

Prof. Jürgen Becker

becker@kit.edu

Karlsruher Institut für Technologie (KIT)

Institut für Technik der Informationsverarbeitung (ITIV)

Digitaltechnik

Funktionseinheiten der Digitaltechnik

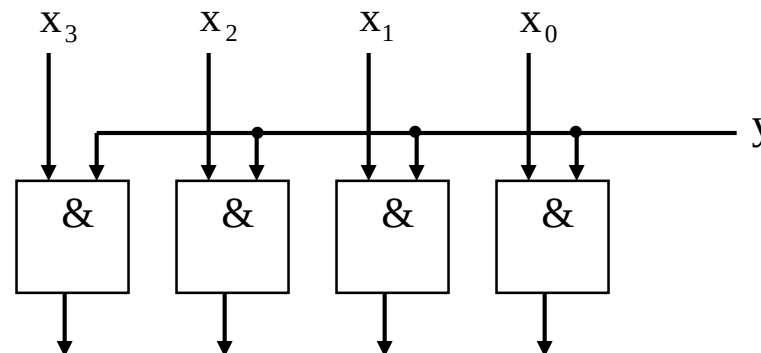
Arithmetische Grundlagen :

- **Multiplikation** einer **Dualzahl** mit einer **einstelligen Dualzahl**:

$$11_{10} \cdot 1_{10} = 1011_2 \cdot 1_2 = 1011_2$$

$$11_{10} \cdot 0_{10} = 1011_2 \cdot 0_2 = 0000_2$$

- die **einstellige Multiplikation** durch eine **UND-Verknüpfung** realisieren



Arithmetische Grundlagen:

■ Multiplikation einer Dualzahl mit einer 2er-Potenz:

Dezimal: $11_{10} \cdot 8_{10} = 88_{10}$

Dual: $1011_2 \cdot 2^3 = 1011_2 \cdot 1000_2$

$$\begin{aligned} &= (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot (1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0) \\ &= (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 1 \cdot 2^3 \\ &\quad + (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 0 \cdot 2^2 \\ &\quad + (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 0 \cdot 2^1 \\ &\quad + (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 0 \cdot 2^0 \\ &= 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 1011000_2 = 88_{10} \end{aligned}$$

- die Multiplikation einer Dualzahl mit einer 2er-Potenz 2^n bedeutet eine **Linksverschiebung** (*links-shift*) um **n-Stellen** und Nachziehen von n Nullen

Arithmetische Grundlagen:

■ Beispiel für eine Multiplikation:

$$\begin{aligned} 11_{10} \cdot 9_{10} &= 1011_2 \cdot 1001_2 \\ &= (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot (1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0) \\ &= (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 1 \cdot 2^0 \\ &\quad + (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 0 \cdot 2^1 \\ &\quad + (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 0 \cdot 2^2 \\ &\quad + (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot 1 \cdot 2^3 \\ &= \begin{array}{r} 1011_2 \\ + \quad 00000_2 \\ + \quad 000000_2 \\ + \quad 1011000_2 \\ \hline 01100011_2 \end{array} \end{aligned}$$

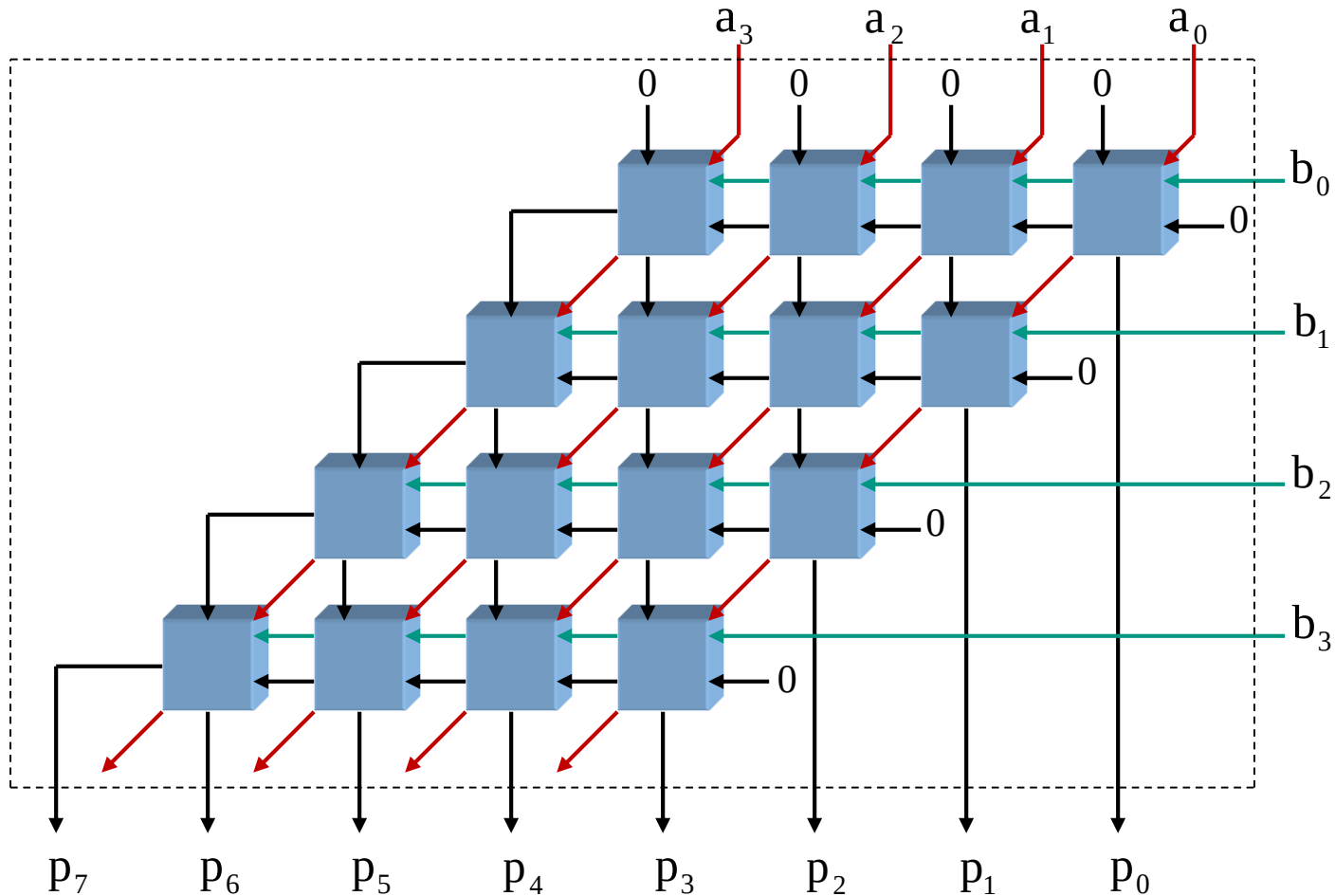
Implementierung:

