	1	0	0	0
1	0	0	1	0	0	1	1
1	0	1	0	0	1	1	1
1	0	1	1	-	-	-	-
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	1
1	1	1	0	0	0	1	0
1	1	1	1	-	-	-	-

Die Minimierung über KN-Diagramme ergibt:



$$z_1^{n+1} = x_1\bar{x}_0\bar{z}_1^n\bar{z}_0^n \vee \bar{x}_1x_0z_0^n \vee \bar{x}_1x_0z_1^n$$
$$z_0^{n+1} = \bar{x}_1x_0\bar{z}_1^n\bar{z}_0^n \vee \bar{x}_1\bar{x}_0z_0^n \vee x_1\bar{x}_0z_1^n$$
$$y_1 = x_1z_1^n \vee x_0z_1^n \vee x_1\bar{x}_0z_0^n$$
$$y_0 = x_1z_0^n \vee x_1\bar{x}_0z_1^n \vee \bar{x}_1x_0z_1^n$$

Bild 3-38 zeigt die Realisierung eines FPLA:

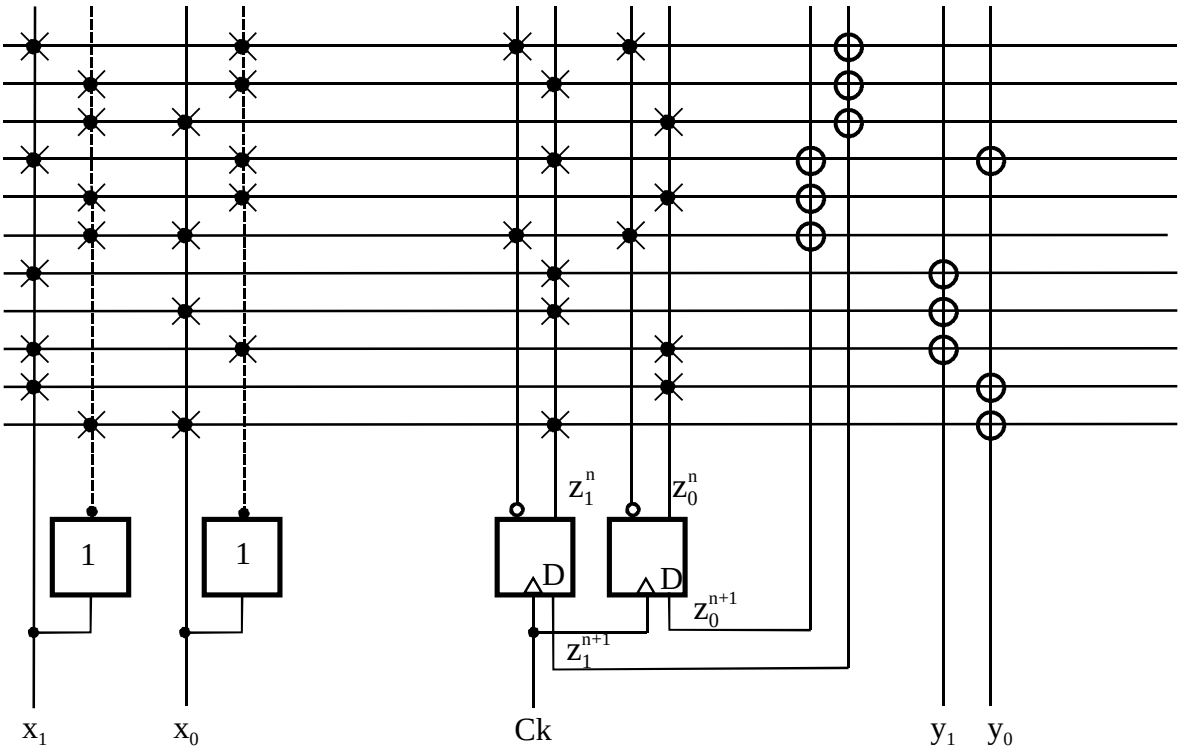


Bild 3-38: FPLA Realisierung