5.2. Sequentielle Schaltungen

5.2.1. Grundlagen

5.2.1.1. Begriff

Sie erinnern sich:

Kombinatorisch heißt ein System dann, wenn seine Ausgangsbelegung in einem Zeitpunkt t_n ausschließlich von der **Eingangsbelegung** im gleichen Zeitpunkt t_n bestimmt wird.

Im Gegensatz dazu:

Sequentiell heißt ein System dann, wenn seine Ausgangsbelegung in einem Zeitpunkt t_n von **der Folge der Eingangsbelegungen** in (nicht notwendig allen) zurückliegenden Zeitpunkten $...,t_{n-2},t_{n-1}$ und dem Zeitpunkt t_n bestimmt wird.

Ein sequentielles System braucht ein **"Gedächtnis"**, das sich die Eingangsbelegungen in den zurückliegenden Zeitpunkten merkt.

5.2.1.2. Der endliche deterministische Automat

Nichts ist so praktisch wie eine gute Theorie!

Das Automatenmodell wird in der Informatik breit angewendet. Sie werden es in den verschiedensten Zusammenhängen wiederfinden, auch in abgewandelter Form. Die Form, die ich Ihnen hier vorstelle, ist die Grundlage für die einheitliche Beschreibung aller sequentiellen Systeme und folglich von immenser Bedeutung.

Def.: Ein Automat A ist ein Quintupel

$$A = (X,Y,Z,f,g)$$

mit

X = Menge der Eingabesymbole (= Eingabealphabet)
Y = Menge der Ausgabesymbole (= Ausgabealphabet)
Z = Menge der Zustände (= Zustandsalphabet)
f = Überführungsfunktion
g = Ergebnisfunktion (Ausgabefunktion, Resultatfunktion)

Da die **Menge der Zustände** (= Gedächtnis des Automaten) **endlich** ist, heißt der Automat **endlicher** Automat.

mit

$f: X \times Z \longrightarrow Z$
$g: X \times Z \longrightarrow Y$

f und g sind Mengenrelationen oder Abbildungen. Jeder Kombination aus einem Symbol des Eingabealphabets und einem Zustand ("alter" Zustand) aus dem Zustandsalphabet wird durch f genau ein Zustand ("neuer" Zustand) aus dem Zustandsalphabet und durch g genau ein Symbol aus dem Ausgabealphabet zugeordnet. Man kann f und g als Mengen geordneter Tripel (Eingabesymbol,alter_Zustand,neuer_Zustand) bzw. (Eingabesymbol,alter_Zustand,Ausgabesymbol) beschreiben.

Da immer **genau ein Folgezustand** und genau ein Ausgabesymbol zugeordnet werden, heißt der Automat **deterministischer** Automat.

Beispiel 5.14.

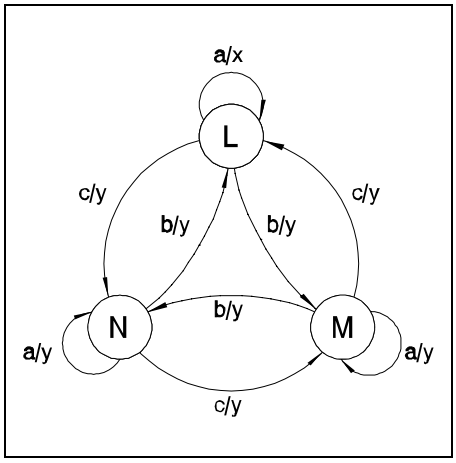
```
X = {a,b,c}
Y = {x,y}
Z = {L,M,N}

f = {(a,L,L),(b,L,M),(c,L,N),
     (a,M,M),(b,M,N),(c,M,L),
     (a,N,N),(b,N,L),(c,N,M)}

g = {(a,L,x),(b,L,y),(c,L,y),
     (a,M,y),(b,M,y),(c,M,y),
     (a,N,y),(b,N,y),(c,N,y)}
```

Diese Darstellung ist zwar korrekt aber wenig übersichtlich. Man stellt die Verhältnisse entweder in einem **Automatengraphen** oder in einer **Automatentabelle** dar:

Automatengraph



Automatentabelle

Ergebnisfunktion g

Überführungsfunktion f

x	z	z'	y
a	L	L	x
b	L	M	y
c	L	N	y
a	M	M	y
b	M	N	y
c	M	L	y
a	N	N	y
b	N	L	y
c	N	M	y

Der Automatengraph ist gut geeignet, einen Überblick zu geben. Für Berechnungen aller Art eignet sich besser die Automatenta-

belle. Wir werden beide Darstellungsformen nebeneinander verwenden.

Für den Fall des binären Automaten sind die Eingabesymbole, die Ausgabesymbole und die Zustände **binär codiert.**

Ein **Eingabesymbol** ist eine **Belegung der Eingänge**, ein **Ausgabesymbol eine Belegung der Ausgänge** und ein **Zustand eine Belegung des Zustandsspeichers**.

Man sagt dann besser:

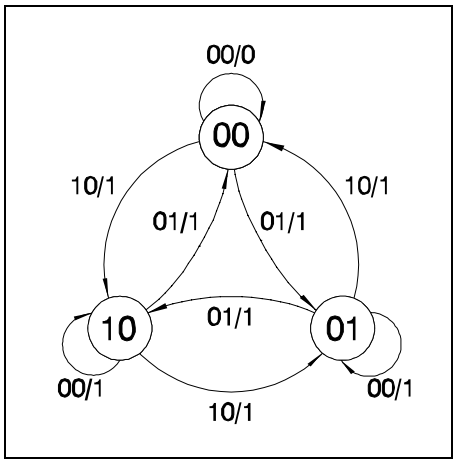
X = Menge der Eingangsbelegungen
Y = Menge der Ausgangsbelegungen
Z = Menge der Zustände
f = Überführungsfunktion
g = Ergebnisfunktion (Ausgabefunktion, Resultatfunktion)

Mit

a = 00, b = 01, c = 10, L = 00, M = 01, N = 10, x = 0, y = 1

geht unser Beispiel über in

Automatengraph



Automatentabelle

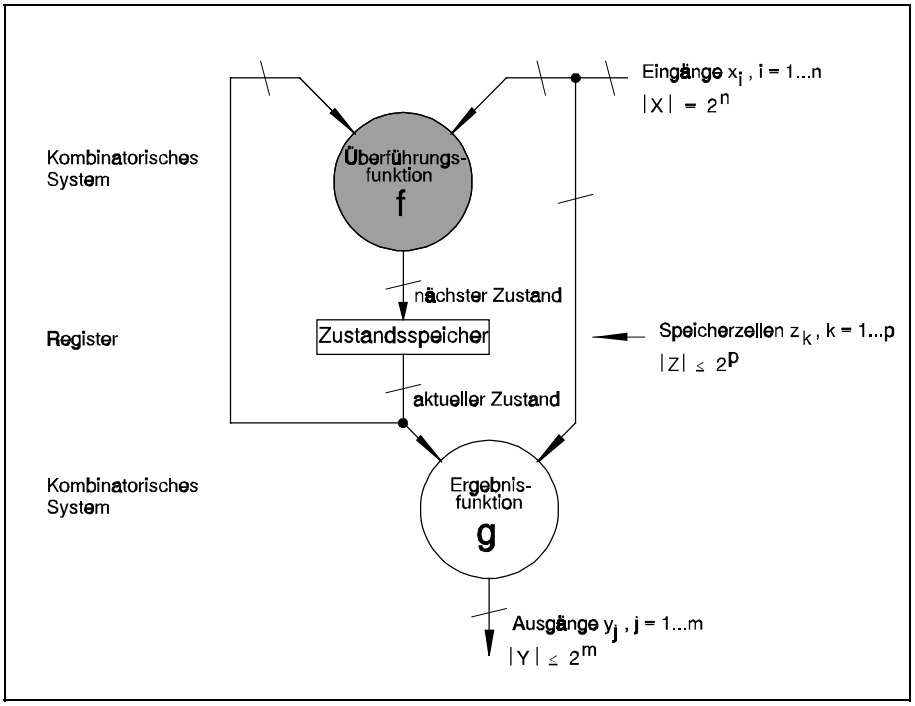
Ergebnisfunktion g

Überführungsfunktion f

x z	z'	y
00 00	00	0
01 00	01	1
10 00	10	1
00 01	01	1
01 01	10	1
10 01	00	1
00 10	10	1
01 10	00	1
10 10	01	1

Sie ahnen wohl sicher schon, daß uns die Tatsache, daß wir die Kodierungen x = 11 und z = 11 nicht verwenden, noch beschäftigen wird.