- der **Multiplikand** muss mit **jeder Stelle** des **Multiplikators multipliziert** werden,
d.h. bitweise **konjunktiv verknüpft** werden
→ das **Ergebnis** also um die **entsprechende Stellenzahl** nach **links verschoben** und zu übrigen Ergebnissen **aufaddiert** werden

$$\begin{array}{rccccccc}
 A_{n-1} \dots A_0 \cdot B_{k-1} \dots B_0 = & & & & & & \\
 & & & & A_{n-1} \dots A_1 A_0 & \cdot B_0 \cdot 2^0 & \\
 + & & & & A_{n-1} \dots A_1 A_0 & \cdot B_1 \cdot 2^1 & \\
 & & & & \dots & & \\
 + & & A_{n-1} \dots A_1 A_0 & & & \cdot B_{k-1} \cdot 2^{k-1} & \\
 \hline
 = P_{n+k} & P_{n+k-1} & \dots & P_2 & P_1 & P_0 &
 \end{array}$$

Implementierung:

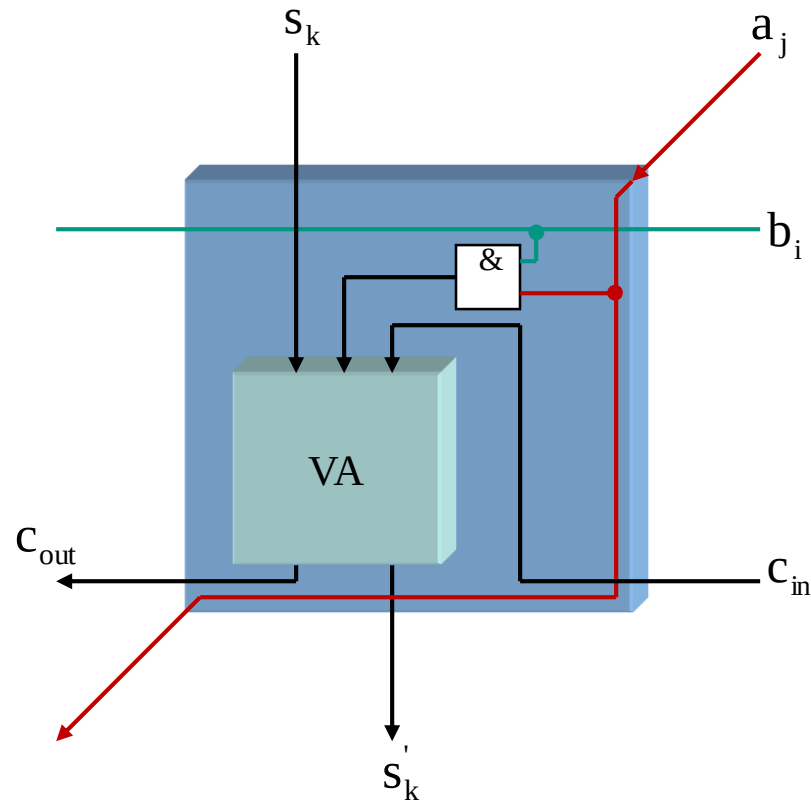
■ Beispiel für einen 4-Bit Multiplizierer:



Multiplizierer

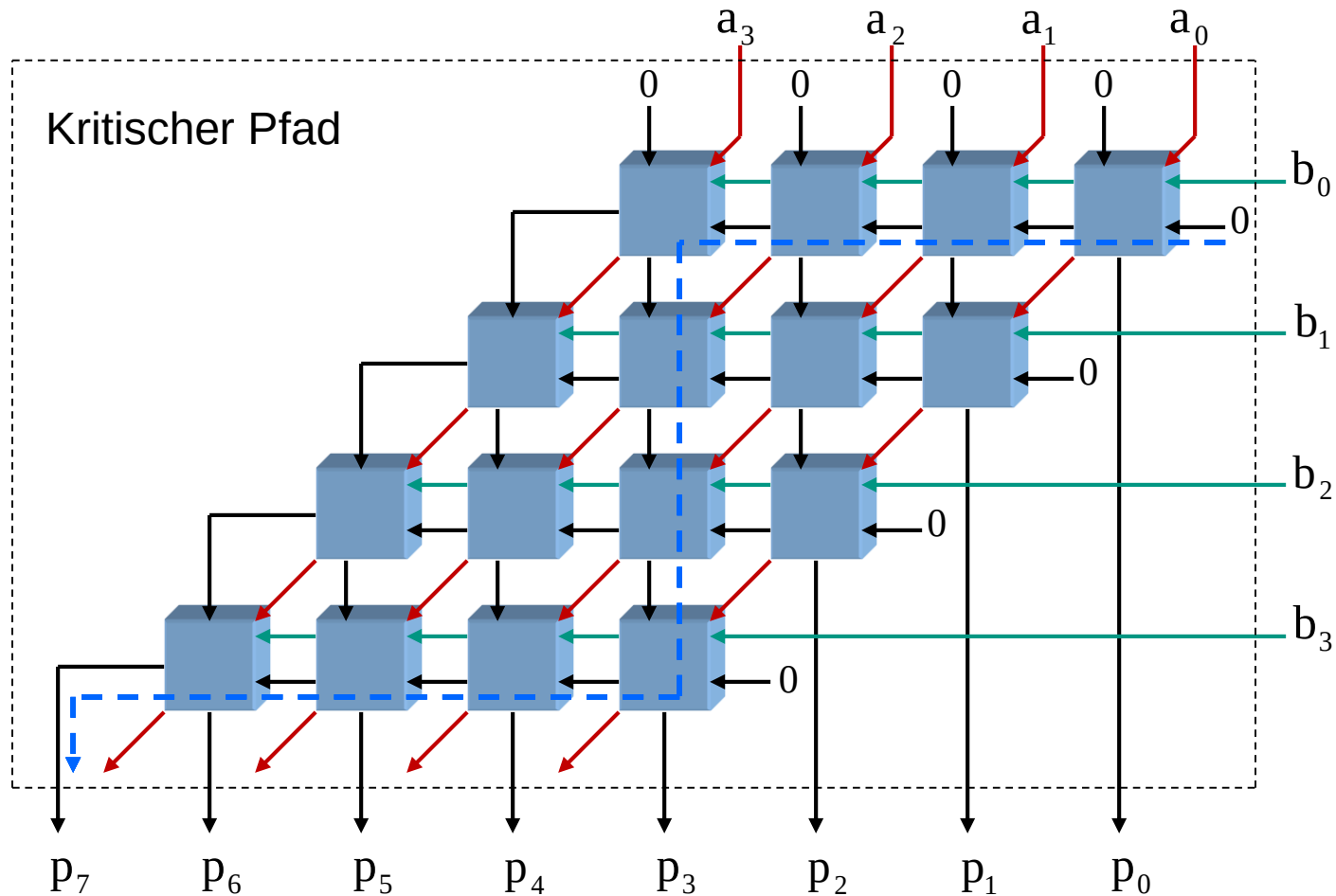
Implementierung:

- Aufbau einzelner Zellen:



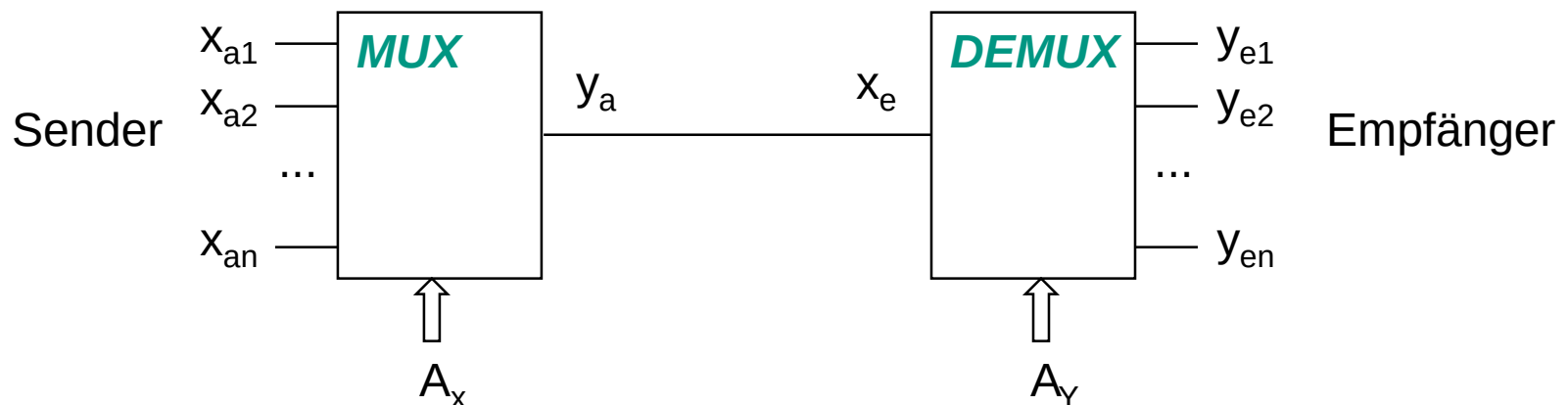
Implementierung:

- **Problem:** sehr lange Laufzeiten, im Beispiel $3n-2$



Multiplexer und Demultiplexer:

- **Verbindungen** zwischen **Systemgruppen** sind platz- und kostenintensiv
- Besonders bei großer räumlicher Distanz und großer Leitungsanzahl
- Falls die zeitlichen Anforderungen es erlauben, kann auf eine **serielle Übertragung** zurückgegriffen werden
- Die **Steuergrößen A_x und A_y** wählen den **aktiven Eingang** oder **Ausgang**



(De-)Multiplexer

Beispiel:

$$X_a = (x_{a4}, x_{a3}, x_{a2}, x_{a1})$$

$$Y_e = (y_{e4}, y_{e3}, y_{e2}, y_{e1})$$

$$r = \lceil \lg 4 \rceil = 2$$

$$A_x = (a_{x2}, a_{x1})$$

$$A_y = (a_{y2}, a_{y1})$$

2 Steuerleitungen nötig

4:1 Multiplexer

a_{x2}	a_{x1}	y_a
0	0	x_{a1}
0	1	x_{a2}
1	0	x_{a3}
1	1	x_{a4}

1:4 Demultiplexer

a_{y2}	a_{y1}	y_{e1}	y_{e2}	y_{e3}	y_{e4}
0	0	x_e	0	0	0
0	1	0	x_e	0	0
1	0	0	0	x_e	0
1	1	0	0	0	x_e

(De-)Multiplexer

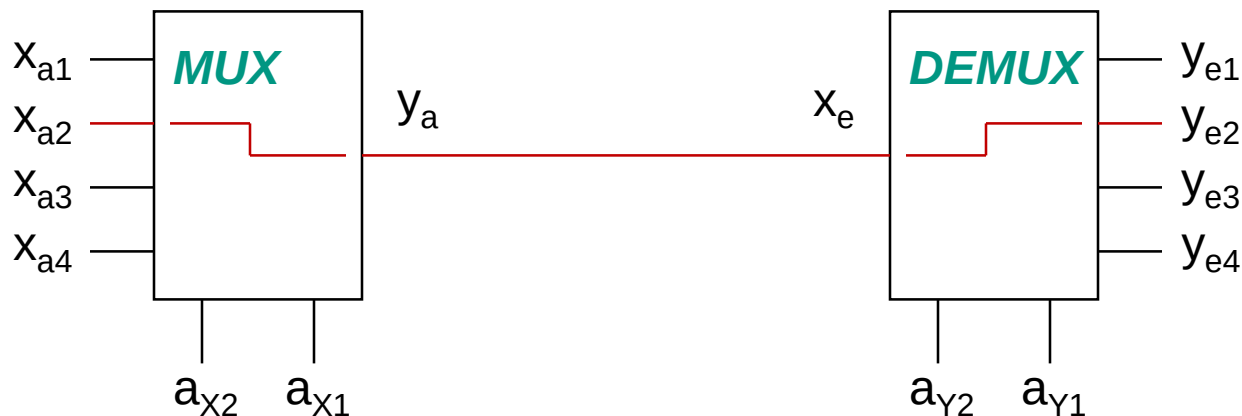
Beispiel:

4:1 Multiplexer

a_{x2}	a_{x1}	y_a
0	0	x_{a1}
0	1	x_{a2}
1	0	x_{a3}
1	1	x_{a4}

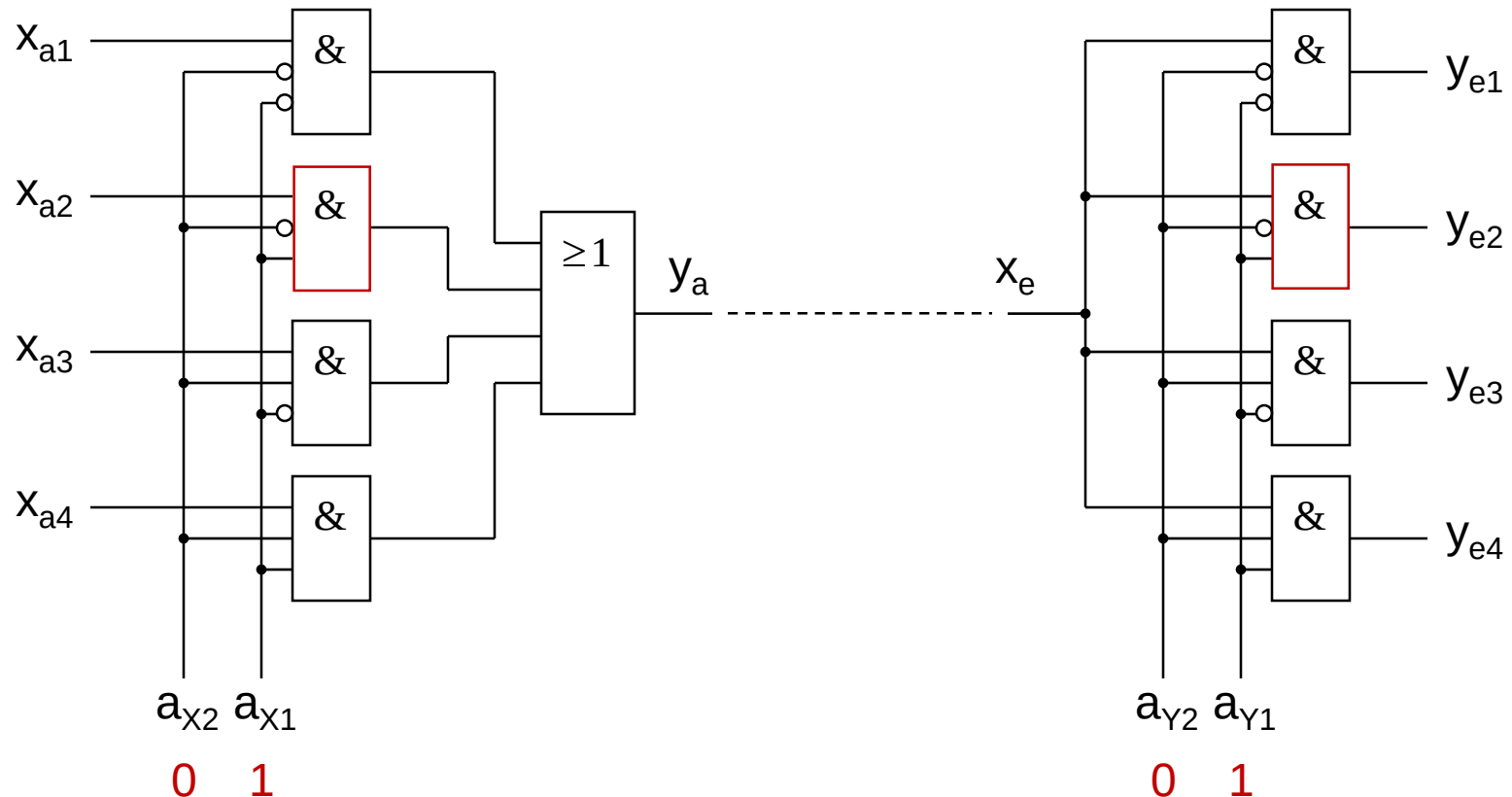
1:4 Demultiplexer

a_{y2}	a_{y1}	y_{e1}	y_{e2}	y_{e3}	y_{e4}
0	0	x_e	0	0	0
0	1	0	x_e	0	0
1	0	0	0	x_e	0
1	1	0	0	0	x_e



Typische Schaltnetze für Multiplexer und Demultiplexer:

- Die **Minterme** der **Adressvektoren** A_x und A_y geben immer **genau ein UND-Gatter „frei“**

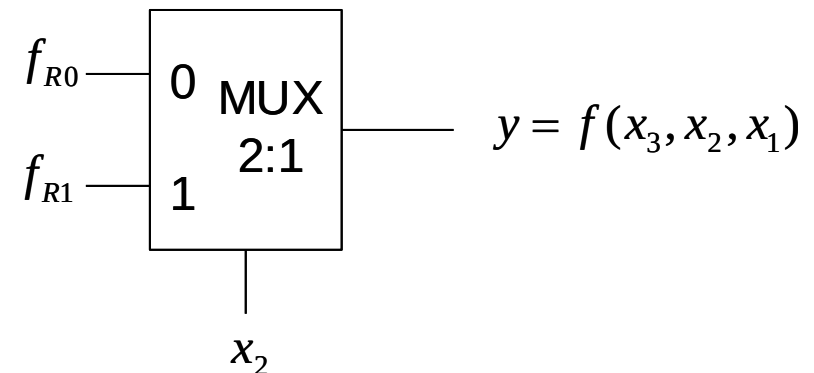


Anwendung:

- mit **Multiplexern** kann man noch **viele andere Aufgaben** lösen
 - z.B. **Strukturausdrücke** nach dem **Entwicklungssatz** sind **direkt umsetzbar**
 - **Auswahl** zwischen **zwei Werten** eines **Literals** übernimmt **2:1 Multiplexer**

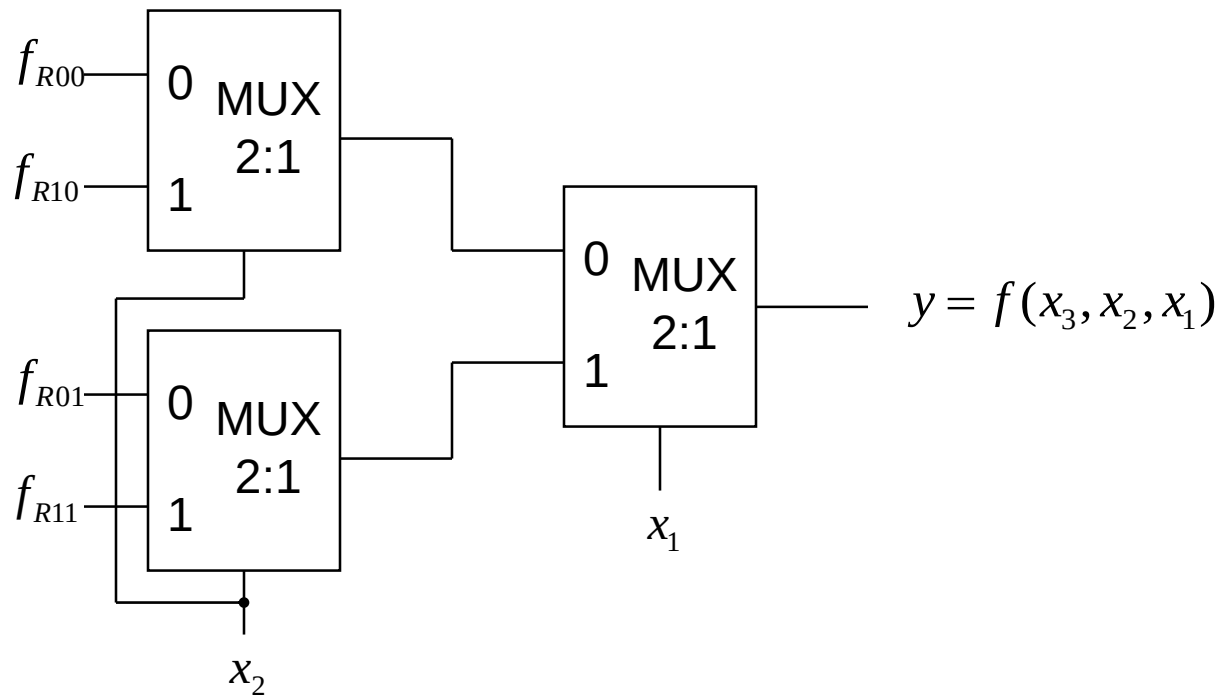
Beispiel:

$$\begin{aligned}y &= f(x_3, x_2, x_1) \\&= (\overline{x_2} \& f(x_3, 0, x_1)) \vee (x_2 \& f(x_3, 1, x_1)) \\&= (\overline{x_2} \& f_{R0}) \vee (x_2 \& f_{R1})\end{aligned}$$



