technische universität
dortmund

fakultät für informatik

Rechnerstrukturen, Teil 1

Vorlesung 4 SWS WS 19/20

Prof. Dr. Jian-Jia Chen
Fakultät für Informatik – Technische Universität Dortmund
jian-jia.chen@cs.uni-dortmund.de
http://ls12-www.cs.tu-dortmund.de

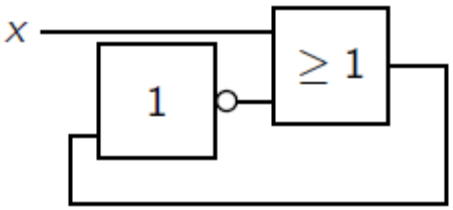
Übersicht

1. Organisatorisches ✓
2. Einleitung ✓
3. Repräsentation von Daten ✓
4. Boolesche Funktionen und Schaltnetze ✓
5. Rechnerarithmetik ✓
6. Optimierung von Schaltnetzen ✓
7. Programmierbare Bausteine ✓
8. **Synchrone Schaltwerke**

8. Synchrone Schaltwerke

8.1 Einleitung

Sequenzielle Schaltungen

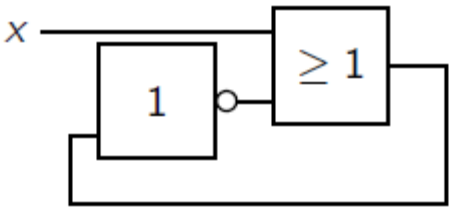


Beobachtung

- Das ist **kein** Schaltnetz.
- Es ist eine ”baubare“ Schaltung.
- Was passiert in dieser Schaltung?

8.1 Einleitung

Eine konkrete sequenzielle Schaltung

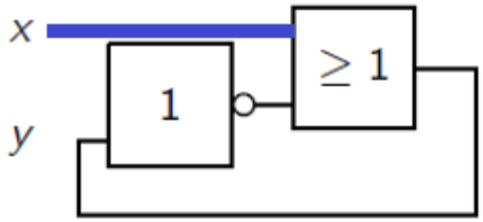


Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

8.1 Einleitung

Eine konkrete sequenzielle Schaltung

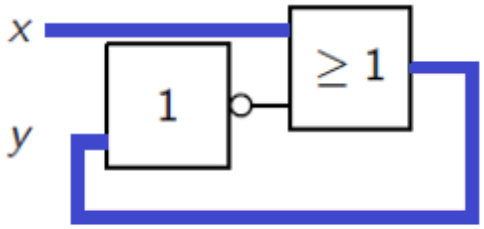


Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

8.1 Einleitung

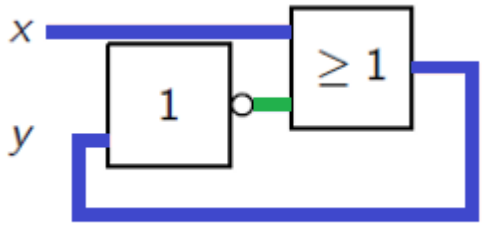
Eine konkrete sequenzielle Schaltung



Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

Eine konkrete sequenzielle Schaltung

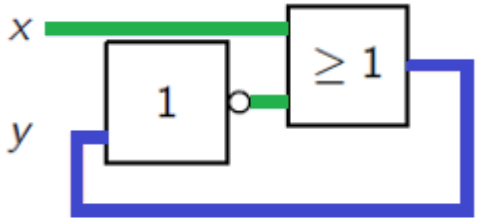


Was passiert in dieser Schaltung? Offensichtlich stabil.

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

8.1 Einleitung

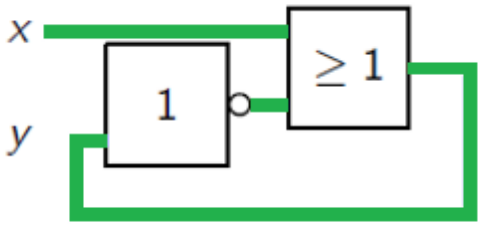
Eine konkrete sequenzielle Schaltung



Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

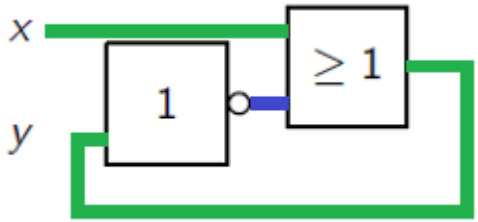
Eine konkrete sequenzielle Schaltung



Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

Eine konkrete sequenzielle Schaltung

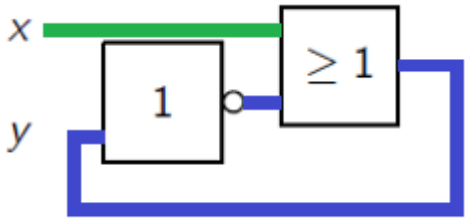


Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

8.1 Einleitung

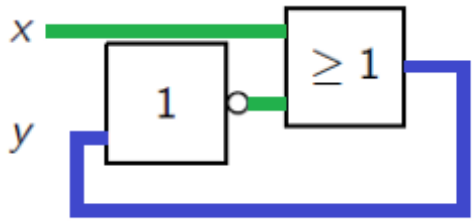
Eine konkrete sequenzielle Schaltung



Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

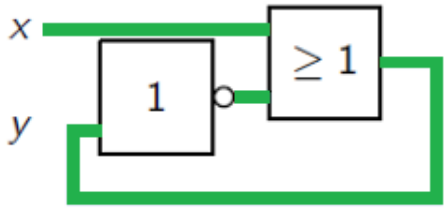
Eine konkrete sequenzielle Schaltung



Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

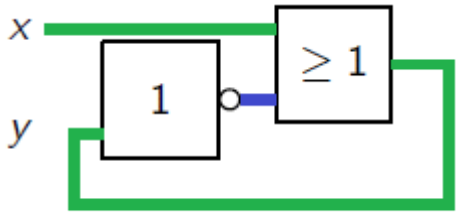
Eine konkrete sequenzielle Schaltung



Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

Eine konkrete sequenzielle Schaltung



Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

Beobachtung — und immer so weiter. . .

natürlich — in der Realität viel schneller

darum heißt die Schaltung **Flimmerschaltung**

Bewertung des Effekts

Unkontrolliertes Flimmern ist sehr unschön.

Also Kreise konsequent verbieten?

Wozu können Kreise gut sein?

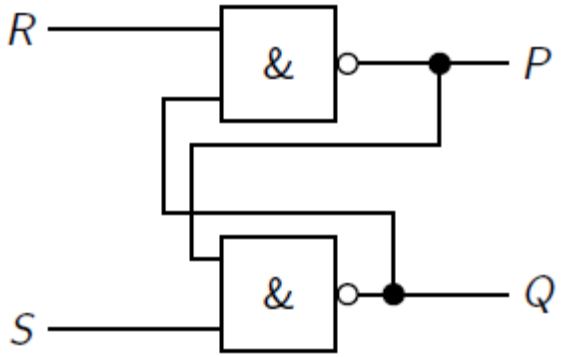
Beobachtung	Ausgänge werden zu Eingaben. . .
etwas anders	Man kann schon Berechnetes noch einmal ”sehen“.
Einsicht	Das realisiert so etwas wie Speicher.

8. Synchrone Schaltwerke

1. Einleitung ✓
2. **Bistabile Kippstufe**
3. Automaten
4. Synchrone Schaltwerke
5. Serienaddierwerke
6. Speicher & Schieberegister
7. Takt

8.2 Bistabile Kippstufe (engl. Flip-Flop)

Ein zweites Beispiel

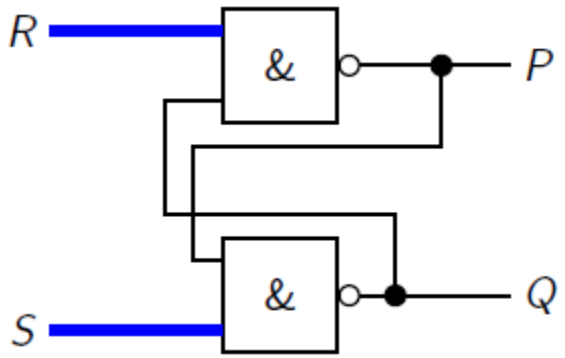


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0				
0	1				
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

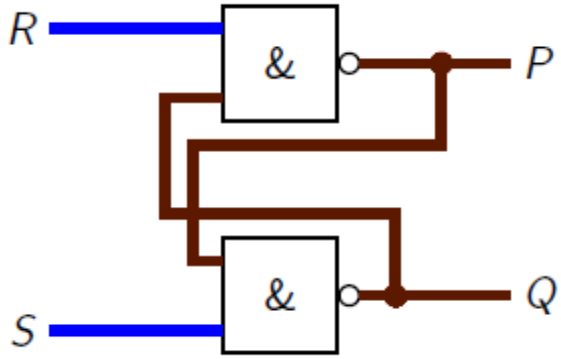


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0				
0	1				
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