Eine weitere Darstellungsmöglichkeit, die uns den Weg für die schaltungstechnische Realisierung öffnet, ist eine Art Ersatzschaltung des Automaten:



Das Automatenmodell, welches wir bisher verwendet haben, ist ein sogenannter MEALY-Automat. Außerdem kennen wir noch dem MOORE- und den MEDVEDEV-Automaten. Sie unterscheiden sich nur in der Ausgabefunktion:

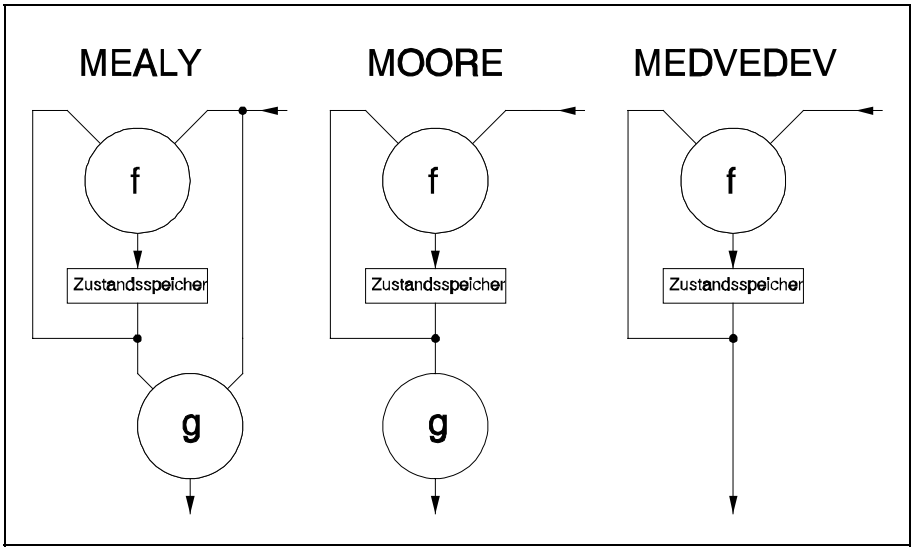
MEALY g: X x Z ⟶ Y

MOORE g: Z ⟶ Y

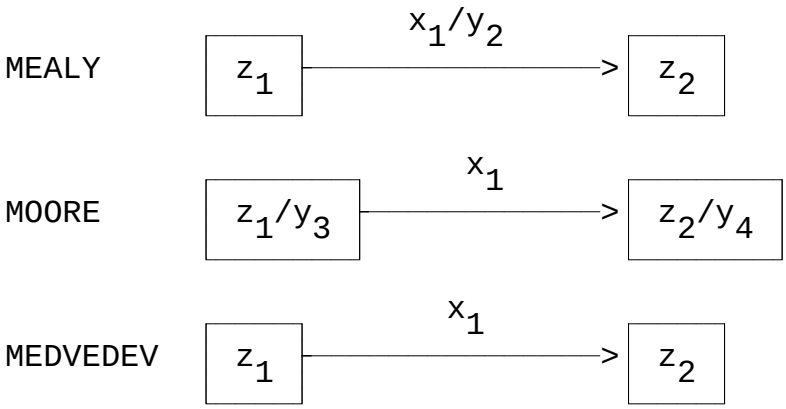
MEDVEDEV g: Z = Y

mit $z_i = y_i$ für alle i, d. h. die Speicherzellen sind direkt herausgeführt!

In der Ersatzschaltung zeigt sich das so:



Die Zustands- und Kantenbeschriftung im Automatengraphen ist für die drei Automatentypen unterschiedlich:



In der Automatentabelle fehlt beim MEDVEDEV-Automaten die Spalte für die Ausgangsbelegungen (y).

Beispiel 5.15.

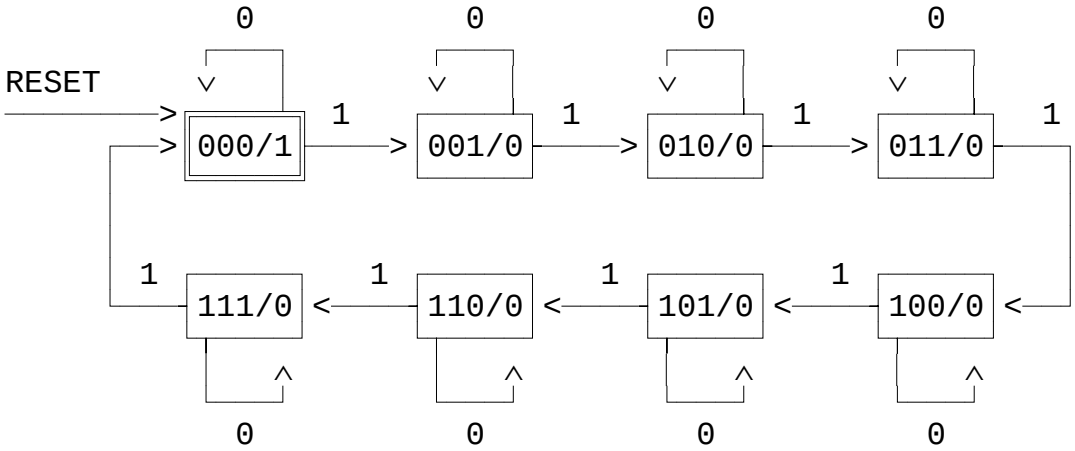
Ein 3-Bit-Binärzähler soll bei x = 1 in Schritten von 1 vorwärtszählen und bei x = 0 nicht zählen. Bei z = 000 soll er y = 1 ausgeben, sonst y = 0. Der Zustand z = 000 sei der Initialzustand (Anfangszustand), in den der Zähler auf geeignete, hier nicht näher betrachtete Weise voreinstellbar (rücksetzbar) ist.

Aus der Aufgabenstellung leiten wir ab

Z = {000,001,010,011,100,101,110,111}
X = {0,1}
Y = {0,1}

Da die Ausgabe unabhängig von der Eingabe ist, liegt ein MOORE-Automat vor.

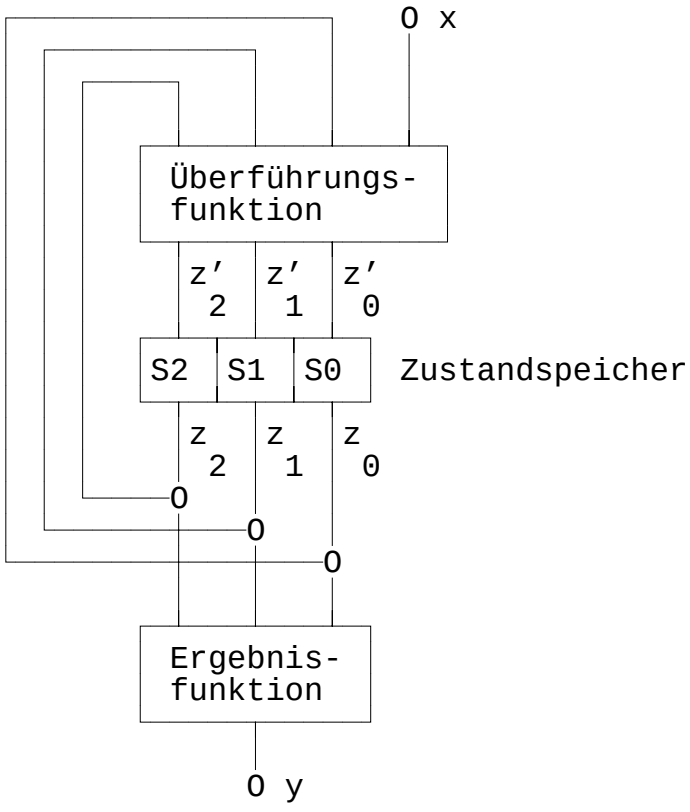
Automatengraph



Automatentabelle

x	$z_2 z_1 z_0$	$z'_2 z'_1 z'_0$	y
0	0 0 0	0 0 0	1
0	0 0 1	0 0 1	0
0	0 1 0	0 1 0	0
0	0 1 1	0 1 1	0
0	1 0 0	1 0 0	0
0	1 0 1	1 0 1	0
0	1 1 0	1 1 0	0
0	1 1 1	1 1 1	0
1	0 0 0	0 0 1	1
1	0 0 1	0 1 0	0
1	0 1 0	0 1 1	0
1	0 1 1	1 0 0	0
1	1 0 0	1 0 1	0
1	1 0 1	1 1 0	0
1	1 1 0	1 1 1	0
1	1 1 1	0 0 0	0

Ersatzschaltung



Vorgriff auf den Automatenentwurf:

Es war bereits betont worden, daß Überführungsfunktion und Ergebnisfunktion durch kombinatorische Systeme zu realisieren sind. Es sollte also möglich sein, die BOOLEschen Funktionen $z_2' = f(x,z_2,z_1,z_0)$, $z_1' = f(x,z_2,z_1,z_0)$, $z_0' = f(x,z_2,z_1,z_0)$ und $y = f(x,z_2,z_1,z_0)$ aus der Automatentabelle abzulesen und zu minimieren und damit entsprechende Gatternetze zu erhalten.

z_2'	0011	z_1
	0110	z_0
00	0000	
01	1111	
11	1101	
10	0010	
xz_2		

$\mathbf{z_2'} = \bar{x}z_2 \vee z_2\bar{z}_1 \vee z_2\bar{z}_0 \vee x\bar{z}_2z_1z_0 = \mathbf{z_2 \not\vee xz_1z_0}$

z_1'	0011	z_1
	0110	z_0
00	0011	
01	0011	
11	0101	
10	0101	
xz_2		

$\mathbf{z_1'} = z_1\bar{z}_0 \vee \bar{x}z_1 \vee x\bar{z}_1z_0 = \mathbf{z_1 \not\vee xz_0}$