Entwicklung nach zwei Variablen:

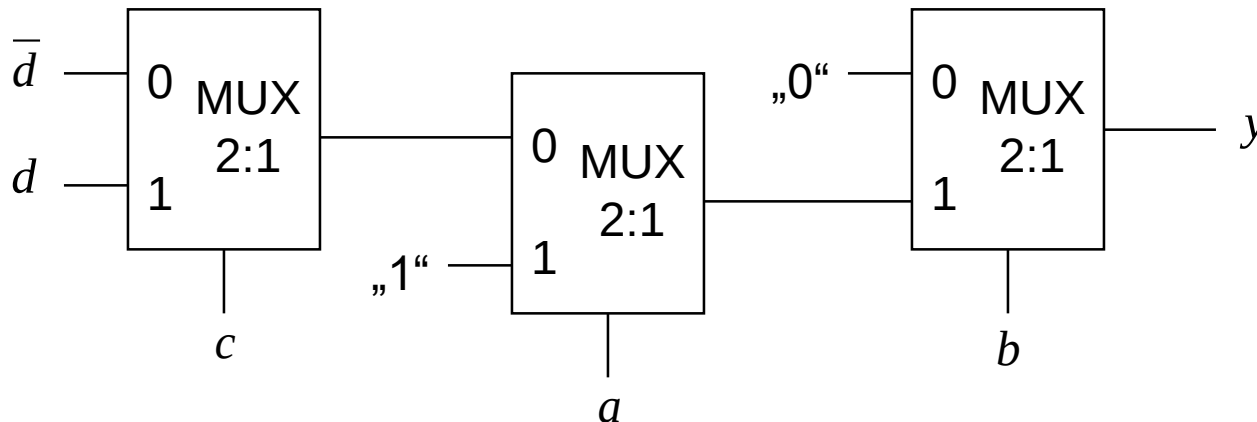
→ Realisierung mit 2:1 Multiplexern



Bei der **Realisierung** einer **Funktion** mit **n Literalen** über **Multiplexer** muss **nach n-1 Variablen entwickelt** werden

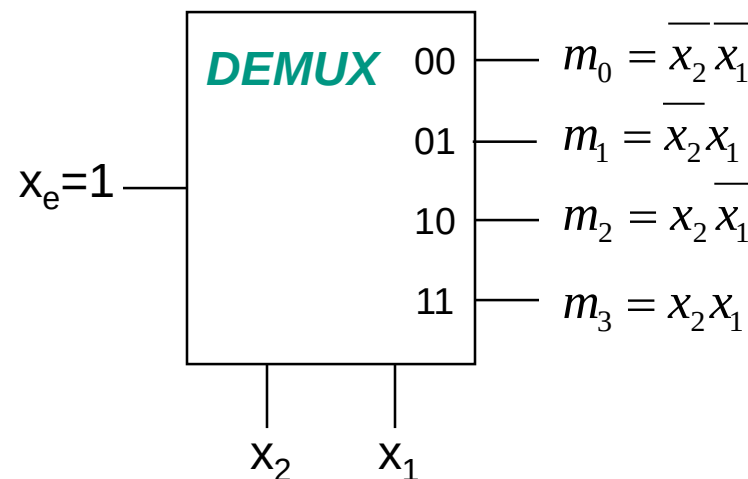
Beispiel:

$$\begin{aligned}y &= b \& a \vee (b \& (c \equiv d)) \\&= b \& [a \vee (c \equiv d)] \vee \bar{b} \& 0 \\&= b \& [a \& 1 \vee \bar{a} \& (c \equiv d)] \vee \bar{b} \& 0 \\&= b \& [a \& 1 \vee \bar{a} \& (c \& d \vee \bar{c} \& \bar{d})] \vee \bar{b} \& 0\end{aligned}$$



Anwendung:

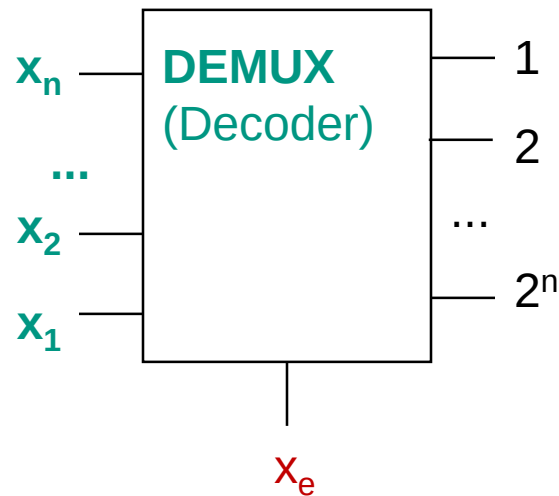
- Demultiplexer können zur **Ansteuerung** von **Steuerleitungen** von **Speicherzellen** verwendet werden
- An diesem Beispiel wird auch die Bezeichnung **Adressvektor** für die Steuerleitungen klar
- Liegt der **Wert “1”** an dem Eingang des **Demultiplexers**, so repräsentieren die **Ausgänge** sämtliche **Minterme** des **Adressvektors**



Anwendung:

- In der Anwendung als **Decoder** einer Adresse wird der **Demultiplexer** auch gerne als Codewandler so gezeichnet:

Adressleitungen



Enable-Variable

Anwendungsfälle:

Auch bei **Schaltwerken** gibt es einige Beispiele, die in **unterschiedlichsten Anwendungsfällen** eingesetzt werden

- Anzahl von Ereignissen oder Aktionen festzuhalten
- Informationen zeitlich oder räumlich zu ordnen oder umzusetzen
- Spalten- bzw. Zeilenselektion in einem Speicher
- Tastaturmatrix

Zähler

- **Zähler-Schaltwerke** dienen dazu auf einer speziellen Signalleitung beobachtete **Ereignisse** zu **zählen**
- Das Zählen entspricht einer **Abbildung** in den **Zustandsraum**, sodass ein **Zählcode** entsteht
- Wie bei Mechanischen Zählern können noch **weitere Eingriffmöglichkeiten** eingebaut werden:
 - Überlaufmeldung
 - Nullstellungsmeldung
 - Rückstellmöglichkeit auf 0

Beispiel:

- Es soll ein Zähler als Schaltwerk realisiert werden, der...
 - **dual zyklisch** von 0 bis 7 zählt
 - eine **Rückstellmöglichkeit** auf 0 besitzt
 - bei dem Wert 7 das **Überlaufsignal** auf 1 setzt
 - auf den **Wechsel** der **Zählervariable Z** von **0 auf 1** reagiert
- Zur **Realisierung** sollen **JK-Flipflops** mit **0/1-Flankensteuerung** verwendet werden

Beispiel:

■ Eingangsvariable:

$N = 1$: Rückstellen auf 0

$N = 0$: sonst

■ Ausgangsvariable:

$\ddot{U} = 1$: Zählerstand ist 7

$\ddot{U} = 0$: sonst

■ Bei Zählern wird die **Zählervariable Z** häufig nicht als Eingangsvariable sondern direkt **als Taktsignal** angeschlossen

■ Das **Zählereignis** „Wechsel von 0 auf 1“ kann so direkt mit einem **synchronen Schaltwerk** mit **positiver Flankensteuerung** realisiert werden

Schaltwerkstabelle:

- Die **Kodierung** der **Zustände** ist die **duale Zahlendarstellung**
- Bei **N = 1** ist der **Folgezustand** auf jeden Fall **0** (Rücksetzen)
- Die **Übertragungsfunktion** kann direkt abgelesen werden:

$$\ddot{U} = q_3^v \& q_2^v \& q_1^v$$

Q^v			X^v	Q^{v+1}			Y^v
q_3	q_2	q_1	N	q_3	q_2	q_1	$\ddot{U}$
0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	1	1	0
0	1	0	1	0	0	0	0
0	1	1	0	1	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	1	0	1	0
1	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0
1	0	1	1	0	0	0	0
1	1	0	0	1	1	1	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	1

Ansteuerfunktionen:

- durch **Ablesen** oder mit Hilfe von **Symmetriediagrammen** erhält man:

$$K_1 = 1$$

$$K_2 = N \vee q_1^v$$

$$K_3 = N \vee q_2^v q_1^v$$

$$J_1 = \overline{N}$$

$$J_2 = \overline{N} q_1^v$$

$$J_3 = \overline{N} q_2^v q_1^v$$

Q ^v			X ^v	Q ^{v+1}			Y ^v						
q ₃	q ₂	q ₁	N	q ₃	q ₂	q ₁	Ü	J ₃	K ₃	J ₂	K ₂	J ₁	K ₁
0	0	0	0	0	0	1	0	0	-	0	-	1	-
0	0	0	1	0	0	0	0	0	-	0	-	0	-
0	0	1	0	0	1	0	0	0	-	1	-	-	1
0	0	1	1	0	0	0	0	0	-	0	-	-	1
0	1	0	0	0	1	1	0	0	-	-	0	1	-
0	1	0	1	0	0	0	0	0	-	-	1	0	-
0	1	1	0	1	0	0	0	1	-	-	1	-	1
0	1	1	1	0	0	0	0	0	-	-	1	-	1
1	0	0	0	1	0	1	0	-	0	0	-	1	-
1	0	0	1	0	0	0	0	-	1	0	-	0	-
1	0	1	0	1	1	0	0	-	0	1	-	-	1
1	0	1	1	0	0	0	0	-	1	0	-	-	1
1	1	0	0	1	1	1	0	-	0	-	0	1	-
1	1	0	1	0	0	0	0	-	1	-	1	0	-
1	1	1	0	0	0	0	1	-	1	-	1	-	1
1	1	1	1	1	0	0	1	-	1	-	1	-	1

Blockschema des Zählers:

Ansteuerfunktionen:

$$J_1 = \overline{N}$$

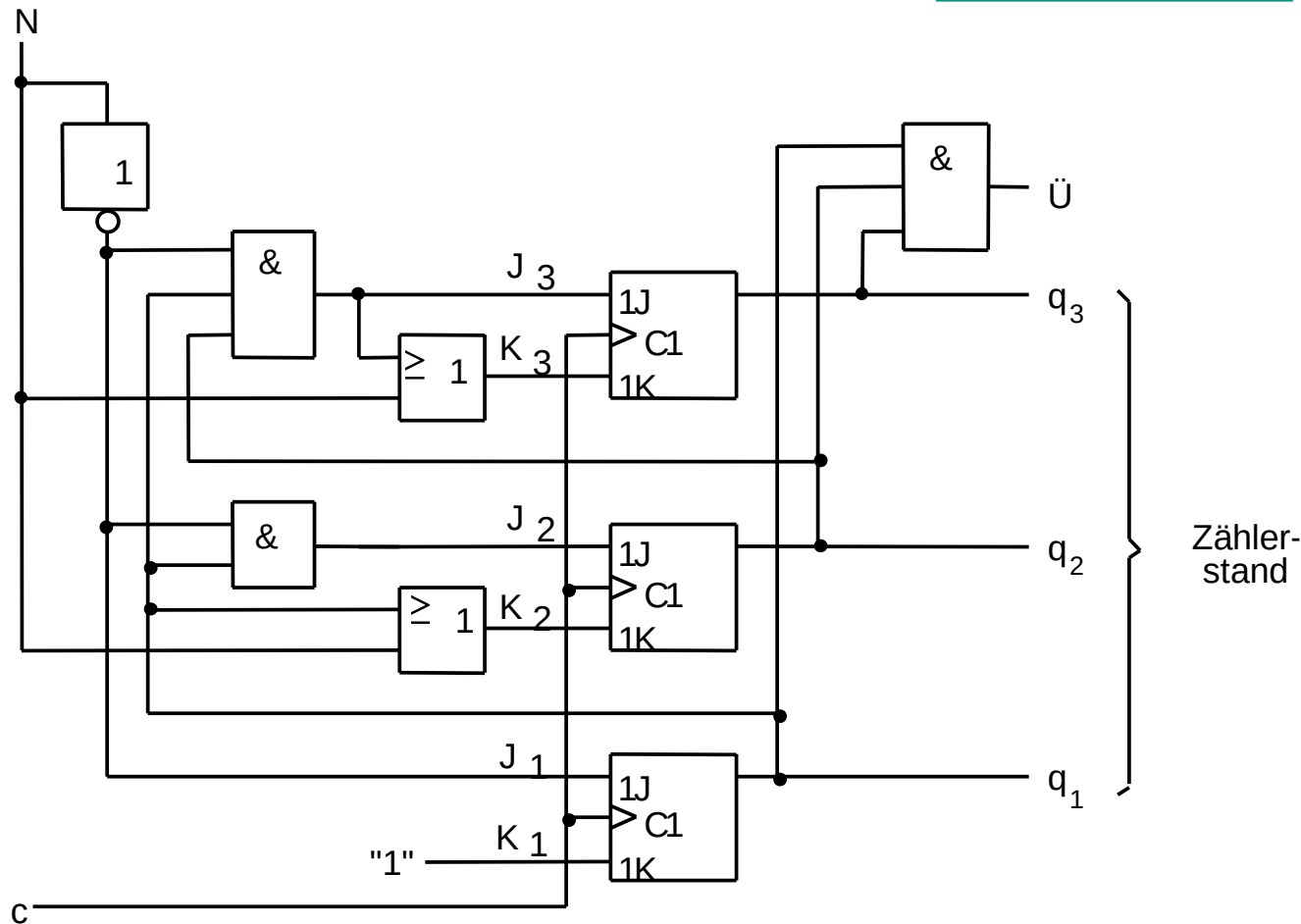
$$J_2 = \overline{N}q_1^v$$

$$J_3 = \overline{N}q_2^vq_1^v$$

$$K_1 = 1$$

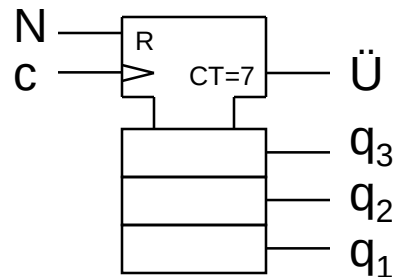
$$K_2 = N \vee q_1^v$$

$$K_3 = N \vee q_2^vq_1^v$$



Symboldarstellung:

- Um bei Zählern von **Realisierungsdetails** zu **abstrahieren** führt man **Blocksymbole** ein
- Das **Symbol** besteht aus einem **Daten-** und einem **Steuerteil**
- Für das besprochene Beispiel sieht der **Zähler** so aus:



Eigenschaften:

- Hier noch die **wichtigsten Merkmale** von **Zählern**:
 - **Art des Zählcodes**
→ Dualzahlen, 1-aus-n-Code, BCD-Code,...
 - **Art des Zählereignisses**
→ $0 \rightarrow 1$, $1 \rightarrow 0$, Auftreten von Werten, ...
 - **Steuersignal** für **Zählen/Anhalten**
 - **Steuersignal** für **Zählrichtung** (aufwärts/abwärts)
 - **Synchrone** oder **asynchrone Rückstellmöglichkeit**
 - **Steuersignal** für Zählen/Laden eines **Anfangswertes**

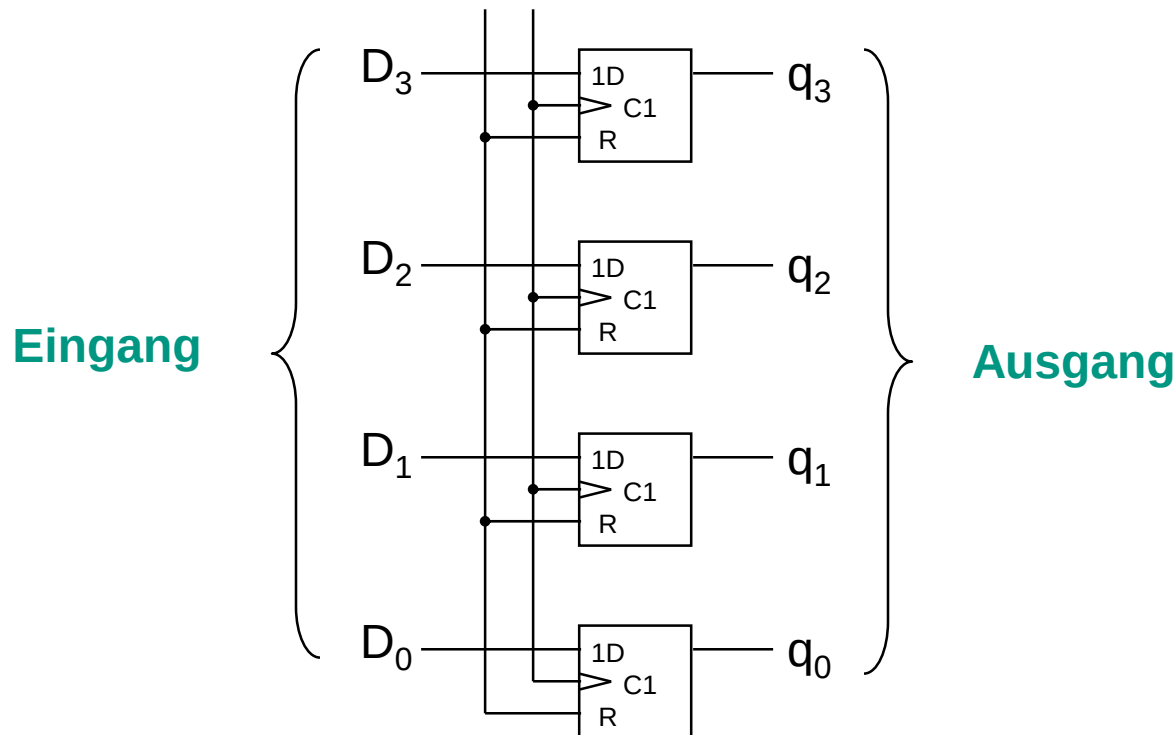
Register:

- Zur Darstellung von Informationen genügen nur selten einzelne Bits
- Daher ist es zweckmäßig **mehrere FlipFlops zu einem Verbund** zusammenzuschließen
 - Sie können so mit **gemeinsamem Takt** und Rücksetzmöglichkeiten versorgt werden
- So ein **FlipFlop-Verbund** wird auch **Register** genannt

Beispiel:

- **Register**, das ein **4-bit Datenwort** bei ansteigender Taktflanke speichert und sich asynchron auf 0 rücksetzen lässt:

Rücksetzen *Takt*



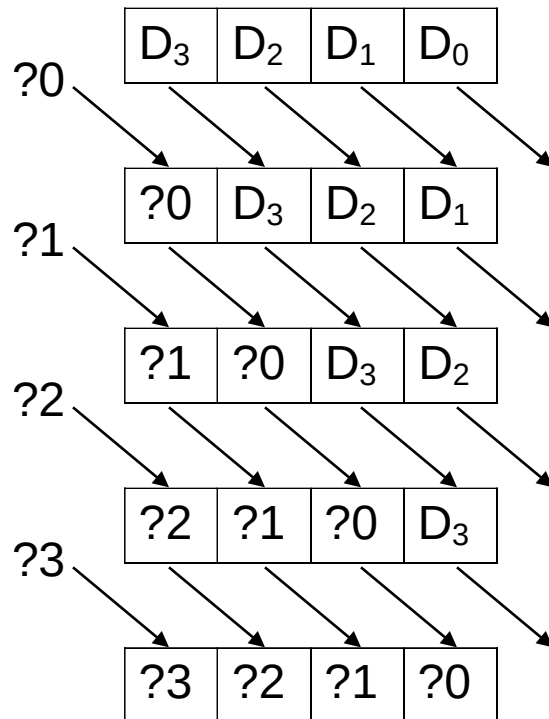
Spezialregister:

- **Datentechnisch** sind die **Speicherzellen** eines **Registers** im allgemeinen **nicht miteinander verbunden**
- **Register** werden jedoch oft mit **Funktionen** ausgestattet, die auf **mehreren Speicherzellen operieren**
- Besonders **Funktionen**, bei denen sich nur **benachbarte Speicherzellen** beeinflussen, lassen sich **effizient** im **Register realisieren**
- Ein einfach zu realisierendes Spezialregister ist das **Schieberegister**
- Beim **Schieberegister** können **alle Speicherinhalte** um **eine Zelle weitergeschoben** werden
- Die **Auslösung** dieses Vorgangs erfolgt **durch ein Taktsignal**

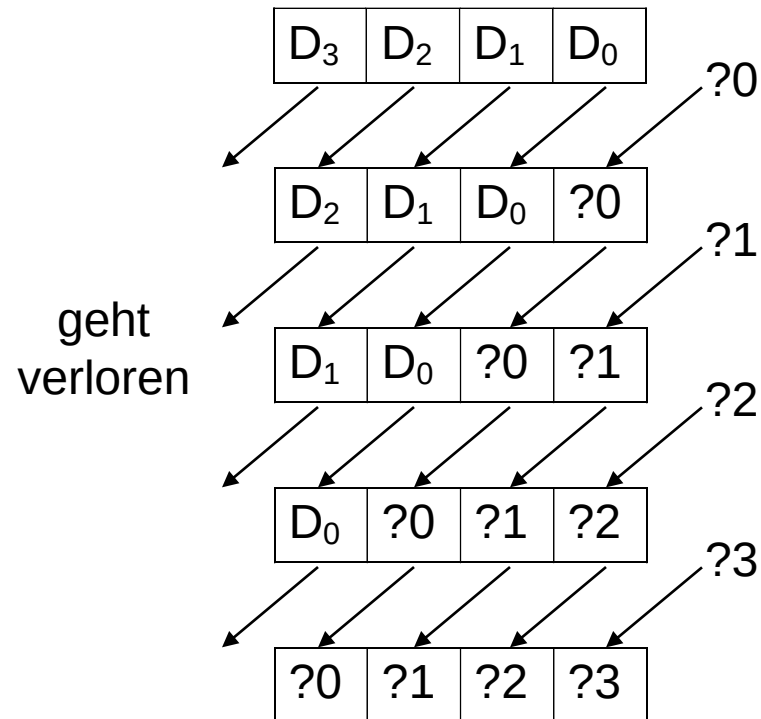
Beispiel:

■ 4-Bit Schieberegister:

Rechtsschieben



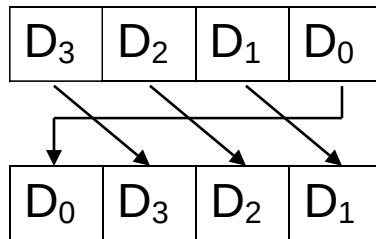
Linksschieben



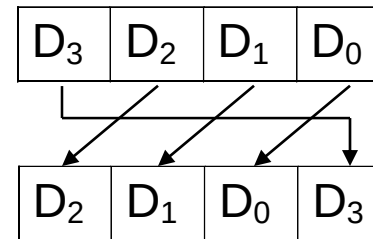
Schieberegister:

- Beim **Schieben** geht jeweils das in **Schieberichtung** voranstehende **Datenbit verloren**
- Das **letzte Datenbit** wird mit einem **Inhalt gefüllt**, der von den **Randbedingungen** dieser Zelle abhängt
- Damit die **Daten nicht verloren gehen**, kann man **zirkulär rotieren** und den Inhalt des „rausgeschobenen“ Bits auf der anderen Seite einspeisen

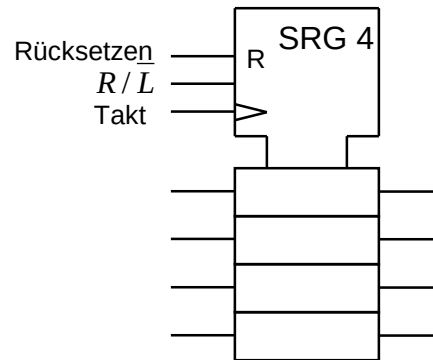
Rechtsschieben



Linksschieben



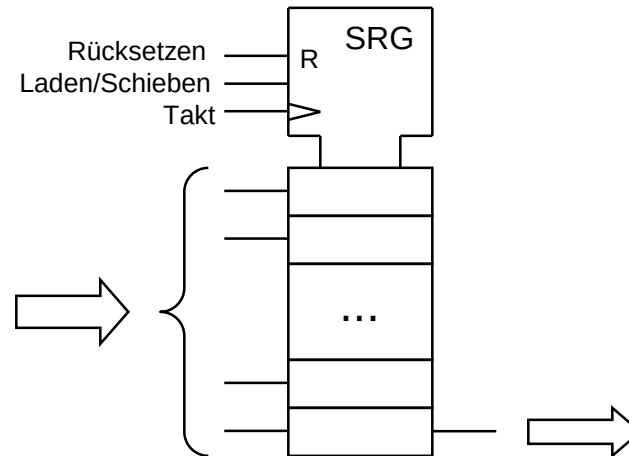
Blockschaltbild:



Anwendung:

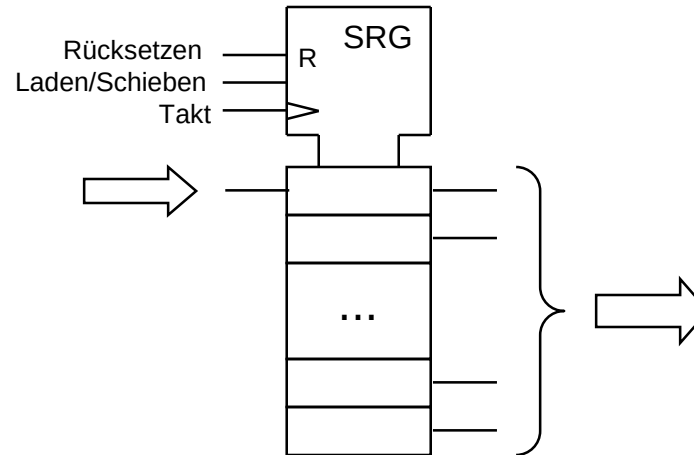
- **Schieberegister** werden auch verwendet, um **Datenbitgruppen** von **räumlichen Folgen** in eine **zeitliche Folge umzuwandeln**
- Dies wird **Parallel/Serienwandlung** bzw. **Serien/Parallelwandlung** genannt

Parallel/Serienwandlung:



- Bei der **Parallel/Serienwandlung** werden z.B. die **Daten ($D_n, \dots, D_1, D_0$)** **parallel** in das **Register** geladen
- Danach können sie **seriell** in der **zeitlichen Reihenfolge ausgegeben** werden

Serien/Parallelwandlung:



- Bei der **Serien/Parallelwandlung** werden die **Daten in zeitlicher Folge** in das **Register** geschoben
- Danach können sie **“gebündelt ausgelesen”** werden