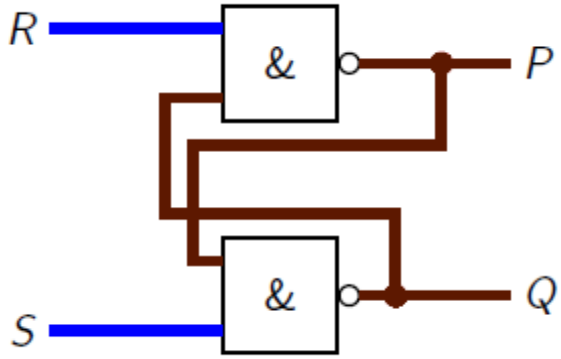


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1		
0	1				
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

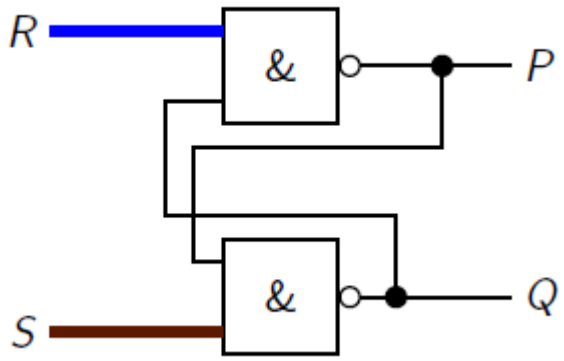


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1				
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

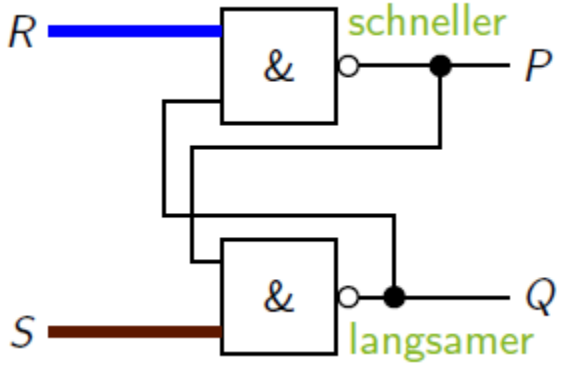


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1				
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

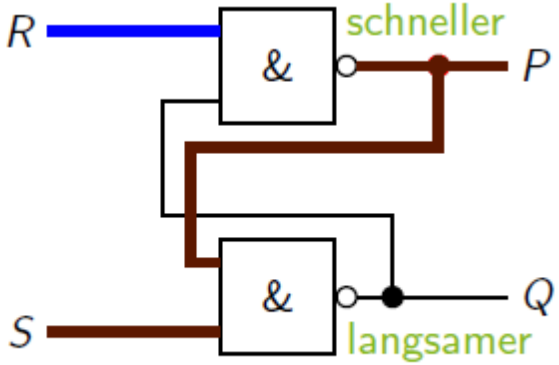


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1				
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

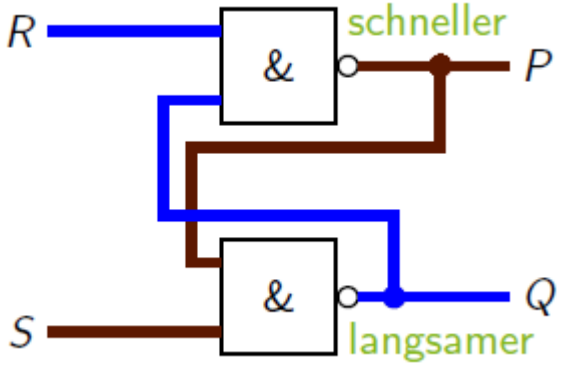


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1				
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

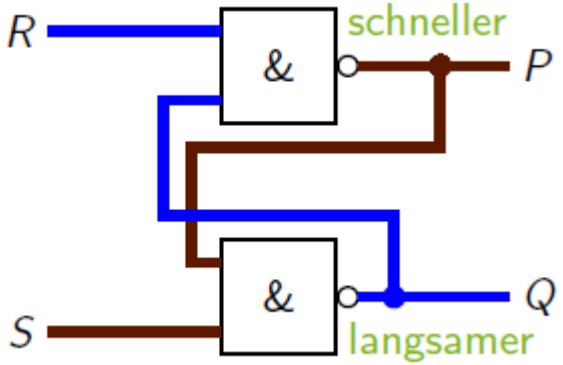


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0		
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

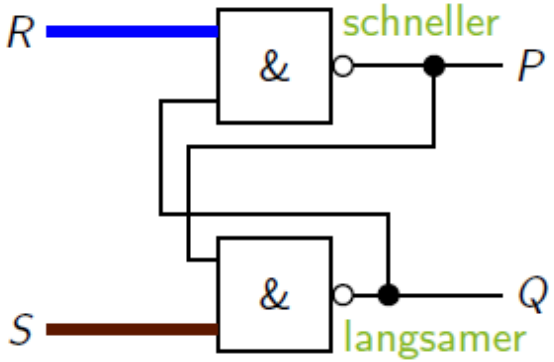


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

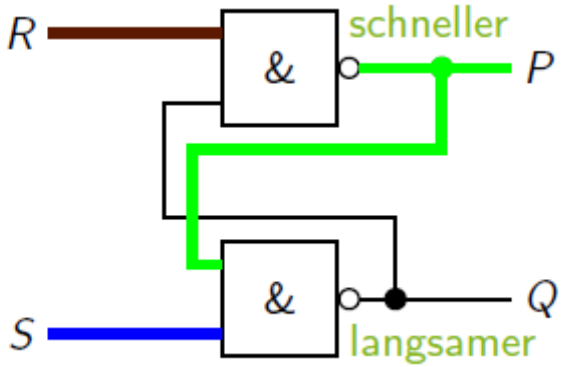


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

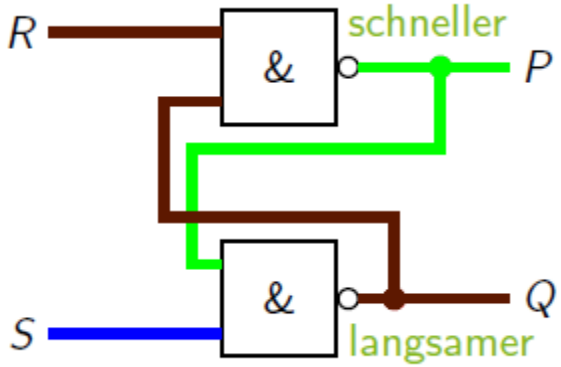


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0				
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

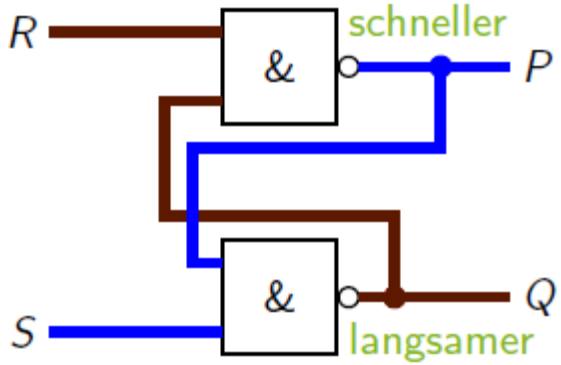


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1		
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

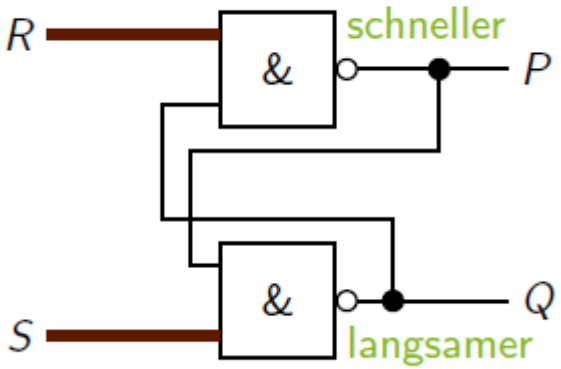


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

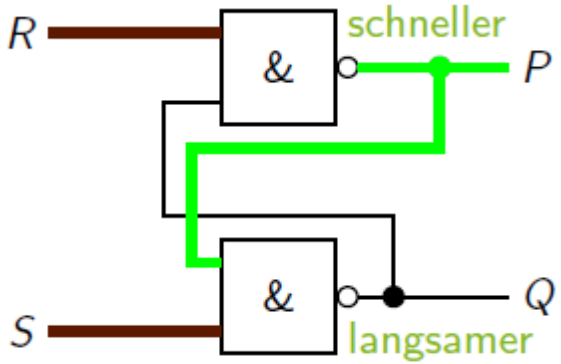


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

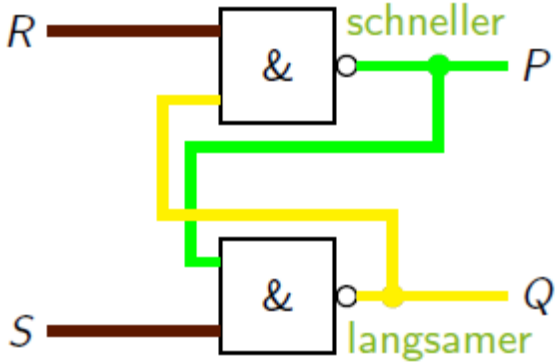


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1				

8.2 Bistabile Kippstufe

Ein zweites Beispiel

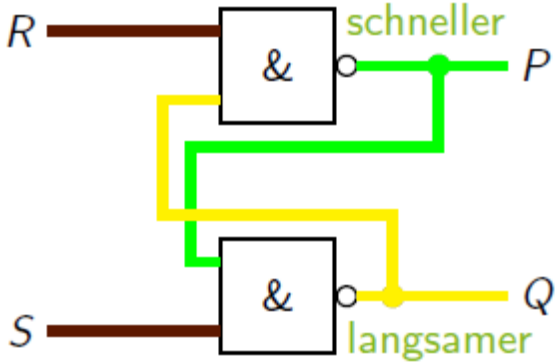


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1	$\overline{Q_t}$	Q_t		

8.2 Bistabile Kippstufe

Ein zweites Beispiel

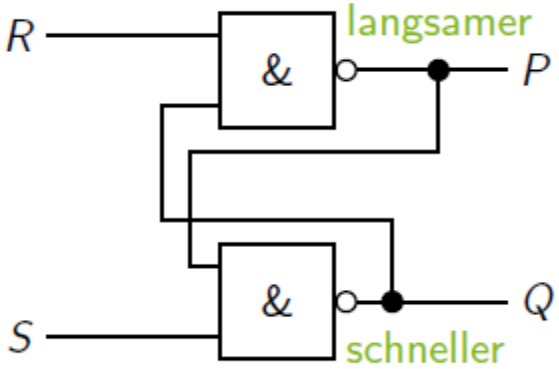


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1	$\overline{Q_t}$	Q_t	$\overline{Q_t}$	Q_t

8.2 Bistabile Kippstufe

Ein zweites Beispiel

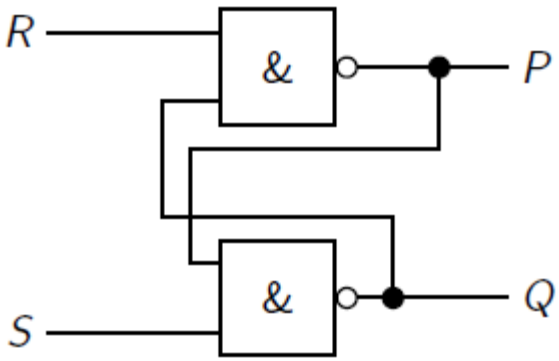


x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	$\overline{P}_t$	1	0
1	0	0	1	0	1
1	1	P_t	$\overline{P}_t$	P_t	$\overline{P}_t$

8.2 Bistabile Kippstufe

Bi-stabile NAND-Kippstufe



Anmerkung

heißt auch Latch

positiv kippt, flimmert nicht

negativ Verhalten hängt von Schaltzeiten der beiden Gatter ab

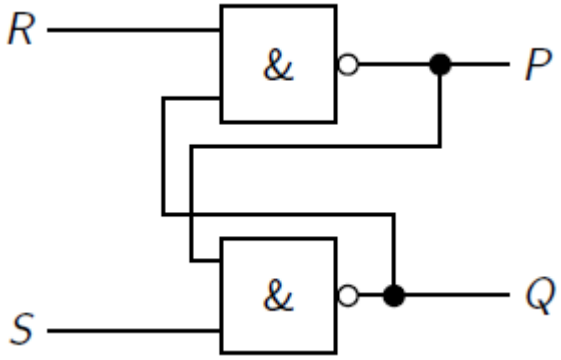
genauer beobachtet Verhalten hängt manchmal von Schaltzeiten der beiden Gatter ab

8.2 Bistabile Kippstufe

Analyse der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1	$\overline{Q_t}$	Q_t	$\overline{Q_t}$	Q_t

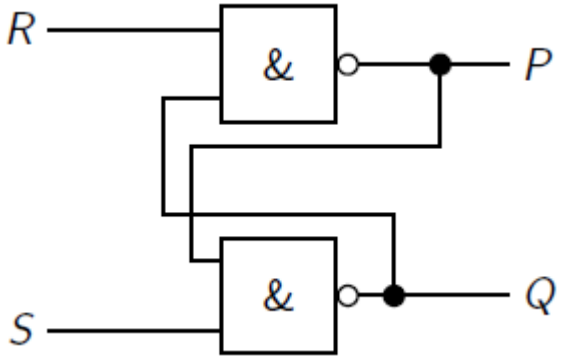


8.2 Bistabile Kippstufe

Analyse der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1	$\overline{Q_t}$	Q_t	$\overline{Q_t}$	Q_t



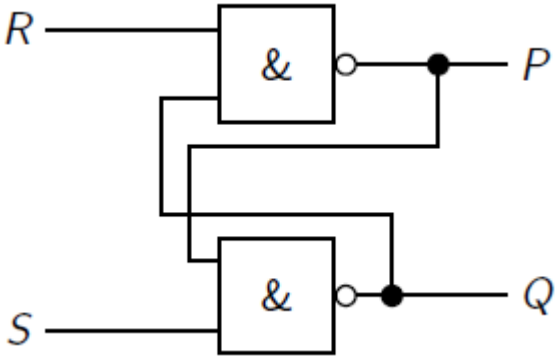
$$P_{t+2\Delta} = \overline{RQ_{t+\Delta}}$$

8.2 Bistabile Kippstufe

Analyse der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1	$\overline{Q_t}$	Q_t	$\overline{Q_t}$	Q_t



$$P_{t+2\Delta} = \overline{RQ_{t+\Delta}} = \bar{R} \vee \overline{Q_{t+\Delta}} = \bar{R} \vee \overline{\overline{SP_{t+\Delta}}} = \bar{R} \vee SP_{t+\Delta}$$

$$= \bar{R} \vee S\overline{RQ_t} = \bar{R} \vee S(\bar{R} \vee \overline{Q_t}) = \bar{R} \vee S\bar{R} \vee S\overline{Q_t}$$

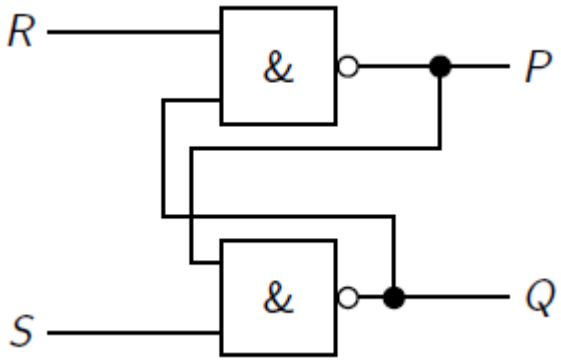
$$= \bar{R} \vee S\overline{Q_t}$$

8.2 Bistabile Kippstufe

Analyse der bi-stabilen NAND-Kippstufe

2. Fall unteres NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	$\overline{P_t}$	1	0
1	0	0	1	0	1
1	1	P_t	$\overline{P_t}$	P_t	$\overline{P_t}$

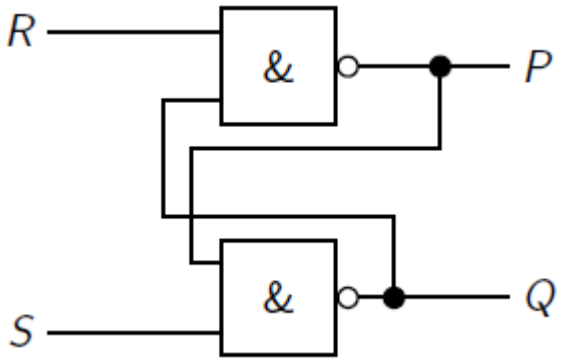


8.2 Bistabile Kippstufe

Analyse der bi-stabilen NAND-Kippstufe

2. Fall unteres NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	$\overline{P_t}$	1	0
1	0	0	1	0	1
1	1	P_t	$\overline{P_t}$	P_t	$\overline{P_t}$

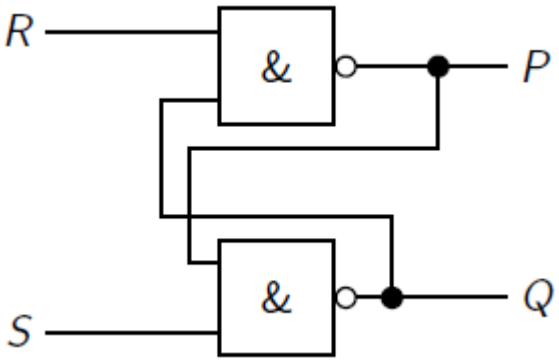


$$P_{t+2\Delta} = \overline{RQ_{t+2\Delta}}$$

Analyse der bi-stabilen NAND-Kippstufe

2. Fall unteres NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	$\bar{P}_t$	1	0
1	0	0	1	0	1
1	1	P_t	$\bar{P}_t$	P_t	$\bar{P}_t$



$$P_{t+2\Delta} = \overline{RQ_{t+2\Delta}} = \bar{R} \vee \overline{Q_{t+2\Delta}} = \bar{R} \vee \overline{\overline{SP_{t+\Delta}}} = \bar{R} \vee SP_{t+\Delta}$$

$$= \bar{R} \vee S\overline{RQ_{t+\Delta}} = \bar{R} \vee S(\bar{R} \vee \overline{Q_{t+\Delta}}) = \bar{R} \vee S\bar{R} \vee S\overline{Q_{t+\Delta}}$$

$$= \bar{R} \vee S\overline{Q_{t+\Delta}} = \bar{R} \vee S\overline{\overline{SP_t}} = \bar{R} \vee SSP_t = \bar{R} \vee SP_t$$

Fazit zum Ausgang P der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller

$$P_{t+2\Delta} = \overline{R} \vee S\overline{Q_t}$$

2. Fall unteres NAND-Gatter schneller

$$P_{t+2\Delta} = \overline{R} \vee SP_t$$

Beobachtung Wenn $P_t = \overline{Q_t}$, ist das Verhalten an P_t

stabil, also von den Schaltzeiten der Gatter unabhängig.

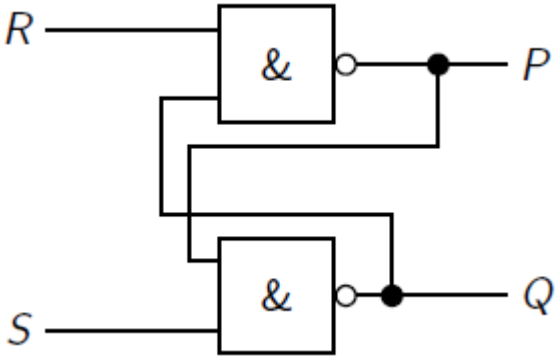
Was ist mit dem anderen Ausgang?

8.2 Bistabile Kippstufe

Analyse der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	$\overline{Q_t}$	1	0	1
1	1	$\overline{Q_t}$	Q_t	$\overline{Q_t}$	Q_t



$$Q_{t+2\Delta} = \overline{SP_{t+2\Delta}} = \bar{S} \vee \overline{P_{t+2\Delta}} = \bar{S} \vee \overline{\overline{RQ_{t+\Delta}}} = \bar{S} \vee RQ_{t+\Delta}$$

$$= \bar{S} \vee R\overline{SP_{t+\Delta}} = \bar{S} \vee R(\bar{S} \vee \overline{P_{t+\Delta}}) = \bar{S} \vee R\bar{S} \vee R\overline{P_{t+\Delta}}$$

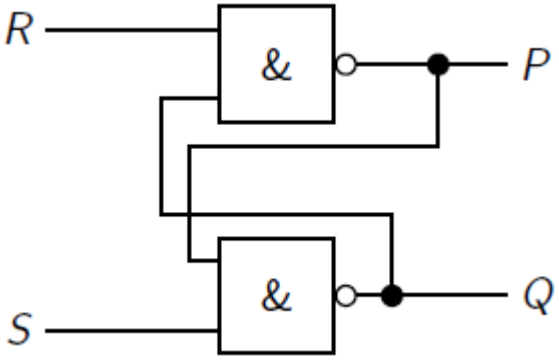
$$= \bar{S} \vee R\overline{P_{t+\Delta}} = \bar{S} \vee R\overline{\overline{RQ_t}} = \bar{S} \vee RRQ_t = \bar{S} \vee RQ_t$$

8.2 Bistabile Kippstufe

Analyse der bi-stabilen NAND-Kippstufe

2. Fall unteres NAND-Gatter schneller

R_t	S_t	$P_{t+\Delta}$	$Q_{t+\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	$\overline{P_t}$	1	0
1	0	0	1	0	1
1	1	P_t	$\overline{P_t}$	P_t	$\overline{P_t}$



$$Q_{t+2\Delta} = \overline{SP_{t+\Delta}} = \bar{S} \vee \overline{P_{t+\Delta}} = \bar{S} \vee \overline{\overline{RQ_{t+\Delta}}} = \bar{S} \vee RQ_{t+\Delta}$$

$$= \bar{S} \vee R\overline{SP_t} = \bar{S} \vee R(\bar{S} \vee \overline{P_t}) = \bar{S} \vee R\bar{S} \vee R\overline{P_t}$$

$$= \bar{S} \vee R\overline{P_t}$$

8.2 Bistabile Kippstufe

Fazit der Analyse der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller

$$P_{t+2\Delta} = \bar{R} \vee S\overline{Q_t}$$

$$Q_{t+2\Delta} = S \vee RQ_t$$

2. Fall unteres NAND-Gatter schneller

$$P_{t+2\Delta} = \bar{R} \vee SP_t$$

$$Q_{t+2\Delta} = S \vee R\overline{P_t}$$

also Wenn $Q_t = \overline{P_t}$, so ist das Verhalten

stabil, von den Schaltzeiten der Gatter unabhängig.

also Forderung $P_t \neq Q_t$

8.2 Bistabile Kippstufe

Wertetabelle bi-stabile NAND-Kippstufe

		oberes Gatter schneller		unteres Gatter schneller	
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	1	$\overline{Q_t}$	Q_t	P_t	$\overline{P_t}$

8.2 Bistabile Kippstufe

Wertetabelle bi-stabile NAND-Kippstufe

		oberes Gatter schneller		unteres Gatter schneller	
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	1	$\overline{Q_t}$	Q_t	P_t	$\overline{P_t}$

Beobachtung

Wir müssen nur R = S = 0 ausschließen, damit die Forderung $P_t \neq Q_t$ gilt.

8.2 Bistabile Kippstufe

Wertetabelle bi-stabile NAND-Kippstufe

		oberes Gatter schneller		unteres Gatter schneller	
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	1	$\overline{Q_t}$	Q_t	P_t	$\overline{P_t}$

Beobachtung Wir müssen nur R = S = 0 ausschließen.

R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	1	1	0
1	0	0	1
1	1	P_t	$\overline{P_t}$

- (R, S) = (0, 1) setzt P = 1
- (R, S) = (1, 0) setzt P = 0
- (R, S) = (1, 1) lässt P unverändert

Fazit Bi-stabile NAND-Kippstufe realisiert 1-Bit-Speicher!

8.2 Bistabile Kippstufe

Erstes Fazit zu sequenziellen Schaltungen

Bi-stabile NAND-Kippstufe realisiert 1-Bit-Speicher.

Beobachtung

- Kreise in „Schaltnetzen“ manchmal sinnvoll
- neue Funktionalität
- Analyse schwierig

Wunsch strukturierter Entwurf

8. Synchrone Schaltwerke

1. Einleitung ✓
2. Bistabile Kippstufe ✓
3. **Automaten**
4. Synchrone Schaltwerke
5. Serienaddierwerke
6. Speicher & Schieberegister
7. Takt

8.3 Automaten

Automaten

Wunsch formales Modell eines Automaten

Was ist überhaupt ein Automat?

Beispiele

- Getränke-Automat
- einfache Ampelsteuerung
- Steuerung einer Waschmaschine

Gegenbeispiele

- Geldspielautomat (wegen der Zufalls-Komponente)
- Computer (zu komplex)
- Mensch (für uns nicht formal beschreibbar)

8.3 Automaten

Automatenmodell

Grobbeschreibung

- verarbeitet eine Eingabe
- erzeugt eine Ausgabe
- ist in einem Zustand
- arbeitet in Takten
- arbeitet deterministisch (exakt vorhersagbar)

jetzt exakte, formale Beschreibung

Definition Mealy-Automat

Definition

Ein Mealy-Automat $M = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$ ist definiert durch:

- endliche **Zustandsmenge** Q
- **Startzustand** $q_0 \in Q$
- endliches **Eingabealphabet** Σ
- endliches **Ausgabealphabet** Δ
- **Zustandsüberführungsfunktion** $\delta : Q \times \Sigma \rightarrow Q$
- **Ausgabefunktion** $\lambda : Q \times \Sigma \rightarrow \Delta \cup \{\varepsilon\}$

8.3 Automaten

Definition Mealy-Automat

Definition

Ein Mealy-Automat $M = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$ ist definiert durch:

- endliche **Zustandsmenge** Q
- **Startzustand** $q_0 \in Q$
- endliches **Eingabealphabet** Σ
- endliches **Ausgabealphabet** Δ
- **Zustandsüberführungsfunktion** $\delta : Q \times \Sigma \rightarrow Q$
- **Ausgabefunktion** $\lambda : Q \times \Sigma \rightarrow \Delta \cup \{\varepsilon\}$
- In einem **Takt** mit aktuellem Zustand q und Eingabesymbol w
  - **schreibt** der Automat $\lambda(q, w)$,
  - **wechselt** der Automat in den Zustand $\delta(q, w)$.

Beispiel Mealy-Automat

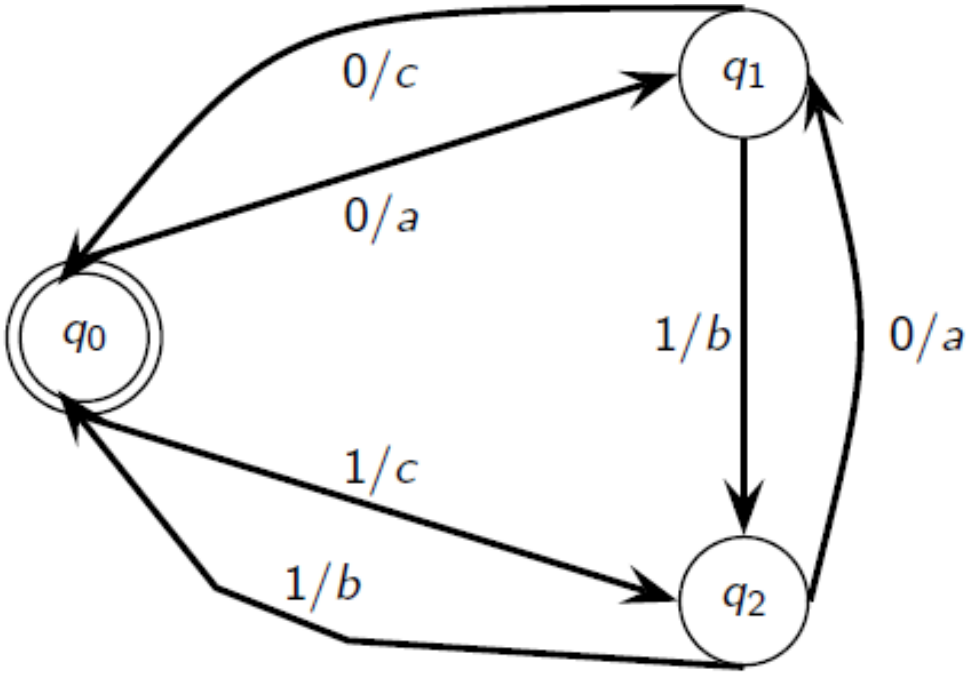
$Q = \{q_0, q_1, q_2\}, \quad \sum = \{0, 1\}, \quad \Delta = \{a, b, c\}$

aktueller Zustand	Eingabe	Folge-zustand	Ausgabe
$q \in Q$	$w \in \sum$	$\delta(q, w)$	$\lambda(q, w)$
q_0	0	q_1	a
q_0	1	q_2	c
q_1	0	q_0	c
q_1	1	q_2	b
q_2	0	q_1	a
q_2	1	q_0	b

Eingabe 0 1 0 0 Ausgabe

Beispiel Mealy-Automat

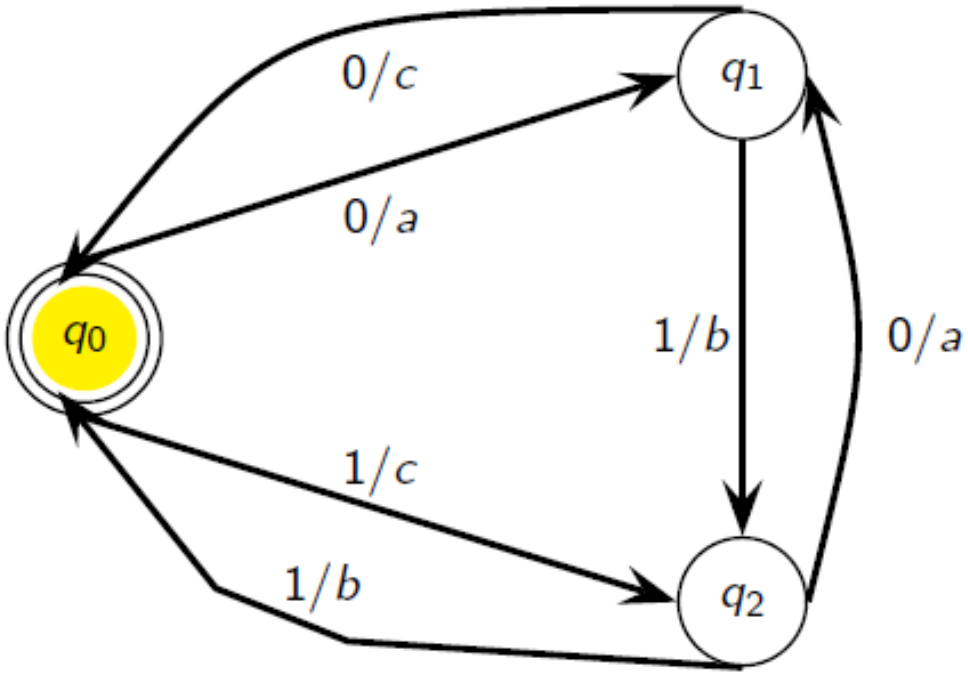
$Q = \{q_0, q_1, q_2\}, \quad \sum = \{0, 1\}, \quad \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe

Beispiel Mealy-Automat

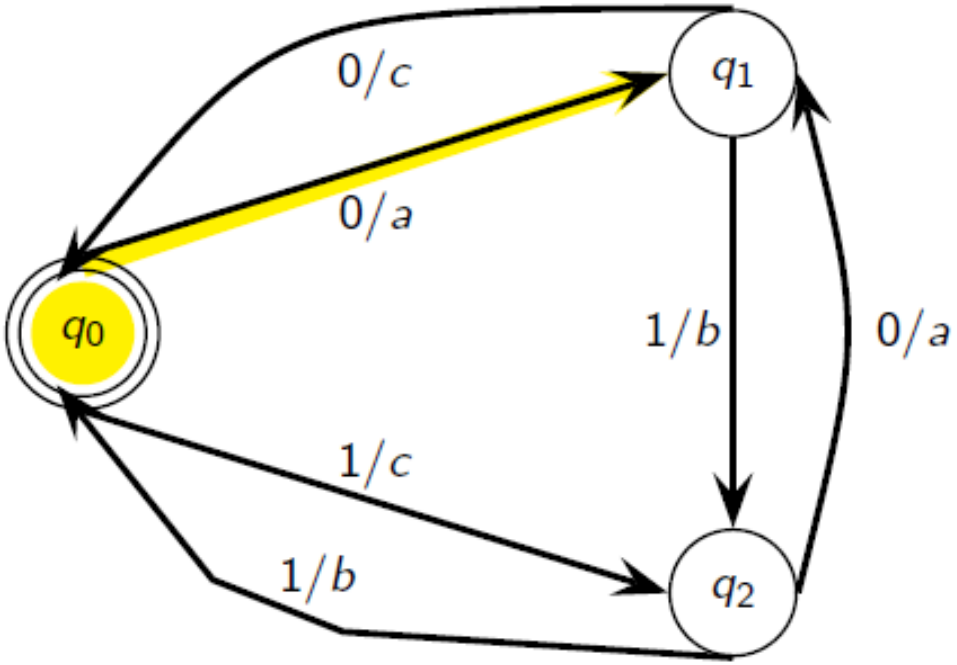
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe

Beispiel Mealy-Automat

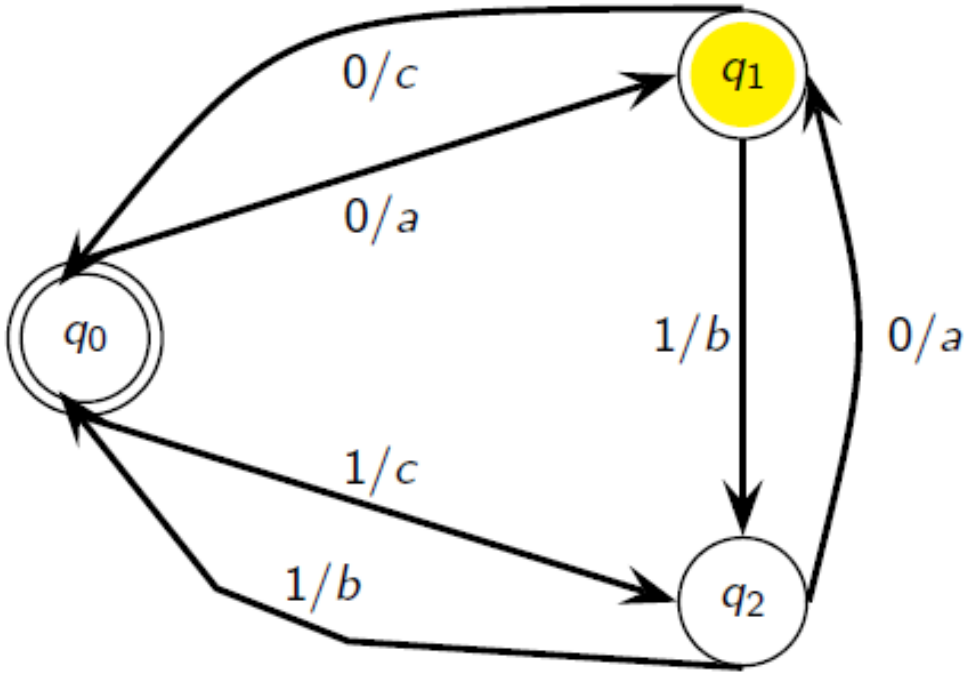
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe

Beispiel Mealy-Automat

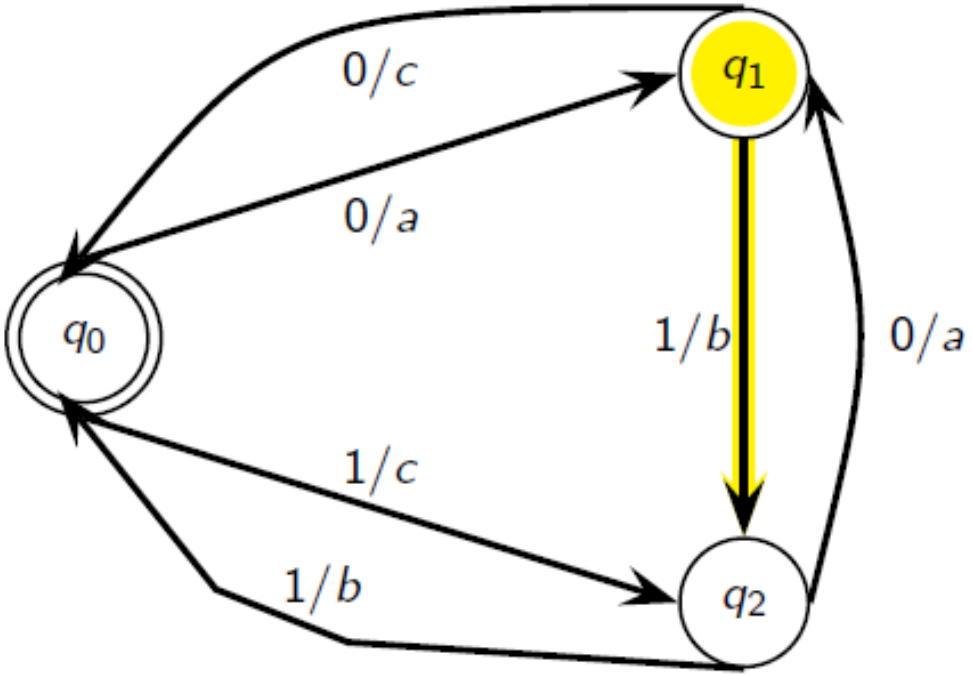
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a

Beispiel Mealy-Automat

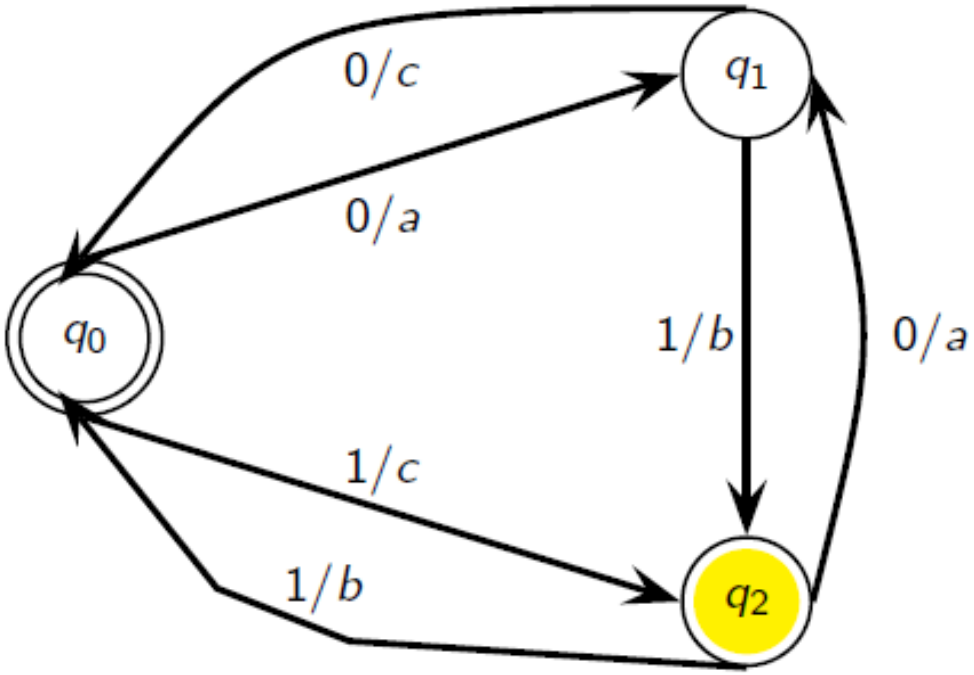
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a b

Beispiel Mealy-Automat

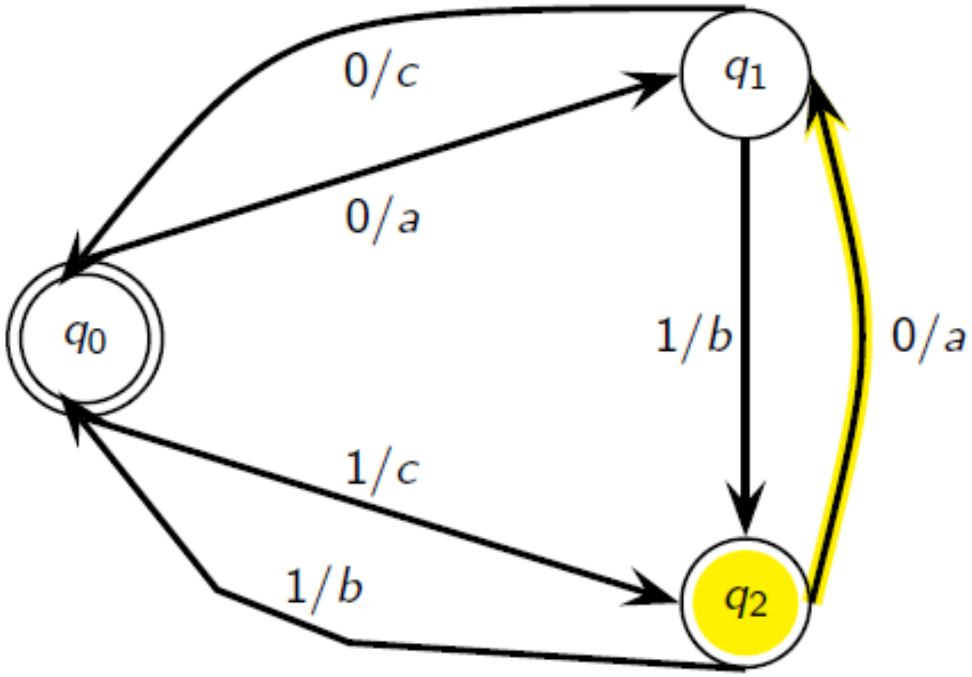
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a b

Beispiel Mealy-Automat

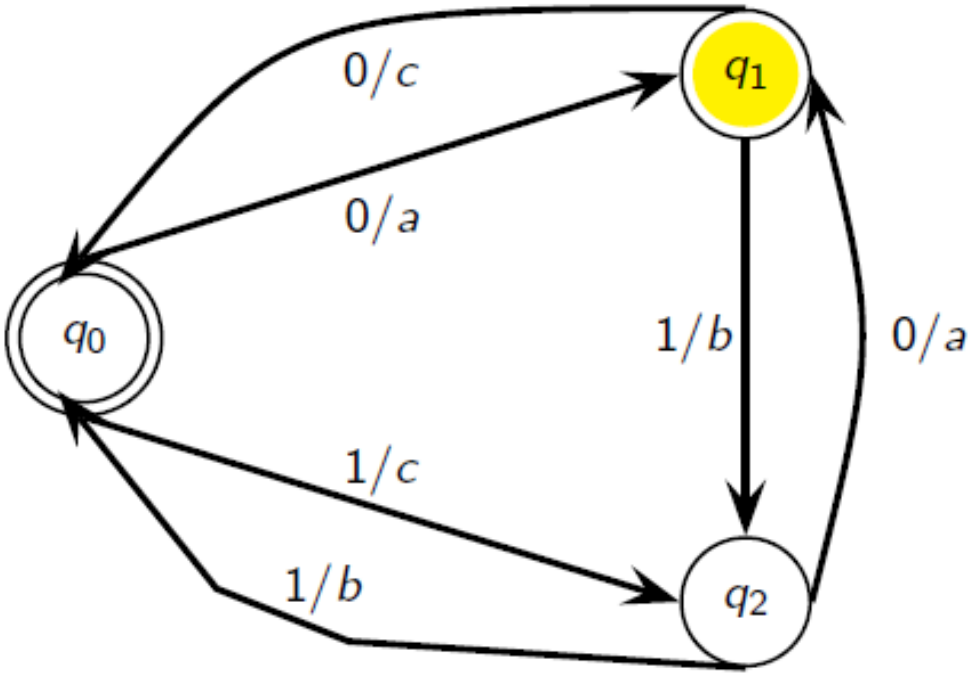
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a b a

Beispiel Mealy-Automat

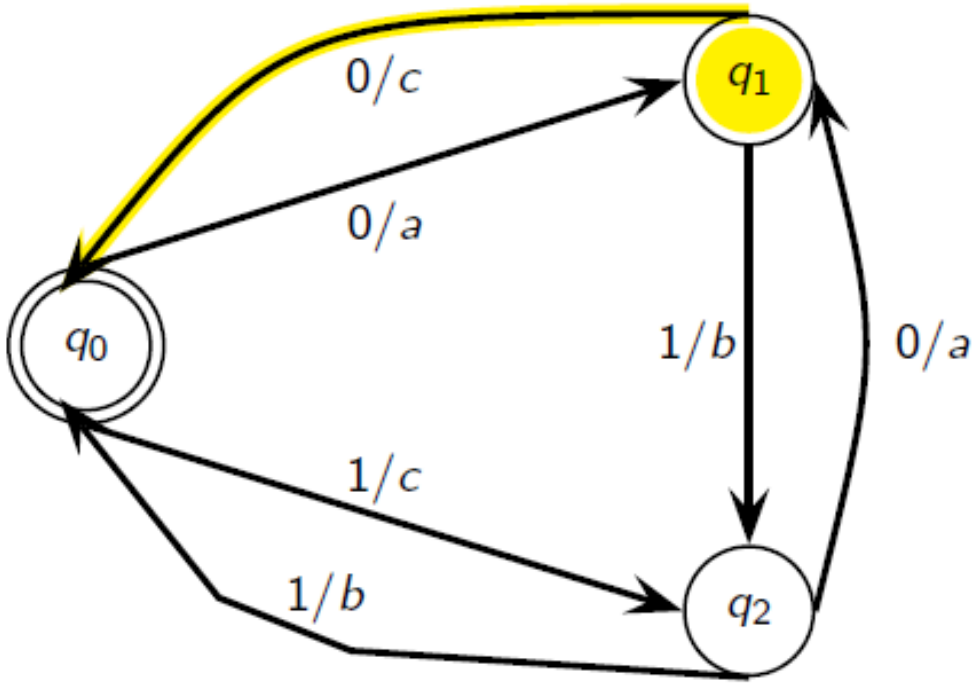
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a b a

Beispiel Mealy-Automat

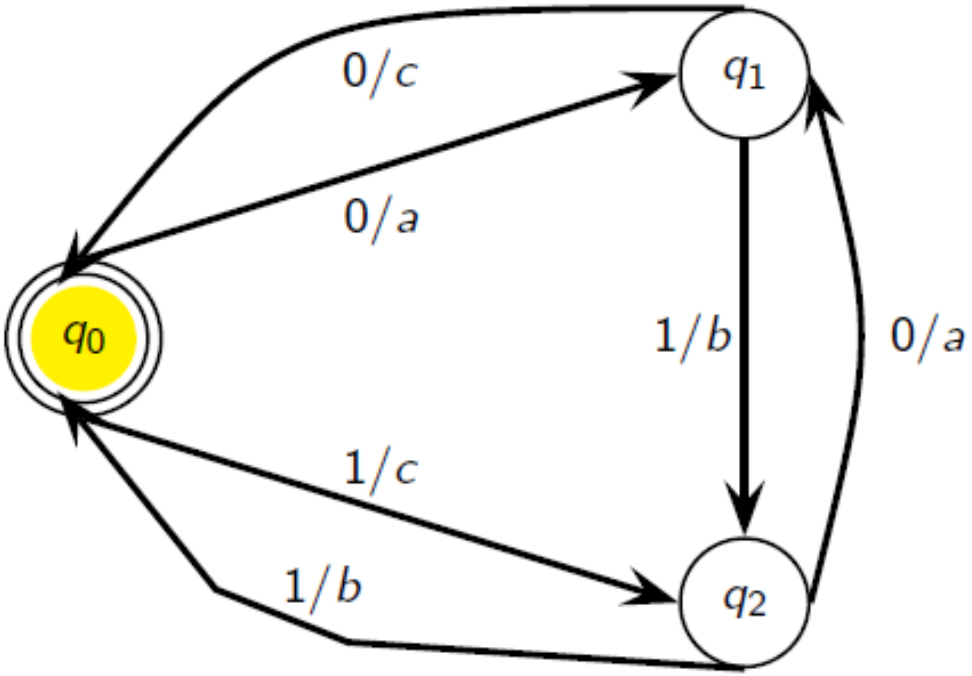
$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a b a

Beispiel Mealy-Automat

$Q = \{q_0, q_1, q_2\}, \sum = \{0, 1\}, \Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a b a c

Äquivalenz von Automaten

Definition

Zwei Mealy-Automaten heißen **äquivalent**, wenn sie für jede Eingabe $w \in \sum^*$ die gleiche Ausgabe $a \in \Delta^*$ erzeugen.

Beobachtung

- Äquivalente Automaten können unterschiedlich groß sein.
- Man wünscht sich möglichst kleine Automaten.
- Komplexer Problemkreis,
  - umfasst auch effiziente Minimierung von Automaten
  - Näher i. d. Theoretischen Informatik (GTI)

Hier: noch ein anderes (ähnliches!) Automaten-Modell

Definition Moore-Automat

Definition

Ein Moore-Automat $M = (Q, q_0, \sum, \Delta, \delta, \lambda)$ ist definiert durch:

- endliche **Zustandsmenge** Q
- **Startzustand** $q_0 \in Q$
- endliches **Eingabealphabet** $\sum$
- endliches **Ausgabealphabet** Δ
- **Zustandsüberführungsfunktion** $\delta : Q \times \sum \rightarrow Q$
- **Ausgabefunktion** $\lambda: Q \rightarrow \Delta \cup \{\varepsilon\}$

8.3 Automaten

Definition Moore-Automat

Definition

Ein Moore-Automat $M = (Q, q_0, \sum, \Delta, \delta, \lambda)$ ist definiert durch:

- endliche **Zustandsmenge** Q
- **Startzustand** $q_0 \in Q$
- endliches **Eingabealphabet** $\sum$
- endliches **Ausgabealphabet** Δ
- **Zustandsüberführungsfunktion** $\delta : Q \times \sum \rightarrow Q$
- **Ausgabefunktion** $\lambda: Q \rightarrow \Delta \cup \{\varepsilon\}$

In einem **Takt** mit aktuellem Zustand q und Eingabesymbol w

- schreibt der Automat $\lambda(\delta(q, w))$,
- wechselt der Automat in den Zustand $\delta(q, w)$.

8.3 Automaten

Definition Moore-Automat (Unterschied zum Mealy-Automat)

Definition

Ein Moore-Automat $M = (Q, q_0, \sum, \Delta, \delta, \lambda)$ ist definiert durch:

- endliche **Zustandsmenge** Q
- **Startzustand** $q_0 \in Q$
- endliches **Eingabealphabet** $\sum$
- endliches **Ausgabealphabet** Δ
- **Zustandsüberführungsfunktion** $\delta : Q \times \sum \rightarrow Q$
- **Ausgabefunktion** $\lambda: Q \rightarrow \Delta \cup \{\varepsilon\}$ $\quad\quad \lambda : Q \times \Sigma \rightarrow \Delta \cup \{\varepsilon\}$

In einem **Takt** mit aktuellem Zustand q und Eingabesymbol w

- schreibt der Automat $\lambda(\delta(q, w))$,
- wechselt der Automat in den Zustand $\delta(q, w)$.

Mealy- und Moore-Automaten

Beobachtung Zu jedem Moore-Automaten gibt es einen äquivalenten Mealy-Automaten.

denn zu Moore-Automat $A = (Q, q_0, \sum, \Delta, \delta, \lambda)$

ist Mealy-Automat $A' = (Q, q_0, \sum, \Delta, \delta, \lambda')$

mit $\lambda'(q, w) := \lambda(\delta(q, w))$

offensichtlich **äquivalent**

Mealy- und Moore-Automaten

Beobachtung Zu jedem Mealy-Automaten gibt es eine äquivalenten Moore-Automaten.

denn zu Mealy-Automaten $A = (Q, q_0, \sum, \Delta, \delta, \lambda)$

ist Moore-Automat $A' = (Q', q_0', \sum, \Delta, \delta', \lambda')$

mit $Q' := Q \times (\Delta \cup \{\varepsilon\})$, $q_0' := (q_0, \varepsilon)$,

$$\delta'(q', w) = \delta'((q, v), w) := (\delta(q, w), \lambda(q, w)),$$

$$\lambda(\delta(q', w)) = \lambda'((q, v)) := v$$

äquivalent

8.3 Automaten

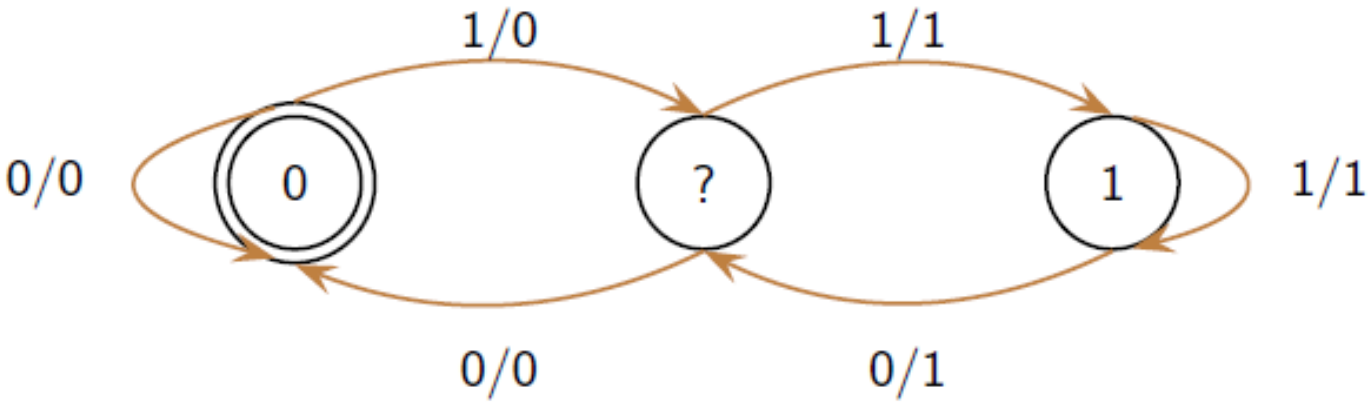
Einfacher Beispiel-Automat

Aufgabe: einfache „Datenglättung“

Filtere isolierte Bits aus Datenstrom aus.

Es gilt: $\sum := \{0, 1\}, \Delta := \{0, 1\}$

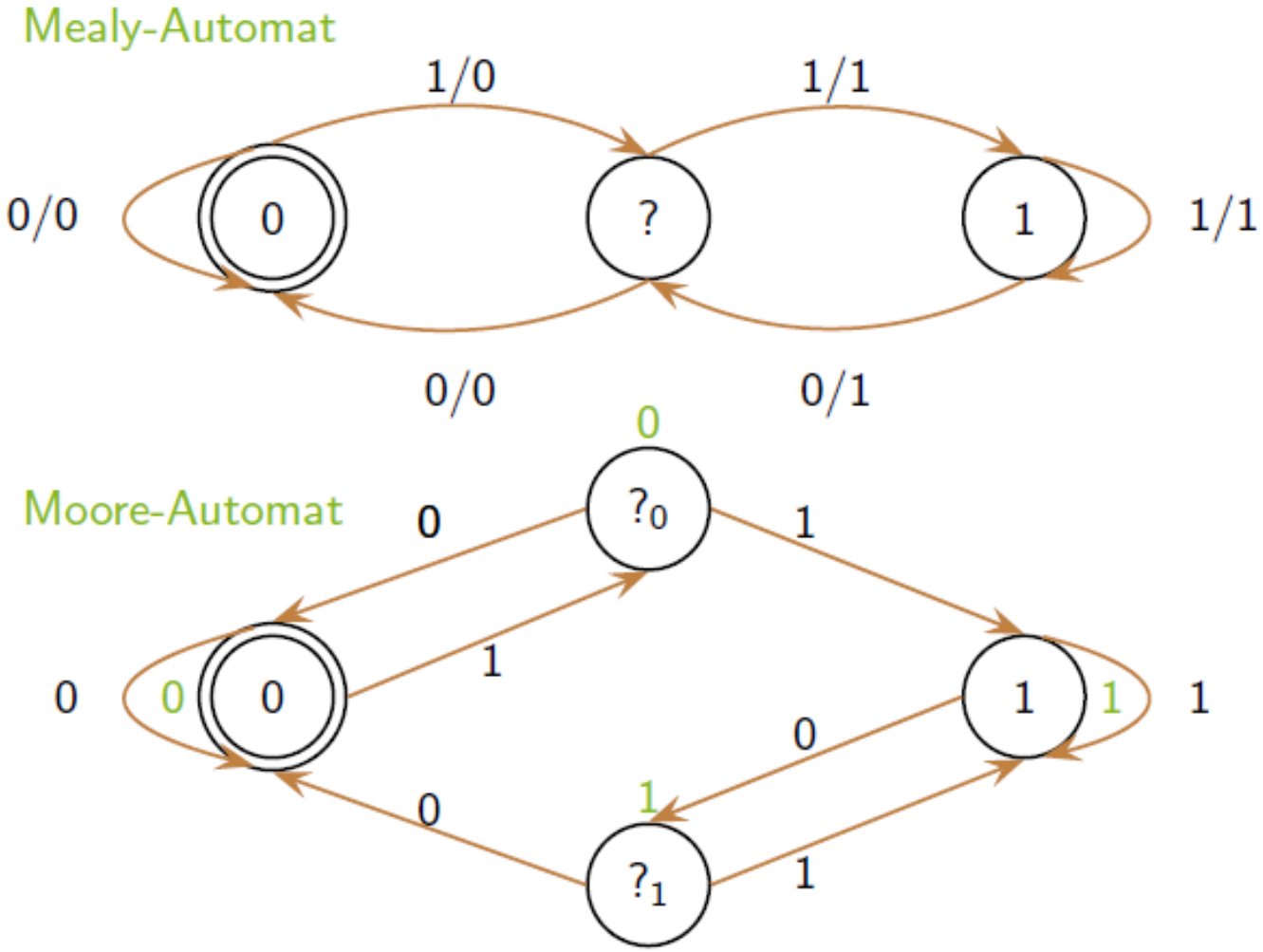
Mealy-Automat



mit $Q := \{0, 1, ?\}$, $q_0 := 0$

8.3 Automaten

Äquivalente Automaten „Bit-Filter“

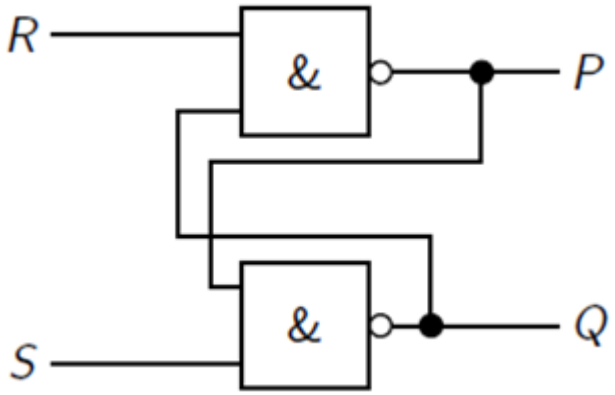


8. Synchrone Schaltwerke

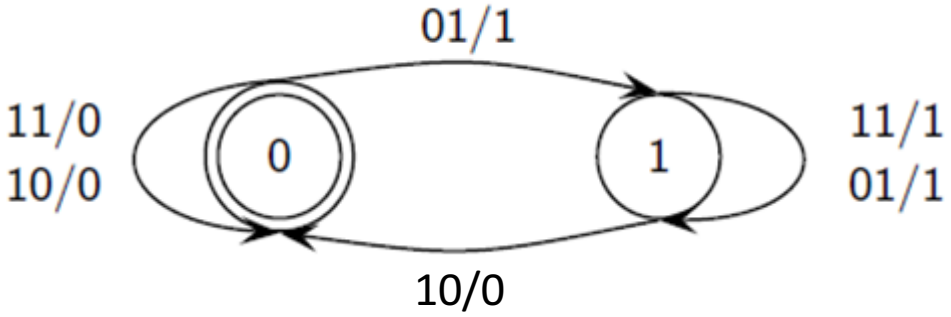
1. Einleitung ✓
2. Bistabile Kippstufe ✓
3. Automaten ✓
4. **Synchrone Schaltwerke**
5. Serienaddierwerke
6. Speicher & Schieberegister
7. Takt

Automaten & Schaltungen: Synchr. