

9 Synchrone Schaltwerke

9.1 Einleitung

Sequentielle Schaltungen (Schaltwerke)



Ein Schaltwerk besteht aus einem **Schaltnetz** und einem **Speicherteil** zur Sicherung vorangegangener Informationen. Die Ausgangsvariablen (Steuervariablen) z^t hängen von den Eingangsvariablen x^t und von den gespeicherter Information (aktueller Zustand) y^t ab. Als Speicherelemente kommen hauptsächlich Master-Slave-D-Flipflops (CMOS) zum Einsatz (siehe 6.3.4). Zur Synchronisation wird der Speicher getaktet (synchrone Schaltwerk).

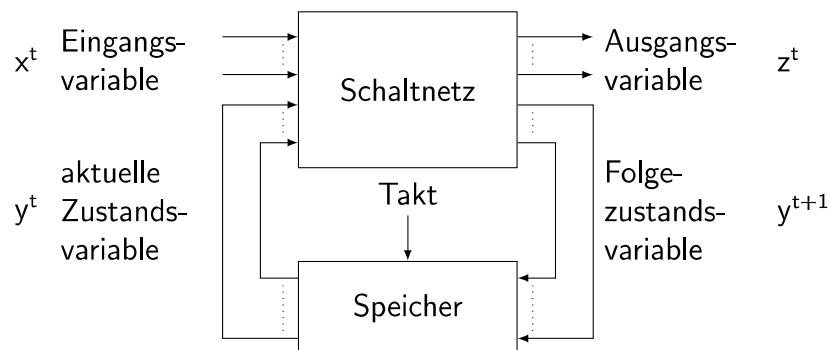


Abbildung 9.1: Schaltwerksstruktur

Darstellungen für Schaltwerke

Schaltwerke können durch Zustandstabellen bzw. Zustandsgraphen/Übergangsgraphen dargestellt werden. Bei Zustandstabellen werden ausgehend vom aktuellen Zustand y^t in Verbindung mit den Eingangsvariablen x^t die Folgezustände y^{t+1} und die Ausgangsvariablen z^t aufgezeigt. Zustands-/Übergangsgraphen visualisieren die Funktionsweise eines Schaltwerks graphisch. Hierbei repräsentieren die Knoten die Zustände. Die Kanten zeigen die Übergänge mit den zugehörigen Eingangs- und Ausgangssignalen an. Die Ausgangssignale können den Knoten oder den Kanten zugeordnet werden.

Arten von Automaten

Im Wesentlichen kann man zwei Arten von Schaltwerken/Automaten unterscheiden. Der **Mealy-Automat** ist dadurch gekennzeichnet, dass die Ausgangsvariablen z^t eine Funk-

tion der Eingangsvariablen x^t und des aktuellen Zustands y^t ist. Die Folgezustände y^{t+1} sind eine Funktion der Eingangsvariablen x^t und des aktuellen Zustands y^t .

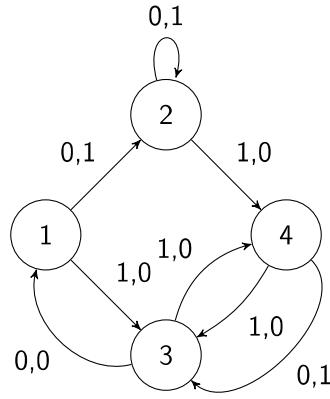
$$y^{t+1} = g(x^t, y^t)$$

$$z^t = f(x^t, y^t)$$

Beispiel 82. Mealy-Automat mit Zustandstabelle und Zustandsgraphen

y^t	$x = 0$		$x = 1$	
	y^{t+1}	z	y^{t+1}	z
1	2	1	3	0
2	2	1	4	0
3	1	0	4	0
4	3	1	3	0

(a) Zustandstabelle



(b) Zustandsgraph

Abbildung 9.2: Mealy-Automat

Der zweite Automatentyp wird **Moore-Automat** genannt. Dieses Schaltwerk bestimmt die Ausgangsvariablen z^t nur anhand des aktuellen Zustandes y^t . Der Folgezustand y^{t+1} bleibt eine Funktion der Eingangsvariablen x^t und des aktuellen Zustands y^t .

$$y^{t+1} = g(x^t, y^t)$$

$$z^t = f(y^t)$$

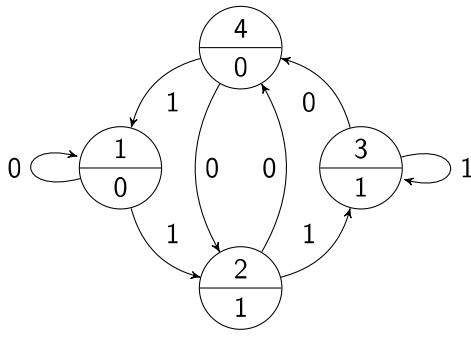
Beispiel 83. Moore-Automat mit Zustandstabelle und Zustandsgraphen

Eigenschaften der Automatentypen

Beide Steuerwerksarten sind äquivalent und jeweils in den anderen umwandelbar. Die Synthese beider Automatentypen ist ähnlich und muss dementsprechend nicht separat erklärt werden. Moore-Automaten erfordern in der Regel mehr Zustände. Mealy-Automaten hingegen können komplexere Übergangsfunktionen aufweisen. Die Steuerfunktionen können beim Mealy-Automaten asynchron reagieren, wenn sie nicht zwischengespeichert werden.

y^t	$x = 0$	$x = 1$	z
	y^{t+1}	y^{t+1}	
1	1	2	0
2	4	3	1
3	4	3	1
4	2	1	0

(a) Zustandstabelle



(b) Zustandsgraph

Abbildung 9.3: Moore-Automat

9.2 Schaltwerkssynthese

Schritte zur Schaltwerkssynthese

Die Schaltwerkssynthese kann im Wesentlichen in zwei Schritten realisiert werden. Zu Beginn wird dabei aus der Verhaltensbeschreibung einer sequentiellen Funktion eine Zustandstabelle bzw. ein Zustandsgraph abgeleitet. Die Ableitung einer Zustandstabelle aus der Verhaltensbeschreibung (High-Level-Synthese) bietet bisher noch nicht umfassend effiziente Lösungen und bleibt z. Z. aktueller Forschungsgegenstand. Der zweite Schritt beinhaltet die Realisierung der Zustandstabelle durch ein Schaltwerk. Die Umsetzung der Zustandstabelle in ein Schaltwerk ist ein weitgehend erforschter Bereich mit effizienten Lösungsansätzen.

Zustandszuordnung (state assignment)

Als eine Zustandszuordnung (state assignment) bezeichnet man eine Zuweisung einer Kombination von Zustandsvariablen y_i zu einem Zustand des Schaltwerks. Mit zweiwertigen Speicherelementen sind für n Zustände des Schaltwerks $\log_2 n$ Speicherelemente erforderlich. Daraus ergeben sich dann $n!$ verschiedene Zustandszuordnungen, deren Realisierungen zu unterschiedlich hohem Aufwand führen kann. Es sind verschiedene Algorithmen bekannt, die zu einer optimalen Zustandszuordnung führen. Ziel sind dabei z.B. eine minimale Anzahl von Gattereingängen, eine minimale Verzögerung oder minimale Verlustleistung. Eine Minimierung ist aber nur bei Schaltnetzrealisierung relevant, nicht bei ROM-Implementierung.

Beispiel 84. *Sequentieller serieller Bitprüfer als Moore-Automat mit MS-D-Flipflops*

Verhaltensbeschreibung

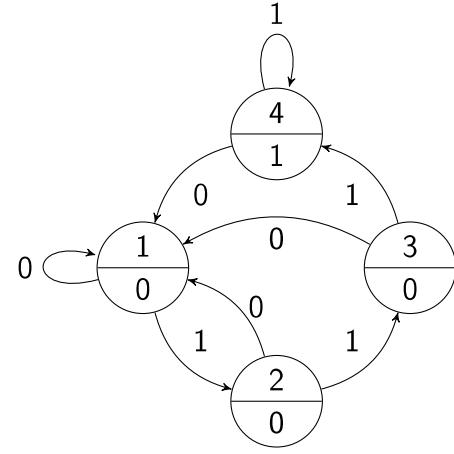
Die Schaltung soll für die Steuerfunktion z eine 1 erzeugen, wenn in einem bitseriellen Datenstrom 3 oder mehr aufeinanderfolgende Einsen auftreten.

Ableitung einer Zustandstabelle bzw. eines Zustandsgraphen

Zu Beginn muss gemäß der Verhaltensbeschreibung ein Ablaufschema entwickelt werden. Als mögliche Darstellungsform eignen sich eine Zustandstabelle und/oder ein Zustandsgraph.

aktueller Zustand y^t	neuer Zustand y^{t+1}		Ausgang z
	$x = 1$	$x = 0$	
1	2	1	0
2	3	1	0
3	4	1	0
4	4	1	1

(a) Zustandstabelle



(b) Zustandsgraph

Abbildung 9.4: Bitprüfer

Realisierung der Zustandstabelle durch ein Schaltwerk

Auf Grundlage der Verhaltensanalyse durch die Zustandstabelle bzw. den Zustandsgraphen, ist eine Zustandscodierung vorzunehmen. Für die Codierung der vier vorgesehenen Zustände benötigt man entsprechend zwei Variablen (y_2, y_1). Folgerichtig ist die Zustandstabelle neu aufzustellen.

Zustand	$y_2 \quad y_1$		$x = 1$		$x = 0$		z
	y_2^t	y_1^t	y_2^{t+1}	y_1^{t+1}	y_2^{t+1}	y_1^{t+1}	
1	0	0	0	1	0	0	0
2	0	1	0	1	1	1	0
3	1	1	1	1	1	0	0
4	1	0	1	0	1	0	1

(a) Zustandszuordnung (Gray-Code)

(b) neue Zustandstabelle

Abbildung 9.5: Zustandstabelle nach Codierung

Aus der neuen Tabelle können nun die Übertragungsfunktionen für den Folgezustand (y_2^{t+1}, y_1^{t+1}) und den Ausgang (z) ermittelt werden.

$$\begin{aligned} y_1^{t+1} &= x \overline{y_2^t} & y_2^{t+1} &= x y_1^t + x y_2^t = x (y_1^t + y_2^t) & z &= \overline{y_1^t} y_2^t \\ \text{NOR-NF: } y_1^{t+1} &= \overline{x + y_2^t} & y_2^{t+1} &= \overline{x} + \overline{(y_1^t + y_2^t)} & z &= y_1^t + \overline{y_2^t} \end{aligned}$$

Der Bitprüfer soll durch ein Schaltwerk aus MS-D-Flipflops realisiert werden. Man benötigt also entsprechende Ansteuerfunktionen für die Flipflops. Hierbei ist anzumerken, dass die Ansteuerung direkt aus der Übertragungsfunktion abgeleitet werden kann ($D = Q^{t+1}$). Man setzt also D_1 für y_1^{t+1} und D_2 für y_2^{t+1} ein. Außerdem wird zur Vereinfachung y_1^t durch y_1 und y_2^t durch y_2 ersetzt.

Ansteuerfunktionen:

$$D_1 = \overline{\bar{x} + y_2} \quad D_2 = \overline{\bar{x} + (\bar{y}_1 + y_2)} \quad z = \overline{y_1 + y_2}$$

Das resultierende Schaltwerk ist in der nachfolgenden Abbildung skizziert.

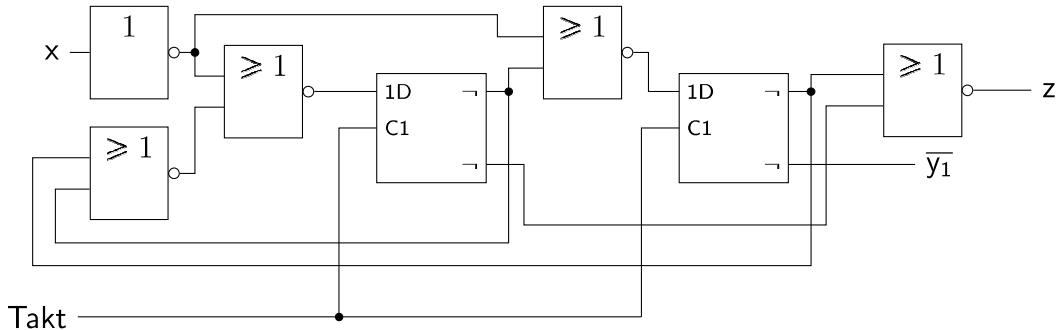


Abbildung 9.6: Prüfschaltung mit MS-D-Flipflops

Entwurf mit Festwertspeichern (ROMs)

Ein Schaltwerk kann gegebenenfalls auch mit einem ROM realisiert werden. Hierbei wird die Zustandstabelle durch den Festwertspeicher abgebildet. Die Eingangsvariablen x^t und Zustandsvariablen y^t werden als Adressbits verwendet. Die Folgezustandsvariablen y^{t+1} und die Ausgangsvariablen z^t werden im ROM gespeichert. Durch die Rückkopplung der Folgezustände auf den Eingang des ROMs wird ein sequentiellen Ablauf erreicht. Die Synchronisation wird mit MS-D-Flipflops oder Latches realisiert.

Entwurfsschritte:

Ausgangspunkt ist die Zustandszuordnungs- und die Ausgangstabelle. Die Tabelle wird als ROM-Übergangstabelle umgeformt und das Schaltwerk ist komplett realisiert. Häufig ist der ROM-Baustein aber größer (mehr Adressen (Zustände) und größere Wortbreite) als für ein spezielles Problem erforderlich. In diesen Fällen wird das ROM nur teilweise genutzt.

Beispiel 85. Erkennung einer Bitsequenz

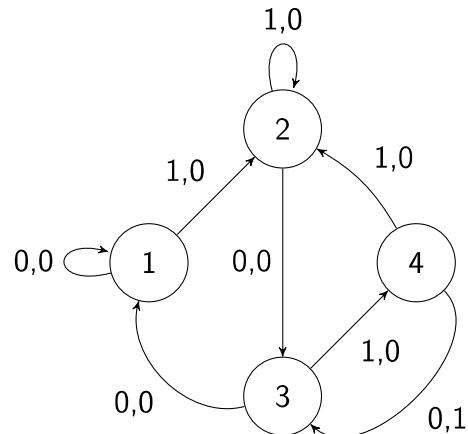
Verhaltensbeschreibung:

Die Schaltung erzeugt die Steuerfunktion $z = 1$, wenn die Bitsequenz 1010 erscheint. In allen anderen Fällen erscheint am Ausgang 0.

Ableitung einer Zustandstabelle bzw. eines Zustandsgraphen:

y^t	$x = 0$		$x = 1$	
	y^{t+1}	z	y^{t+1}	z
1	1	0	2	0
2	3	0	2	0
3	1	0	4	0
4	3	1	2	0

(a) Zustandstabelle



(b) Zustandsgraph

Abbildung 9.7: Bitsequenz-Erkennung

Realisierung der Zustandstabelle durch ein Schaltwerk:

Zustandszuordnung (Gray-Code):

Zustand	y_2	y_1
1	0	0
2	0	1
3	1	1
4	1	0

neue Zustandstabelle:

y_2^t	y_1^t	$x = 0$			z	$x = 1$		
		y_2^{t+1}	y_1^{t+1}	z		y_2^{t+1}	y_1^{t+1}	z
0	0	0	0	0	0	0	1	0
0	1	1	1	0	0	0	1	0
1	1	0	0	0	1	1	0	0
1	0	1	1	1	0	0	1	0

ROM-Übergangstabelle:

Adresse			Ausgang		
A ₂	A ₁	A ₀	D ₂	D ₁	D ₀
y ₂ ^t	y ₁ ^t	x	y ₂ ^{t+1}	y ₁ ^{t+1}	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	1	0	0

Durch eine Umsortierung nach Eingangs- und Ausgangsvariablen erhalten wir die benötigte ROM-Übergangstabelle. Der Eingang wird entsprechend als eine Adresse interpretiert. Schaltwerk:

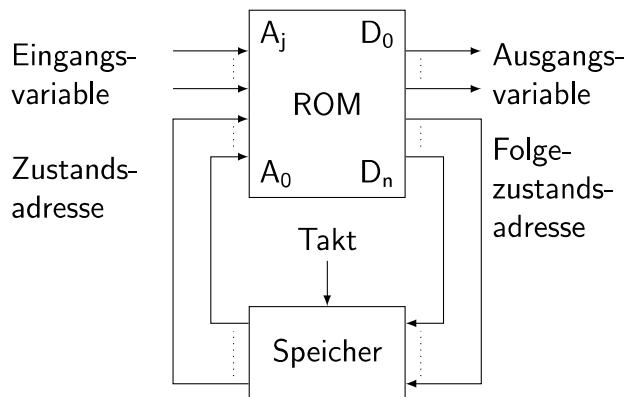


Abbildung 9.8: ROM-Implementierung

9.3 Register

9.3.1 Übersicht

Allgemeines

In Speicherorganisationen mit mehr als einem Bit Speicherkapazität erhalten die Speicherzellen eine gemeinsame Steuerung für die Datenein- und -ausgabe. Aufgebaut wird dieses System dann meist mit Flipflop-Speichern (Parallel- oder Serienspeicher). Speicher mit einer geringen Anzahl von Speicherzellen werden Register genannt. Register finden Anwendung als schnellen Zwischenspeicher für kleine Datenmengen.

Parallelspeicher

Die Datenein-/ausgabe erfolgt für alle Speicherzellen gleichzeitig. Die Steuerleitung T zur Datenübernahme ist parallel an alle Speicherzellen geschaltet. Parallelspeicher können

aus einfachen D-Latches oder aus MS-D-Flipflops aufgebaut werden.

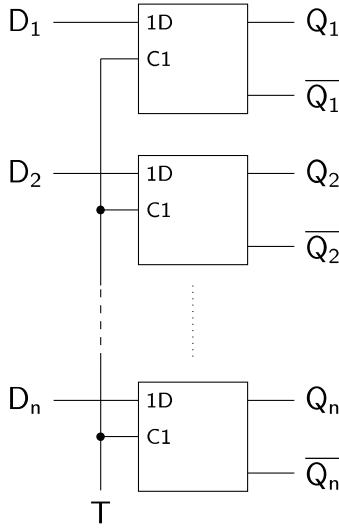


Abbildung 9.9: Parallelregister

Serienspeicher (Schieberegister)

Die Speicherzellen sind bei Serienspeichern in Reihe geschaltet. Jeder Ausgang der vorangehenden Zelle ist mit dem Eingang der nachfolgenden Zelle verknüpft. Die Daten werden beginnend von der ersten Zelle mit jedem Takt T in die nächste Zelle geschoben. Die Ausgabe der Daten erfolgt an der letzten Zelle der Speicherkette. Schieberegister sollten aus Master-Slave-D-Flipflops bestehen, da jede Speicherzelle gleichzeitig Daten übernehmen und weitergeben muss.

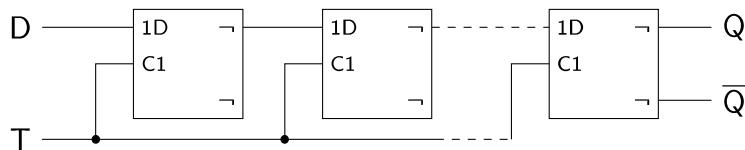


Abbildung 9.10: Serienspeicher (Schieberegister)

9.3.2 Technische Realisierung statischer Parallel- und Serienspeicher

Eigenschaften

Bei Bibliothekskomponenten handelt es sich meistens um Kombinationen aus Parallel- und Serienspeichern, um deren Anwendungsbereich zu erhöhen. Unterschieden werden

die Speicher nach Art der Ein- und Ausgabe der Daten an den Speicherzellen (parallel und/oder seriell). Weitere Eigenschaften der Speicherzellen können die Wahl der Schieberichtung nach rechts oder links sowie ein asynchrones Löschen oder Setzen sein.

Beispiel 86. 8-Bit-Schieberegister mit serieller Ein-/Ausgabe und paralleler Ausgabe

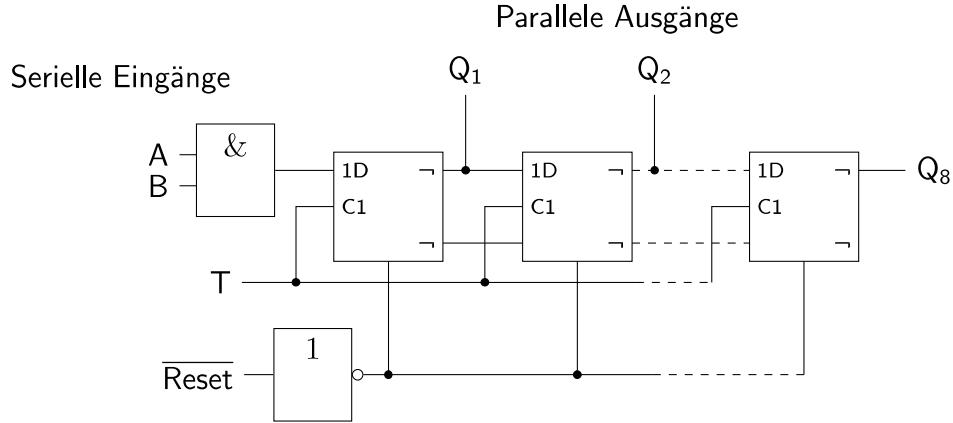


Abbildung 9.11: Schieberegister mit serieller Ein-/Ausgabe und paralleler Ausgabe

Diese Schaltung 9.11 ermöglicht die Serien-/Parallel-Umsetzung der Daten. Von den zwei seriellen Eingänge A und B kann jeweils einer zu Steuerzwecken verwendet werden. Ist einer der beiden Eingänge '1', wird der andere durchgeschaltet. Die Daten werden während der positiven Taktflanke (Slave→Master) durch das Register geschoben und liegen gleichzeitig an den parallelen Ausgängen Q₁,...Q₈ an. Das Register kann asynchron über den Reset-Eingang zurückgesetzt werden.

Beispiel 87. 8-Bit-Schieberegister mit serieller Ein-/Ausgabe und paralleler Eingabe
 Diese Schaltung 9.12 ermöglicht ebenfalls die Serien-/Parallel-Umsetzung der Daten. Dieses Schieberegister ist umschaltbar zwischen Schieberegisterbetrieb und parallelem Laden. Wenn das Signal Setzen = 1 ist, dann wird der Takt an den Flipflops gesperrt und die parallel anliegenden Daten zu den Eingängen I₁,...,I₈ unabhängig vom Takt (asynchron) in die Flipflops übernommen. Andernfalls (Setzen = 0) sind die parallelen Eingänge gesperrt und der Takt ist freigegeben. Die Daten werden dann in Richtung des seriellen Ausgangs Q₈ abhängig vom Takt (synchron) verschoben. Bei gesetztem Sperren-Eingang wird der Schieberegister unterbrochen und ein paralleles Laden ist ebenfalls nicht möglich.

Beispiel 88. Schieberegister mit zwei Schieberichtungen

Bei arithmetischen Operationen sind oftmals beide Schieberichtungen erforderlich. Für Schieberichtung von rechts nach links müssen die Ausgänge der Flipflops mit den Eingängen der links neben ihnen liegenden Flipflops verknüpft werden. Die steuerbare Schieberichtung erfordert einen Multiplexer, der je nach Schieberichtung den Ausgang mit dem Eingang des links oder rechts benachbarten Flipflops verbindet. Bei gesetztem R/L-Signal wird das Schieben nach rechts vorgegeben. Die Daten von D_R werden übernommen und an Q₄ ausgegeben. R/L = 0 bewirkt folgerichtig das Schieben nach links. Das Verschieben der Daten erfolgt mit steigender Taktflanke (0→1). Während des Schiebens darf

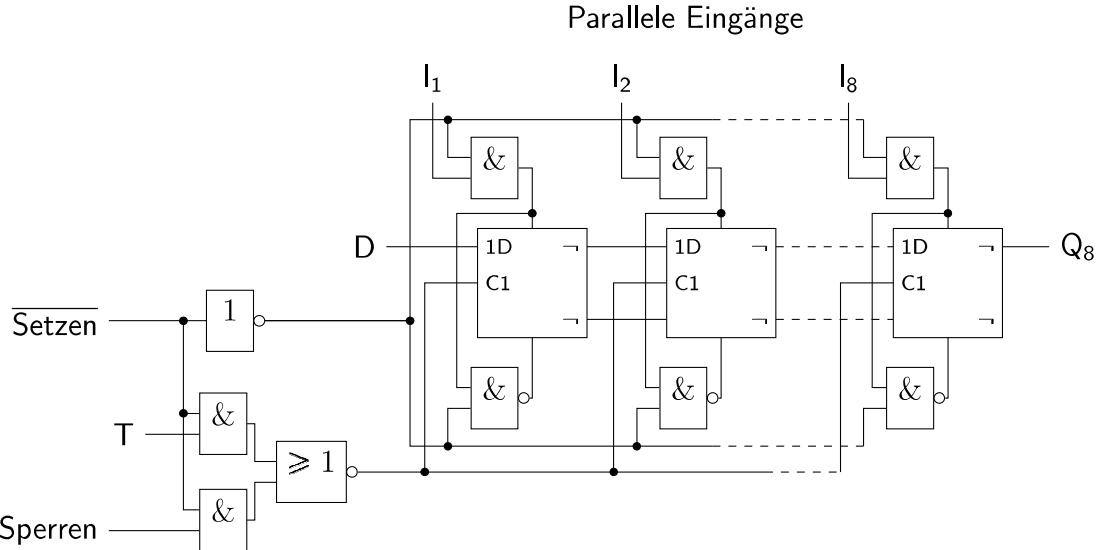


Abbildung 9.12: Schieberegister mit serieller Ein-/Ausgabe und paralleler Eingabe

die Schieberichtung nicht gewechselt werden, da sonst ein undefinierter Zustand auftritt. (siehe Abbildung 9.13)

9.4 Zähler und Untersetzer

9.4.1 Übersicht

Bei einem Zähler stellt jeder Zustand eine Zählerstellung dar. Eine einfache Sonderform der Zähler sind die Untersetzer (Frequenzteiler). Ein Untersetzer bewirkt nur eine Abfrage eines vorgegebenen Teilverhältnis und ist meist einfacher aufgebaut. Die gebräuchlichsten Zähler sind reine Dual-Code-(modulo 2)-Zähler und der BCD-Code-Zähler, bei dem sechs Zustände des Dualcodes übersprungen werden. Unterschieden werden die Systeme durch ihre jeweilige Betriebsart (asynchron oder synchron). Beim asynchronen Betrieb werden die Flipflops immer von dem in der Zählfolge vorangehenden Flipflop getaktet. Bei der synchronen Variante werden alle Flipflops eines Zählers durch eine gemeinsame Clock gleichzeitig getaktet.

9.4.2 Untersetzer

Asynchroner Untersetzer

Ein asynchroner Untersetzer kann z. B. durch eine Kaskadierung von D-Latches realisiert werden. Mit der Beschaltung $D = \bar{Q}$ wirken die n D-Latches als Untersetzer im Untersetzungsverhältnis $2^n : 1$. Der Q-Ausgang jedes Flipflops wird mit dem Takteingang des folgenden Flipflops verbunden. Die Verzögerungszeiten t_{pd} addieren sich mit jeder zusätzlichen Untersetzerstufe (Nachteil). Bei Untersetzern mit einem geringeren

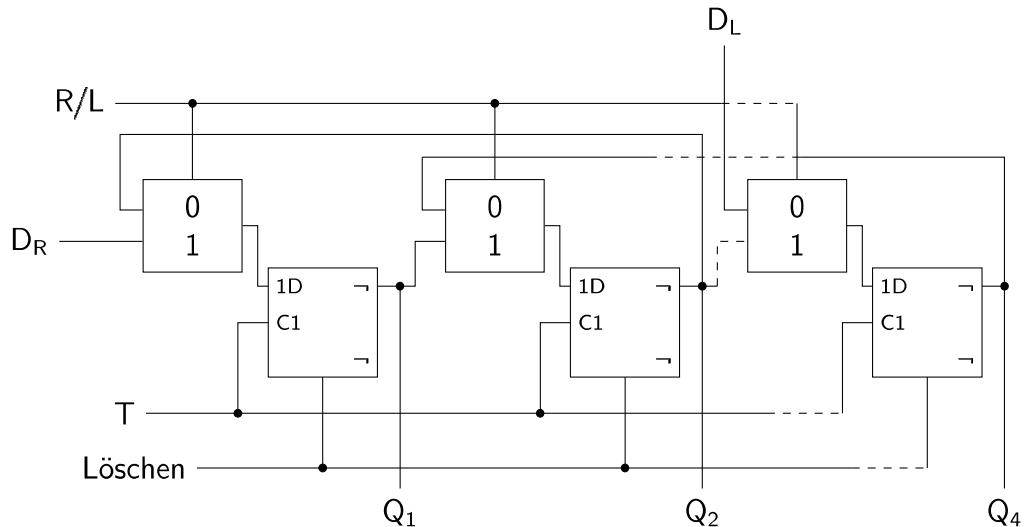


Abbildung 9.13: Schieberegister mit zwei Schieberichtungen

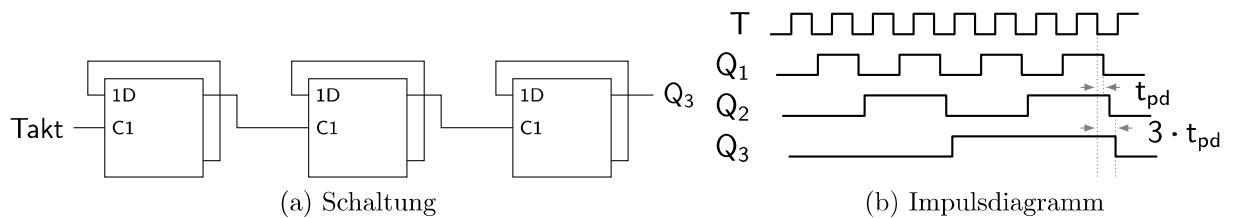


Abbildung 9.14: Asynchroner Binäruntersetzer

Teilverhältnis als $2^n : 1$ müssen durch Rückführungen entsprechend dem gewählten Teilverhältnis Zählerzustände unterdrückt werden.

Synchroner Untersteller

Wie für synchrone Systeme üblich werden alle Flipflops gleichzeitig vom gemeinsamen Takt gesetzt. Als Verzögerungszeit des gesamten Untersetzers tritt also nur die Laufzeit einer Stufe auf. Die Takteingänge können dadurch aber nicht zur Bestimmung des Teilverhältnisses mitbenutzt werden. Damit ergibt sich ein höherer Schaltungsaufwand gegenüber den asynchronen Untersetzen.

In der Abbildung ist ein einfacher synchroner Untersetzer mit Schieberegister dargestellt. Die n Stufen ergeben ein Untersetzungsvorhältnis von $n - 1$. Zusätzlich ist der Ausgang auf den Eingang zurückgeführt. Zu Beginn wird über das S-Signal das erste Flipflop gesetzt ($Q_1 = 1$) und die beiden anderen Flipflops zurückgesetzt. Danach wird mit jedem Taktimpuls die "1" jeweils um eine Stufe verschoben. Bei kreuzweiser Rückkopplung der Schieberegisterausgänge erhöht sich das Untersetzungsvorhältnis auf $2n - 1$. Das Eingangsflipflop wird dann ebenfalls zu Beginn zurückgesetzt.

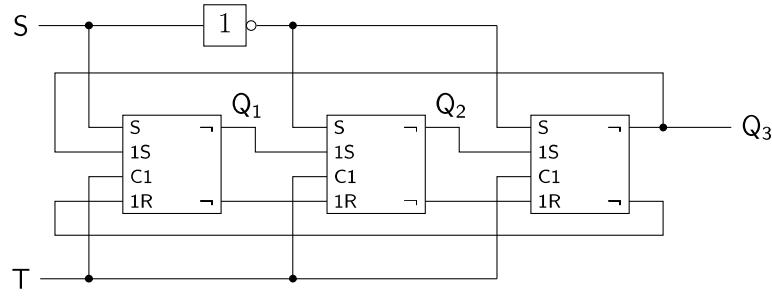


Abbildung 9.15: Synchroner Untersetzer

9.4.3 Zähler

Asynchroner Zähler

Asynchrone Zähler werden ähnlich wie asynchrone Untersetzer entworfen. Es werden entsprechend dem verwendeten Zählcode Zustände übersprungen. Die asynchronen Zähler weisen den gleichen Nachteil wie asynchrone Untersetzer auf. Trotzdem kommt dieser Zählertyp wegen seines geringen Aufwands zum Einsatz. Voraussetzung ist dabei aber, dass keine hohe Zählgeschwindigkeit erforderlich ist.

Beispiel 89. *BCD-Code-Zähler*

Zählerstand	Q ₄	Q ₃	Q ₂	Q ₁
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

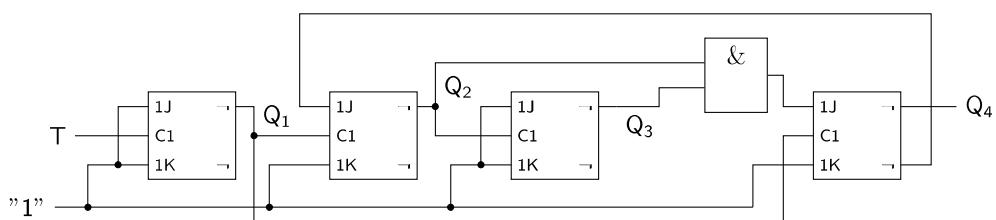


Abbildung 9.16: Asynchroner Zähler für den BCD-Code

Synchroner Zähler

Die Zählfrequenz synchroner Zähler ist unabhängig von der Anzahl der Flipflops. Es addieren sich hierbei nicht die Flipflopplaufzeiten. Der große Nachteil ist wiederum der erhöhte Schaltungsaufwand. Die gebräuchlichsten Zähler arbeiten im reinen Dualcode oder im BCD-Code.

Beispiel 90. *Synchroner 4-Bit Vorwärtszähler im Dualcode*

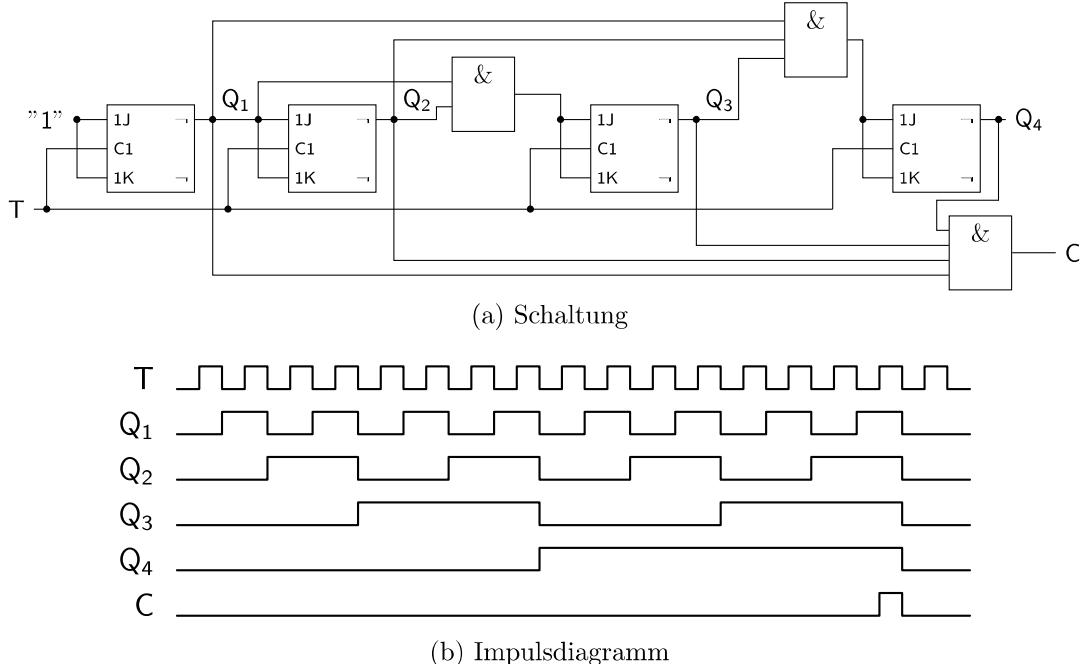


Abbildung 9.17: Synchroner dualer 4-Bit-Vorwärtszähler

Die Schaltung 9.17 ist mit RS-Flipflops realisiert. Der Entwurf des Zählers erfolgt aus der Übergangstabelle, wobei die Zustände entsprechend dem Dualcode durchlaufen werden. Zur Kaskadierung mehrerer Zählerbausteine ist ein Übertragsausgang \bar{U} vorhanden. Das \bar{U} -Signal wird auf den Takteingang des nächsten Zählerbausteins geführt, so dass die Kaskadierung asynchron erfolgt.

Synchroner Rückwärtszähler

Die Vorwärtszählerschaltung kann in einen Rückwärtszähler abgeändert werden, wenn die Q -Ausgänge mit den \bar{Q} -Ausgängen vertauscht werden.

Beispiel 91. *Synchroner dualer 4-Bit Rückwärtszähler (siehe Abbildung 9.18)*

In integrierten Zählerbausteinen wird meistens die Vorwärts- und Rückwärtszählung durch eine ODER-Schaltung kombiniert. Die Zählrichtung kann dann über einen zusätzlichen Steuereingang bestimmt werden.

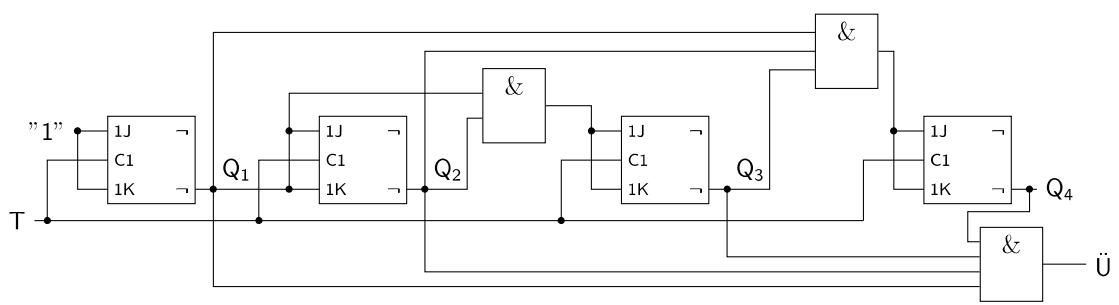


Abbildung 9.18: Synchroner dualer 4-Bit-Rückwärtszähler