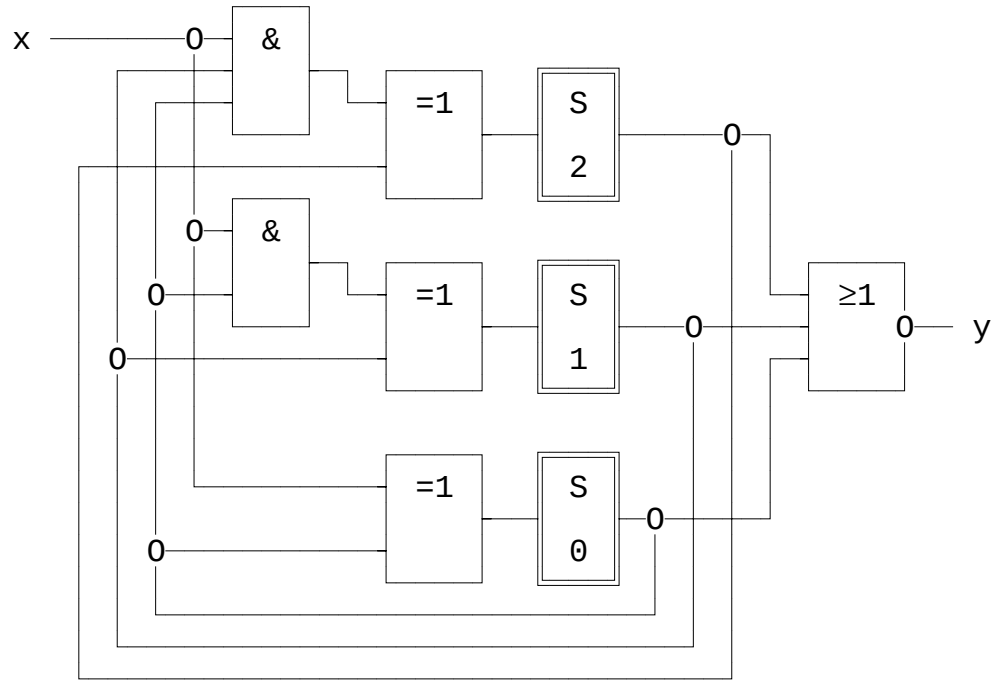
z_0'	0011	z_1
	0110	z_0
00	0110	
01	0110	
11	1001	
10	1001	
xz_2		

$\mathbf{z_0'} = \bar{x}z_0 \vee x\bar{z}_0 = \mathbf{z_0 \not\vee x}$

y	0011	z_1
	0110	z_0
00	1000	
01	0000	
11	0000	
10	1000	
xz_2		

$\mathbf{y} = \bar{z}_2\bar{z}_1\bar{z}_0 = \overline{\mathbf{z_2 \vee z_1 \vee z_0}}$

Antivalenz und NOR bieten sich hier an und führen zu einer sehr übersichtlichen Realisierung. Die resultierende Schaltung sähe dann so aus:



Das könnte tatsächlich bereits ein funktionstüchtiger Automat sein!

Wir haben eine Vorstellung vom Entwurf der Gatternetze für die beiden Funktionen. Unklar ist noch die Realisierung des Zustandsspeichers. Auch auf die oben vereinbarte Rücksetzbarkeit in einen Initialzustand sind wir noch nicht eingegangen. Und schließlich ist noch unklar, was die Zustandsübergänge wann auslöst. In den nächsten Vorlesungen werden wir uns deshalb auf den Zustandsspeicher konzentrieren.

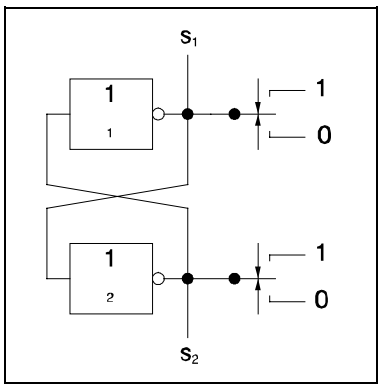
5.2.2. Zustandsspeicher

Ein Zustandsspeicher ist ein eindimensionales Array aus 1-Bit-Speicherzellen, den sog. Flipflops. Ein solches eindimensionales Array heißt auch (Speicher-)Register.

5.2.2.1. Flipflop

5.2.2.1.1. Experiment

Zum besseren Verständnis der Funktion eines Flipflops stellen wir ein Gedankenexperiment voran.



Wir betrachten zwei kreuzgekoppelte Negatoren, einen sog. Flipflop-Kern. An den Ausgängen s_1 und s_2 erkennen Sie zwei Taster. Jeder Taster hat drei Stellungen: 0, 1 und die hier gezeichnete Mittelstellung. Wenn sich ein Taster in der Stellung 0 (1) befindet, dann soll der Ausgang, an den der Taster angeschlossen ist, den Pegel 0 (1) annehmen. Befindet sich der Taster in Mittelstellung, bleibt der Ausgang sich selbst überlassen.

Die beiden Taster sollen unabhängig voneinander jede der drei Stellungen einnehmen können. Damit ist es möglich, den Flipflop-Kern beliebig zu initialisieren (s_1s_2 = 00|01|10|11), ihn anschließend sich selbst zu überlassen und zu beobachten, was er dann tut. Wir verwenden dazu ein Simulationsprogramm, das Sie im nächsten Semester noch genauer kennenlernen werden (VHDL).

In einem ersten Experiment nehmen wir an, daß der Flipflop-Kern symmetrisch ist, d. h. beide Negatoren sollen exakt die gleiche Verzögerungszeit aufweisen (hier: 10 ns).

t	s1 s2	s1 s2	s1 s2	s1 s2	
0 ns	**0 0**	**0 1**	**1 0**	**1 1**	**<— Initialbelegung**
10 ns	1 1	. .	. .	0 0	
20 ns	0 0	. .	. .	1 1	
30 ns	1 1	. .	. .	0 0	
40 ns	0 0	. .	. .	1 1	

In den Initialisierungsfällen s_1s_2 = 00 und s_1s_2 = 11 "schwingt" der Flipflop-Kern, in den Initialisierungsfällen s_1s_2 = 01 und s_1s_2 = 10 behält der Flipflop-Kern die Initialbelegungen bei, er "speichert" sie.

In einem zweiten Experiment nehmen wir an, daß der Flipflop-Kern unsymmetrisch ist, d. h. die beiden Negatoren sollen unterschiedliche Verzögerungszeiten aufweisen (Negator 1: 10 ns, Negator 2: 20 ns).

t	s1 s2	s1 s2	s1 s2	s1 s2	
0 ns	**0 0**	**0 1**	**1 0**	**1 1**	**<— Initialbelegung**
10 ns	1 0	. .	. .	0 1	
20 ns	. .	. .	. .	. .	
30 ns	. .	. .	. .	. .	
40 ns	. .	. .	. .	. .	

In den Initialisierungsfällen s_1s_2 = 00 und s_1s_2 = 11 "gewinnt" der schnellere Negator, d. h. der Negator, der als erster den neuen Ausgangspegel bereitstellt, in den Initialisierungsfällen s_1s_2 = 01 und s_1s_2 = 10 behält der Flipflop-Kern die Initialbelegungen bei, er "speichert" sie.

In einem dritten Experiment nehmen wir an, daß Negator 1 der langsamere (20 ns) und Negator 2 der schnellere (10 ns) ist.

t	s1 s2	s1 s2	s1 s2	s1 s2	
0 ns	**0 0**	**0 1**	**1 0**	**1 1**	**<— Initialbelegung**
10 ns	0 1	. .	. .	1 0	
20 ns	. .	. .	. .	. .	
30 ns	. .	. .	. .	. .	
40 ns	. .	. .	. .	. .	

Auch hier "gewinnt" wieder in den Initialisierungsfällen s_1s_2 =

00 und s_1s_2 = 11 der schnellere Negator, und auch hier "speichert" wieder der Flipflop-Kern in den Initialsierungsfällen s_1s_2 = 01 und s_1s_2 = 10 die Initialsierungsbelegungen.

Wenn man die Ergebnisse der drei Experimente anders zusammenstellt, wird die Aussage noch deutlicher:

t	t1<t2		t1=t2		t1>t2		
	s1	s2	s1	s2	s1	s2	
0 ns	**0**	**0**	**0**	**0**	**0**	**0**	**<— Initialbelegung**
10 ns	1	0	1	1	0	1	
20 ns	.	.	0	0	.	.	
30 ns	.	.	1	1	.	.	
40 ns	.	.	0	0	.	.	

t	t1<t2		t1=t2		t1>t2		
	s1	s2	s1	s2	s1	s2	
0 ns	**0**	**1**	**0**	**1**	**0**	**1**	**<— Initialbelegung**
10 ns	.	.	.	.	.	.	
20 ns	.	.	.	.	.	.	
30 ns	.	.	.	.	.	.	
40 ns	.	.	.	.	.	.	

t	t1<t2		t1=t2		t1>t2		
	s1	s2	s1	s2	s1	s2	
0 ns	**1**	**0**	**1**	**0**	**1**	**0**	**<— Initialbelegung**
10 ns	.	.	.	.	.	.	
20 ns	.	.	.	.	.	.	
30 ns	.	.	.	.	.	.	
40 ns	.	.	.	.	.	.	

t	t1<t2		t1=t2		t1>t2		
	s1	s2	s1	s2	s1	s2	
0 ns	**1**	**1**	**1**	**1**	**1**	**1**	**<— Initialbelegung**
10 ns	0	1	0	0	1	0	
20 ns	.	.	1	1	.	.	
30 ns	.	.	0	0	.	.	
40 ns	.	.	1	1	.	.	

In den Initialsierungsfällen s_1s_2 = 01 und s_1s_2 = 10 "speichert" der Flipflop-Kern die Initialbelegung unabhängig davon, ob er symmetrisch oder unsymmetrisch ist. In den Initialisierungsfällen s_1s_2 = 00 und s_1s_2 11 hängt das Ergebnis davon ab, ob der Flipflop-Kern symmetrisch ist (dann "schwingt" er) oder ob und auf welche Art er unsymmetrisch ist (dann "gewinnt" der schnellere Negator).

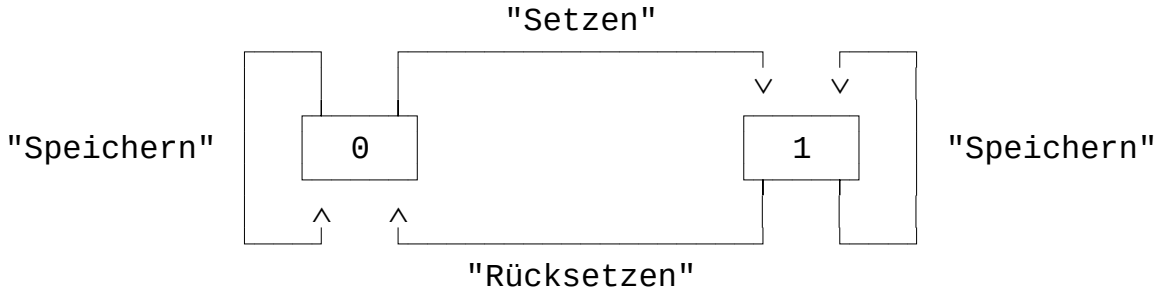
In der technischen Realität ist einerseits die exakte Symmetrie

nicht erreichbar und andererseits ist die Art der Unsymmetrie schwer vorhersagbar. Die Initialisierungsfälle $s_1s_2 = 00$ und $s_1s_2 = 11$ sind deshalb ungeeignet für den Einsatz des Flipflop-Kerns als Speicher. Die Initialisierungsfälle $s_1s_2 = 01$ und $s_1s_2 = 10$ hingegen eignen sich offenbar hervorragend.

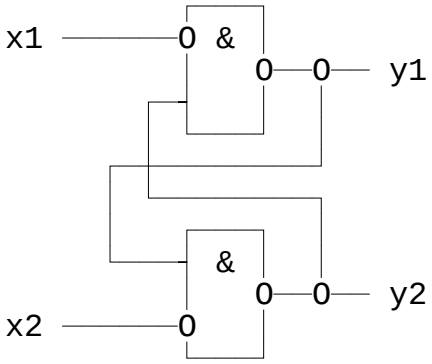
Unklar ist noch, wie die zu speichernde Information in den Flipflop-Kern hineinkommt, da ja kaum davon ausgegangen werden kann, daß die in unserem Gedankenexperiment verwendeten Taster zum Einsatz kommen. Die weiteren Überlegungen werden sich also mit der schaltungstechnischen Realisierung von Flipflops befassen müssen.

5.2.2.1.2. Grund-Flipflop

Das Grund-Flipflop (GFF) ist ein Miniautomat vom Typ MEDVEDEV. Der Automat hat zwei Zustände, Z = {0,1}.



Wir kennen zwei unterschiedliche GFFs, das NAND-Grund-Flipflop (NAND-GFF) und das NOR-Grund-Flipflop (NOR-GFF), und werden aber, obwohl dafür kein zwingender Grund besteht, nur das NAND-GFF näher betrachten und verwenden.



Was leistet das NAND-GFF? Wir analysieren das Flipflop zunächst als kombinatorisches System und ermitteln die Wahrheitstabelle.

x_1	x_2	y_1	y_2	
1	1	1	1	
1	0	1	0	
0	1	0	1	
0	0	?	?	<—— Mit $x_1x_2 = 00$ bleibt sich das Flipflop selbst überlassen!

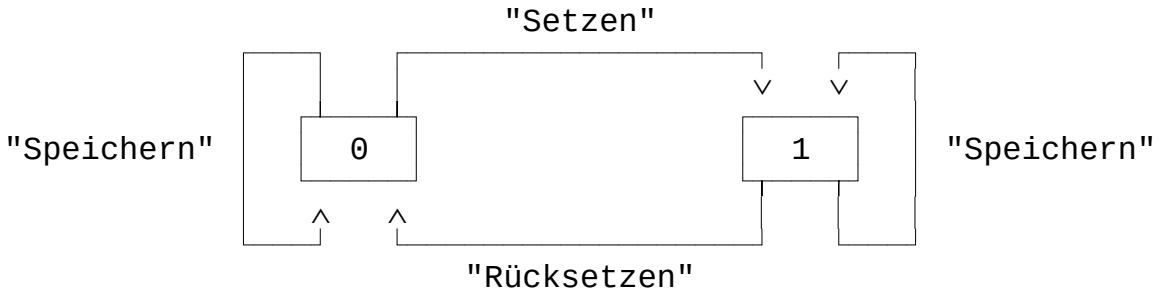
Was passiert genau bei $x_1x_2 = 00$? Wir vermuten, daß die vorherge-

hende Belegung von Bedeutung sein wird, und beziehen sie in die Analyse ein.

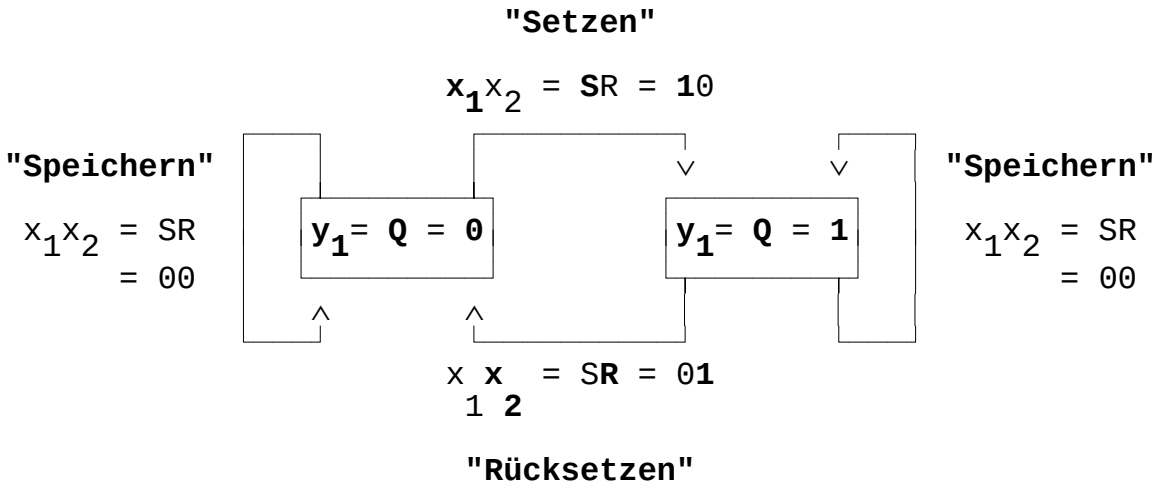
	x_1x_2	y_1y_2	x_1x_2	y_1y_2	x_1x_2	y_1y_2	x_1x_2	y_1y_2
t-1	0 0	? ?	0 1	**0 1**	1 0	**1 0**	1 1	1 1
t	0 0	? ?	0 0	**0 1**	0 0	**1 0**	0 0	x x

? "speichert" "schwingt"

Im Vergleich mit dem o. a. Automatengraphen



gelingt die Deutung

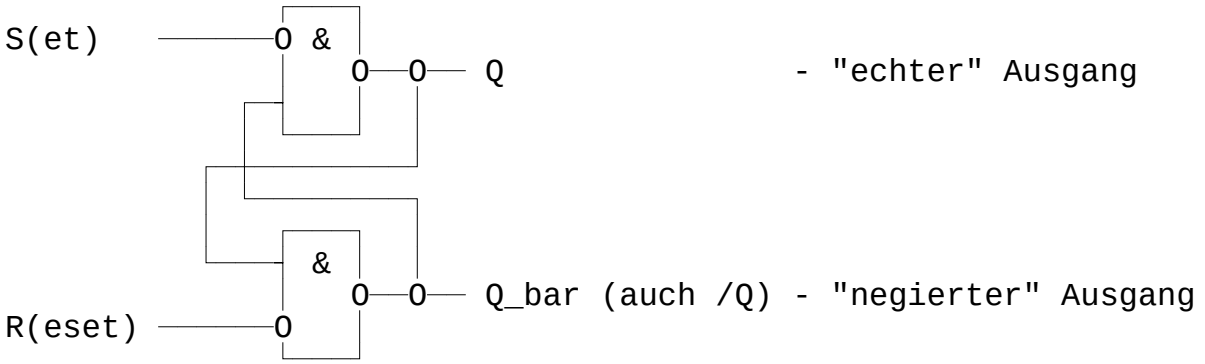


$\overline{S} \wedge \overline{R} = 1$ —-> Weder Setzen noch Rücksetzen ⟶ Speichern

$\overline{S} \wedge R = 1$ —-> Nicht Setzen, aber Rücksetzen ⟶ Rücksetzen

$S \wedge \overline{R} = 1$ —-> Setzen, aber nicht Rücksetzen ⟶ Setzen

$S \wedge R = 1$ —-> Sowohl Setzen als auch Rücksetzen ⟶ "verboten"



Wir wollen dieses Flipflop in den weiteren Betrachtungen **NAND - RS-Grund-Flipflop (NAND-RS-GFF)** nennen.

Die Belegung SR = 11 ist nicht eigentlich verboten, da sie ja zur definierten Ausgangsbelegung QQ_bar = 11 führt (Q_bar = /Q ist dabei zwar nicht erfüllt, aber das ist eher ein zu verschmerzender Mangel). Wenn jedoch auf SR = 11 unmittelbar SR = 00 ("Speichern") folgt, ist die Ausgangsbelegung des Flipflops nicht vorhersagbar (s. 5.2.2.1.1. Experiment).

Bevor wir die zugehörige Automatentabelle angeben, sehen wir uns noch eine erweiterte Automatentabelle an, in der neben Q und Q' auch die negierten Ausgänge Q_bar und Q_bar' aufgeführt sind.

	S R	Q Q_bar	Q' Q_bar'	Bemerkung
	(0 0	0 0	? ?)	QQ_bar = 00 nicht einstellbar
——>	0 0	0 1	0 1	Speichern
——>	0 0	1 0	1 0	Speichern
	0 0	1 1	? ?	Q'Q_bar' nicht vorhersagbar
	(0 1	0 0	0 1)	QQ_bar = 00 nicht einstellbar
——>	0 1	0 1	0 1	Rücksetzen
——>	0 1	1 0	0 1	Rücksetzen
	0 1	1 1	0 1	Rücksetzen
	(1 0	0 0	1 0)	QQ_bar = 00 nicht einstellbar
——>	1 0	0 1	1 0	Setzen
——>	1 0	1 0	1 0	Setzen
	1 0	1 1	1 0	Setzen
	(1 1	0 0	1 1)	QQ_bar = 00 nicht einstellbar
	1 1	0 1	1 1	"verboten"
	1 1	1 0	1 1	"verboten"
	1 1	1 1	1 1	"verboten"

In die eigentliche Automatentabelle werden für dieses Flipflop nur diejenigen Zeilen aufgenommen, für die **Q_bar = /Q** und **S∧R = 0** gilt (mit "——>" markiert). Die Spalten Q_bar und Q_bar' werden weggelassen.

	S R	Q	Q'	Bemerkung
——>	0 0	0	0	Speichern
——>	0 0	1	1	Speichern
——>	0 1	0	0	Rücksetzen
——>	0 1	1	0	Rücksetzen
——>	1 0	0	1	Setzen
——>	1 0	1	1	Setzen

!
S∧R = 0

Gelegentlich findet man allerdings auch Automatentabellen, bei denen die Belegung SR = 11 zusätzlich angegeben ist. Obwohl SR = 11 zu Q'Q_bar'= 11 führt, nimmt man den schlechtesten Fall an, d. h. auf SR = 11 folgt SR = 00 ("Speichern"), und schreibt

	S R	Q	Q'	Bemerkung
——>	0 0	0	0	Speichern
——>	0 0	1	1	Speichern
——>	0 1	0	0	Rücksetzen
——>	0 1	1	0	Rücksetzen
——>	1 0	0	1	Setzen
——>	1 0	1	1	Setzen
	1 1	**0**	**X**	**"verboten"**
	1 1	**1**	**X**	**"verboten"**

Aus dieser Automatentabelle kann die BOOLEsche Funktion Q' = f(S,R,Q) ausgelesen werden. Eintragen in den KARNAUGH-Plan und Minimieren liefert die sog. Charakteristische Gleichung dieses Flipflops. Beim Minimieren dürfen die mit "X" markierten Felder nicht mit in Primimplikanten erfaßt werden.

Q'	0	1	Q
00	0	1	
01	0	0	
11	X	X	
10	1	1	
SR			

Charakteristische Gleichung:

$$\mathbf{Q' = S \vee \bar{R}Q} \quad (\text{mit } S \wedge R \overset{!}{=} 0)$$

Die **Automatentabelle**

S R	Q	Q'	Bemerkung
0 0	0	0	Speichern
0 0	1	1	Speichern
0 1	0	0	Rücksetzen
0 1	1	0	Rücksetzen
1 0	0	1	Setzen
1 0	1	1	Setzen

$S \wedge R \overset{!}{=} 0$

kann vereinfacht werden zur **Flipflop-Tabelle**

S R	Q'	Bemerkung
0 0	Q	Speichern
0 1	0	Rücksetzen
1 0	1	Setzen

$S \wedge R \overset{!}{=} 0$

und aus der Automatentabelle kann die dritte mögliche Tabelle,

die **Substitutionstabelle** (zur Begriffsbildung s. u.), abgeleitet werden

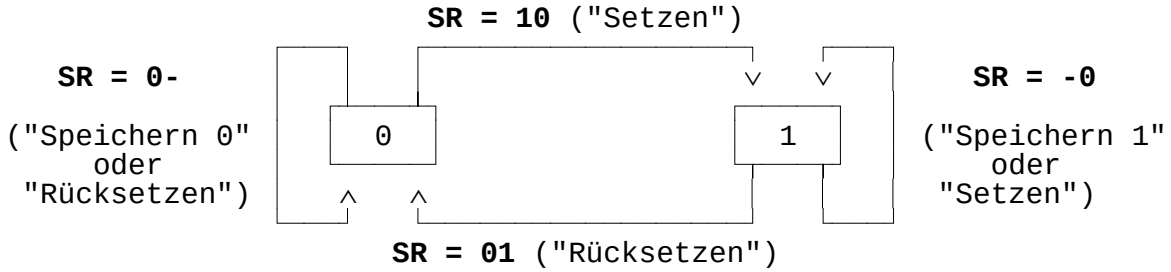
Q Q’	S R	Bemerkung
0 0	0 -	Speichern 0 oder Rücksetzen
1 1	- 0	Speichern 1 oder Setzen
1 0	0 1	Rücksetzen
0 1	1 0	Setzen

$S \wedge R \overset{!}{=} 0$

"-" steht für "Don’t Care" und bedeutet, daß jeder beliebige, gültige Pegel möglich ist (- = 0|1).

Alle drei angegebenen Tabellen enthalten die gleiche Information. Die Information wird nur unterschiedlich präsentiert. Jede der drei Tabellen hat ihre Existenzberechtigung und ist bei der Beantwortung gewisser Fragestellungen hilfreich.

Die **Flipflop-Tabelle** beschreibt in kompakter Form die Funktionsweise des Flipflops selbst. Die **Automatentabelle** ist dann hilfreich, wenn ein Flipflop als Automat aufgefaßt werden muß, z. B. bei der sog. Flipflop-Substitution, bei der ein Flipflop unter Verwendung eines Flipflops anderen Typs (zu Flipflop-Typen s. u.) realisiert werden soll. Die **Substitutionstabelle** beantwortet die Frage: Welche Eingangsbelegung SR muß man anlegen, wenn die Ausgangsbelegung des Flipflops von Q nach Q’ wechseln soll (mit QQ’ = 00|01|10|11)? Sie wird im Automatenentwurf (s. u.) verwendet. Aus ihr kann auch die Kantenbeschriftung des **Automatengraphen** abgelesen werden.

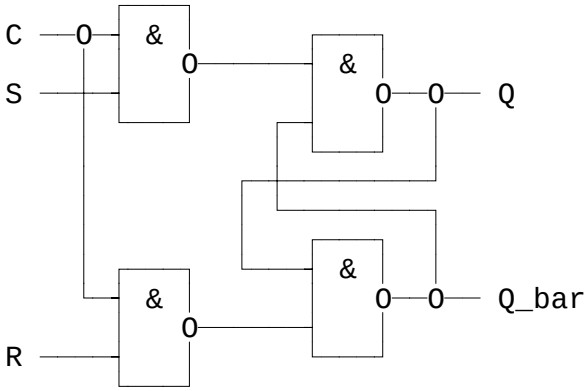


Beachten Sie die Unterschiede in der Kantenbeschriftung im Vergleich zu dem Automatengraphen, der am Anfang der Überlegungen stand! Die zuletzt angegebene Form werden wir in der Folge ausschließlich verwenden!

5.2.2.1.3. Taktzustandsgesteuerte (statische) Flipflops

Für den Einsatz der Flipflops in Zustandsspeichern von Automaten ist es wünschenswert, mit **einem einzigen**, von außen zugeführten Synchronisations- oder Taktsignal ("clock") zwischen Zeitbereichen, in denen Daten in das Flipflop eingetragen werden ("Setzen", "Rücksetzen"), und Zeitbereichen, in denen Daten im Flipflop gehalten werden ("Speichern"), unterscheiden zu können. Diese Forderung führt zunächst zum Konzept des taktzustandsgesteuerten (statischen) Flipflops.

5.2.2.1.3.1. Statisches RS-Flipflop

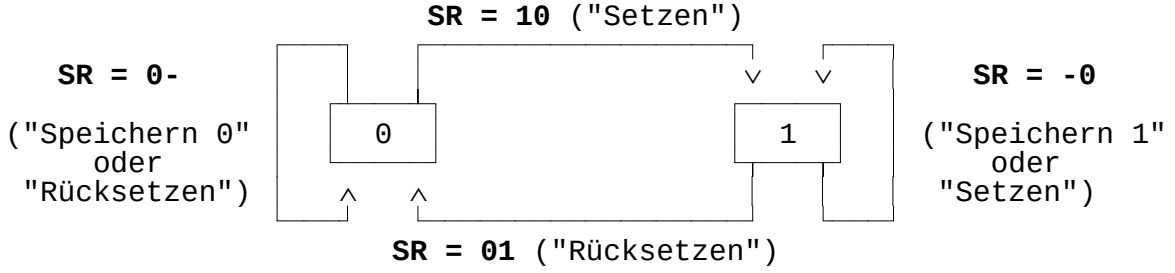


Die (ausführliche) Flipflop-Tabelle ergibt sich zu

C S R	Q'	Bemerkung
0 - -	Q	Speichern
1 0 0	Q	Speichern
1 0 1	0	Rücksetzen
1 1 0	1	Setzen

$S \wedge R \overset{!}{=} 0$

Üblicherweise läßt man aber die Zeile, in der das Verhalten bei inaktiver Taktbelegung (hier: C = 0) beschrieben wird, in der Automatentabelle weg und nimmt diese Zeile auch nicht in den Automatengraphen auf. Die Automatentabelle und der Automatengraph des taktzustandsgesteuerten RS-Flipflops sind folglich identisch mit der Automatentabelle und dem Automatengraphen des NAND-RS-GFF.

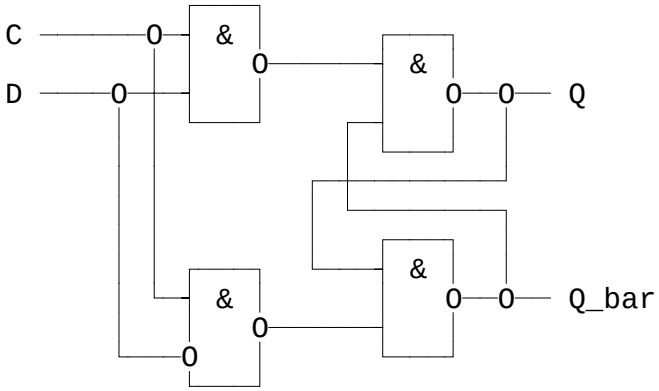


Man vereinbart, daß eine Kante dann und nur dann durchlaufen wird, wenn das Taktsignal aktiv ist (hier: C = 1). Bei inaktivem Taktsignal verharrt das Flipflop in seinem "alten" Zustand.

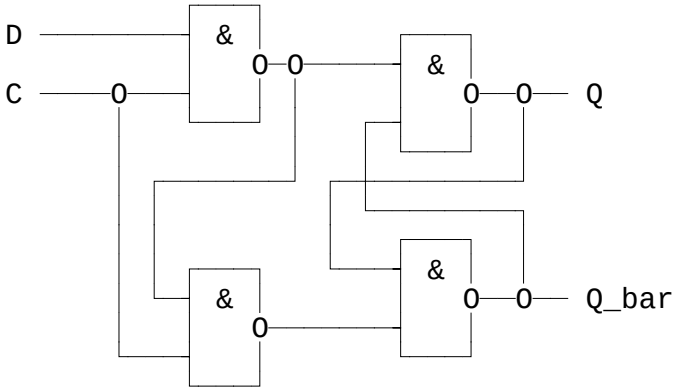
5.2.2.1.3.2. Statisches D-Flipflop

Da einerseits die "verbotene" Belegung (SR = 11) scheinbar nicht sinnvoll genutzt werden kann und andererseits Speichern (SR = 00) bei taktzustandsgesteuerten Flipflops auch mit der inaktiven Taktbelegung realisierbar ist, kann man auf die Belegungen SR =

11|00 auch gänzlich verzichten. Mit S = D und R = /S = /D folgt daraus das statische D-FF (**D** von **d**ata oder auch **d**elay)



Häufiger findet man allerdings eine kostengünstigere Realisierung (Überzeugen Sie sich von der logischen Identität beider Realisierungen!)

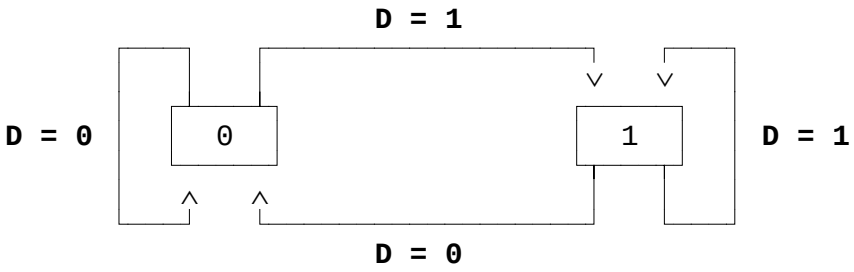


Die Tabellen und der Automatengraph fallen deutlich einfacher aus.

D	Q’
0	0
1	1

D	Q	Q’
0	0	0
0	1	0
1	0	1
1	1	1

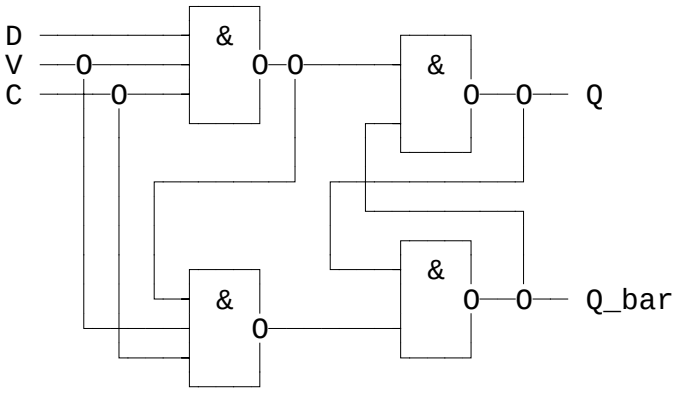
Q	Q’	D
0	0	0
0	1	1
1	0	0
1	1	1



Man könnte der Auffassung sein, daß im Zustandspeicher des Automatem aus Beispiel 5.15. genau solche statischen D-FFs zum Einsatz kommen könnten. Warum nicht, wird uns später klar werden.

5.2.2.1.3.3. Statisches DV-Flipflop

Wenn man Speichern nicht nur mit der inaktiven Taktbelegung realisieren können will, greift man zum DV-FF (**V** von **v**alid).

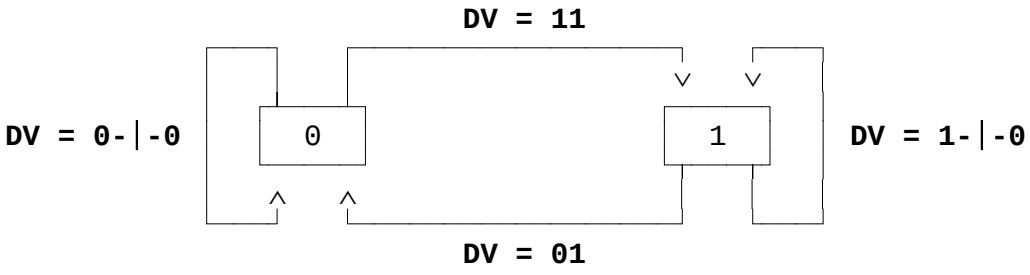


mit

D V	Q'
0 0	Q
0 1	0
1 0	Q
1 1	1

D V	Q	Q'
0 0	0	0
0 0	1	1
0 1	0	0
0 1	1	0
1 0	0	0
1 0	1	1
1 1	0	1
1 1	1	1

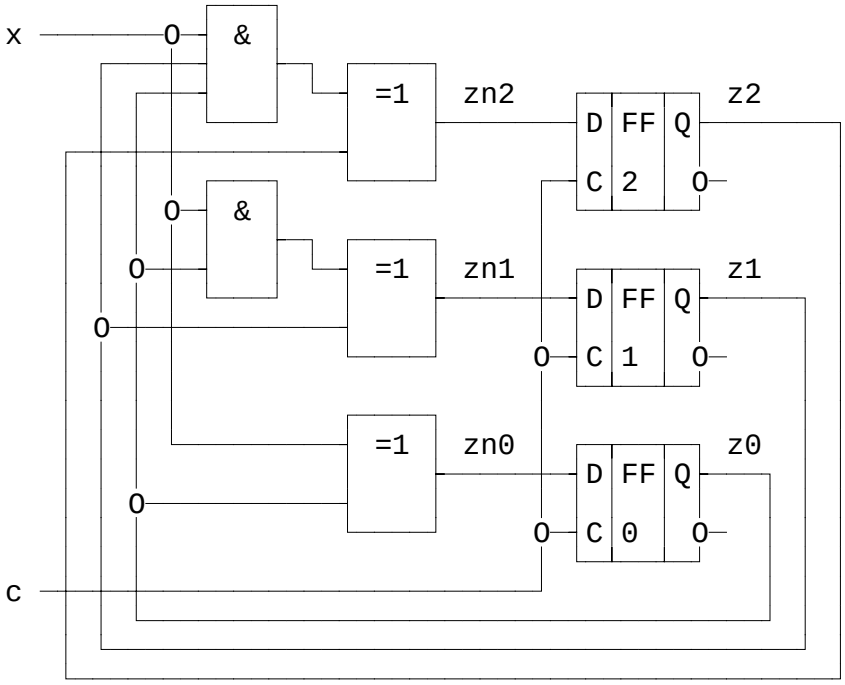
Q Q'	D V	
0 0	0 -	<——
0 0	- 0	<——
0 1	1 1	
1 0	0 1	
1 1	1 -	<——
1 1	- 0	<——



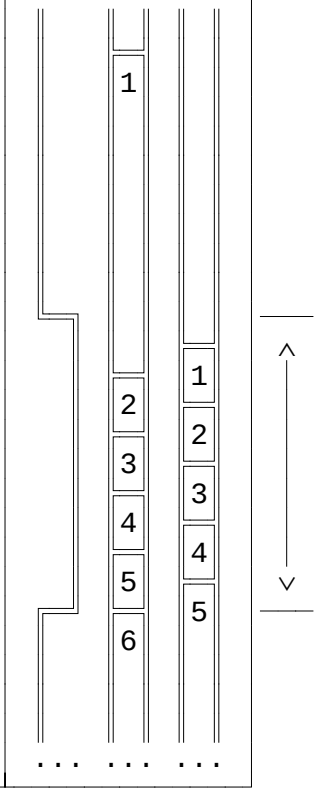
5.2.2.1.3.4. Nachteile statischer Flipflops

Wir betrachten den Automaten aus Beispiel 5.15. Der Zustandsspeicher sei aus statischen D-FFs aufgebaut. Wir nehmen an, daß die Überführungsfunktion und die Flipflops jeweils eine Verzögerungszeit von 5 ns aufweisen. Das Taktsignal habe eine Frequenz von 10 MHz (T_{clock} = 100 ns) und ein Tastverhältnis von 50%. Der Eingang x sei mit 1 belegt.

Wir simulieren das Verhalten des Automatem mit Hilfe eines geeigneten Simlators (VHDL, PSpice, ...).



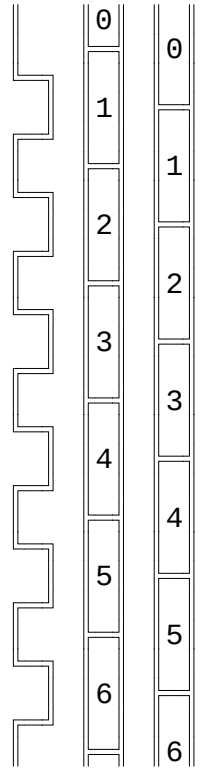
ns	c	zn 210	z 210
0	0	000	000
5	0	001	000
10	0	001	000
15	0	001	000
20	0	001	000
25	0	001	000
30	0	001	000
35	0	001	000
40	0	001	000
45	0	001	000
50	**1**	001	000
55	**1**	001	001
60	**1**	010	001
65	**1**	010	010
70	**1**	011	010
75	**1**	011	011
80	**1**	100	011
85	**1**	100	100
90	**1**	101	100
95	**1**	101	101
100	0	110	101
105	0	110	101
110	0	110	101
115	0	110	101
120	0	110	101
...	.	...	...



Der Automat "läuft durch", solange C = 1 anliegt. Er "schwingt".

Wie läßt sich das "Durchlaufen" verhindern? Das Taktsignal muß offenbar den Wert 0 annehmen, bevor die Überführungsfunktion den nächsten Zustand bereitstellt. Andererseits muß das Taktsignal mindestens solange den Wert 1 annehmen, bis der Zustandsspeicher den nächsten Zustand sicher übernommen hat. Wir wählen T_{clock} = 16 ns und erhalten:

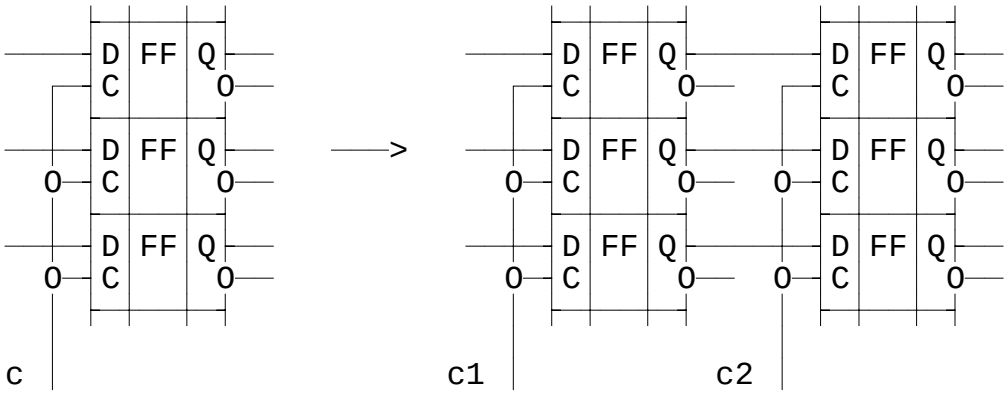
ns	c	zn 210	z 210
0	0	000	000
5	0	001	000
8	1	001	000
13	1	001	001
16	0	001	001
18	0	010	001
24	1	010	001
29	1	010	010
32	0	010	010
34	0	011	010
40	1	011	010
45	1	011	011
48	0	011	011
50	0	100	011
56	1	100	011
61	1	100	100
64	0	100	100
66	0	101	100
72	1	101	100
77	1	101	101
80	0	101	101
82	0	110	101
88	1	110	101
93	1	110	110
96	0	110	110
98	0	111	110
..	.	...	...

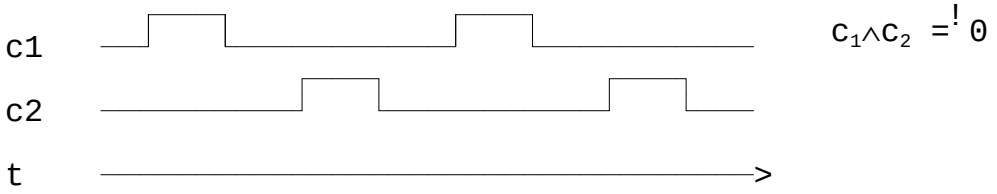


Jetzt gefällt uns das Verhalten des Automaten! Mit jedem Taktimpuls wird der Wert des Zählers um 1 erhöht.

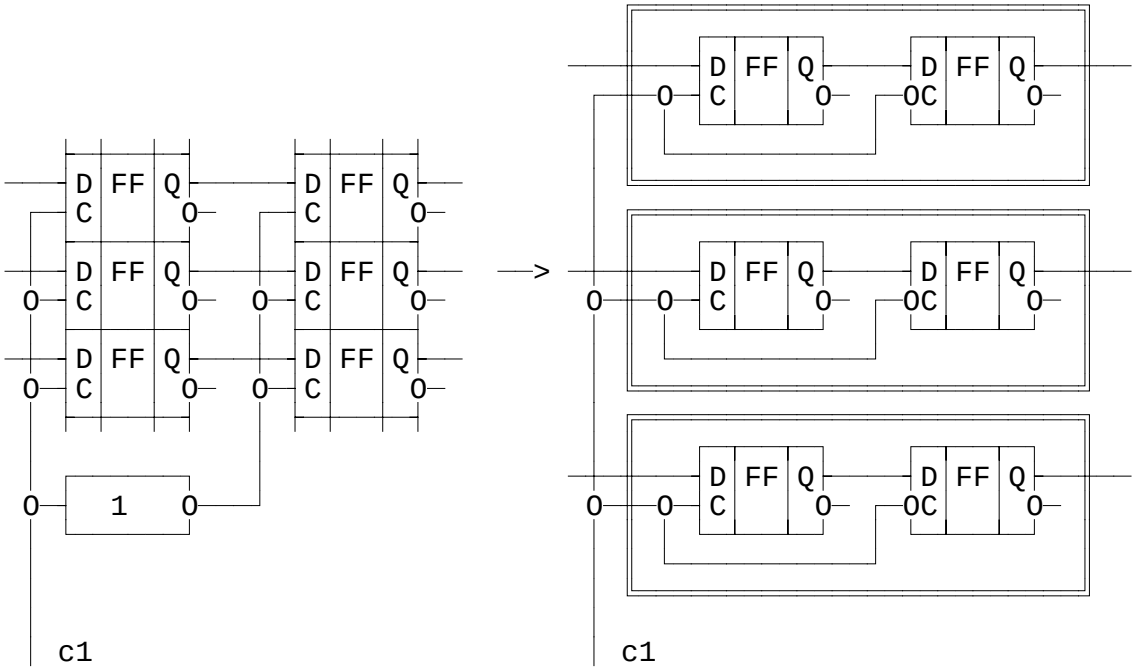
Das Problem der statischen Flipflops ist offenbar ihre Eigenschaft, daß bei aktiver Taktbelegung Änderungen an den Eingangsbelegungen "sofort" auf den Ausgang der Flipflops "durchschlagen" können. Das macht sie für Automaten nur begrenzt geeignet. Kurze Taktimpulse, wie oben verwendet, führen zu erheblichen elektrischen Problemen, und sind als generelle Lösung ungeeignet. Wünschenswert ist ein Verhalten der Flipflops, bei dem unabhängig von der Taktdauer der Automat nicht "durchläuft".

Eine früher oft praktizierte Methode ist das Doppeln des Registers des Zustandsspeichers und das Takten der beiden Register mit orthogonalen Takten.





Die Orthogonalitätsbedingung ist gerade noch erfüllt, wenn gilt $c_2 = /c_1$. Dann können auch zwei in einem Datenstrang liegende statische Flipflops zu einem Doppel-Flipflop zusammengefaßt werden.



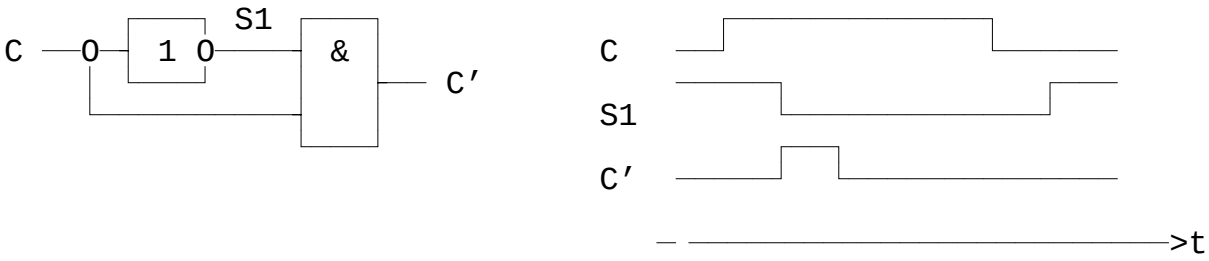
Da das jeweils rechte Teil-Flipflop eines solchen Doppel-Flipflops die Information aus dem jeweils linken Teil-Flipflop "sklavisch" übernimmt, nennt man des rechte Teil-Flipflop "Slave", das linke Teil-Flipflop Master und das gesamte Doppel-Flipflop "Master-Slave-Flip-flop" (MS-FF).

Die Tabellen und der Automatengraph eines MS-FF unterscheiden sich nicht von denen des entsprechenden taktzustandsgesteuerten Flipflops. Man vereinbart, daß eine Kante dann und nur dann durchlaufen wird, wenn das Taktsignal einen vollen Zyklus durchläuft (hier: C = 010). Sonst verharrt das Flipflop in seinem "alten" Zustand.

Eine prinzipiell andere Lösung sind taktflankengesteuerte (dynamische) Flipflops.

5.2.2.1.4. Taktflankengesteuerte (dynamische) Flipflops

Wir hatten in 5.2.2.1.3.4 festgestellt, daß bei hinreichend kurzen Taktsignalen statische Flipflops in Automaten eingesetzt werden können. Die Idee des taktflankengesteuerten Flipflops besteht nun darin, im Flipflop selbst aus der steigenden oder fallenden Flanke (rising edge, falling edge) des Taktsignals durch Differentiation einen hinreichend kurzen Taktimpuls zu gewinnen und damit das Flipflop zu takten.



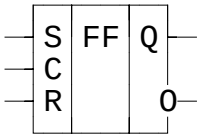
Die Tabellen und der Automatengraph eines taktflankengesteuerten Flipflops unterscheiden sich nicht von denen des entsprechenden taktzustandsgesteuerten Flipflops. Man vereinbart, daß eine Kante dann und nur dann durchlaufen wird, wenn das Taktsignal eine aktive Flanke (hier: steigende Flanke, C = 01) durchläuft. Sonst verharrt das Flip-flop in seinem "alten" Zustand.

Mit der schaltungstechnischen Realisierung der vorgestellten Flipflops befassen Sie sich im Hardwarepraktikum.

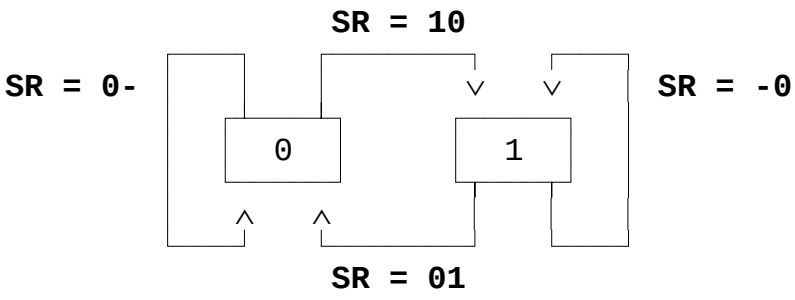
5.2.2.1.5. Zusammenstellung gängiger Flipflop-Typen

Die Art der Taktierung spiegelt sich weder in der Charakteristischen Gleichung, noch in den Tabellen, noch in den Automatengraphen wider. Mit Ausnahme des T- (von engl. to **t**oggle) und des JK-Flipflops (von engl. to **j**ump, to **k**ill oder dem Erfinder **J**ack **K**ilby), die nur als MS-Flipflop oder taktflankengesteuertes Flipflop realisierbar sind (Warum ?), gelten die nachfolgenden Angaben also für alle Varianten der Taktierung. Für jedes Flipflop sind Schaltsymbol, Charakteristische Gleichung, Automatengraph, Flipflop-Tabelle, Automatentabelle und Substitutionstabelle aufgeführt.

RS-FF



$\mathbf{Q' = S \vee \bar{R}Q}$ (mit $S \wedge R \stackrel{!}{=} 0$)

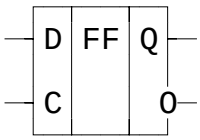


S R	Q'
0 0	Q
0 1	0
1 0	1

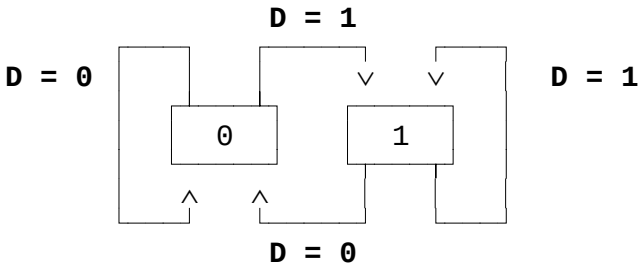
S R	Q	Q'
0 0	0	0
0 0	1	1
0 1	0	0
0 1	1	0
1 0	0	1
1 0	1	1

Q Q'	S R
0 0	0 -
1 1	- 0
1 0	0 1
0 1	1 0

D-FF



$Q' = D$

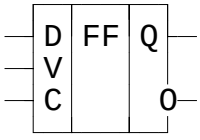


D	Q′
0	0
1	1

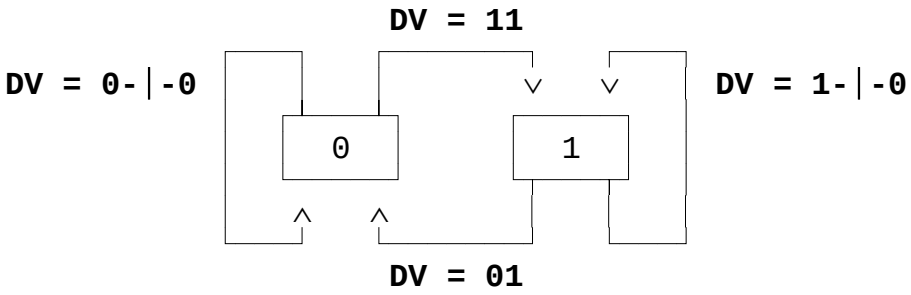
D	Q	Q′
0	0	0
0	1	0
1	0	1
1	1	1

Q	Q′	D
0	0	0
0	1	1
1	0	0
1	1	1

DV-FF



$Q' = DV \vee \overline{V}Q$

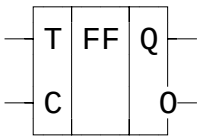


D	V	Q′
0	0	Q
0	1	0
1	0	Q
1	1	1

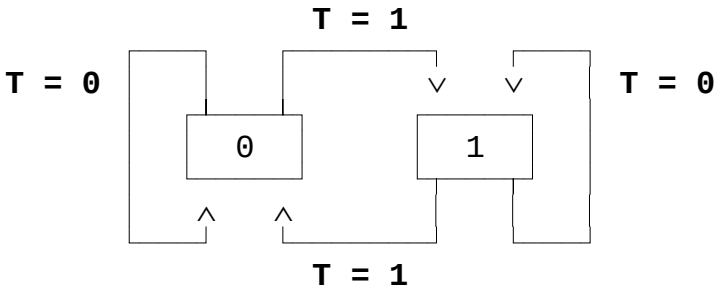
D	V	Q	Q′
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Q	Q′	D	V
0	0	0	-
0	0	-	0
0	1	1	1
1	0	0	1
1	1	1	-
1	1	-	0

T-FF



$Q' = T \not+ Q$

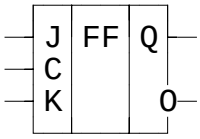


T	Q′
0	Q
1	/Q

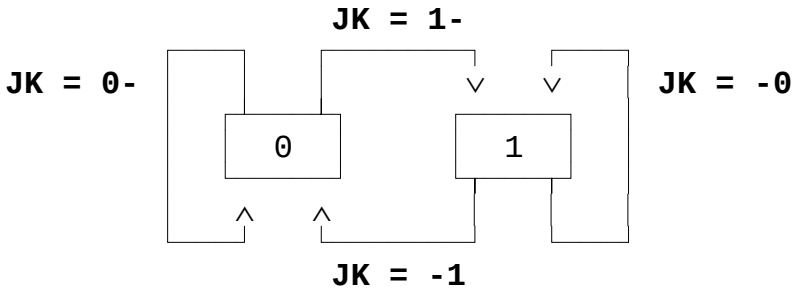
T	Q	Q′
0	0	0
0	1	1
1	0	1
1	1	0

Q	Q′	T
0	0	0
0	1	1
1	0	1
1	1	0

JK-FF



$Q' = J\overline{Q} \vee \overline{K}Q$



J	K	Q′
0	0	Q
0	1	0
1	0	1
1	1	/Q

J	K	Q	Q′
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

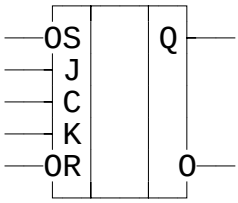
Q	Q′	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

Alle vorgestellten Flipflop-Typen werden Ihnen begegnen. In Automaten werden wir in erster Linie taktflankengesteuerte und Master-Slave-Flipflops einsetzen. Im Hardwarepraktikum werden Sie kennenlernen, daß taktflankengesteuerte und Master-Slave-Flipflops über weite Strecken identisches Verhalten zeigen, daß sie aber durchaus auch Unterschiede aufweisen, die insbesondere bei der Gestaltung des Taktsignals zu beachten sind.

5.2.2.1.6. Handelsübliche Flipflops

Handelsübliche Flipflops, etwa das taktflankengesteuerte D-FF SN7474 oder das JK-MS-FF SN7472, sind zudem statisch setz- und rücksetzbar. Sie vereinen in sich die Eigenschaften eines NAND-RS-GFF und eines taktflankengesteuerten bzw. Master-Slave-Flipflops. Die statische Setz- und Rücksetzbarkeit wird dabei oft zur Einstellung eines definierten Anfangszustands (Initialisierung) des Flipflops verwendet.

Für das JK-MS-FF SN7472 gelten z. B. das Schaltsymbol, die (erweiterte) Flipflop-Tabelle und die "Ersatzschaltung"



/S	/R	C	J	K	Q'	Q_bar'
0	1	-	-	-	1	0
1	0	-	-	-	0	1
0	0	-	-	-	1	1
1	1	⊓	0	0	Q	Q_bar
1	1	⊓	1	0	1	0
1	1	⊓	0	1	0	1
1	1	⊓	1	1	/Q	/Q_bar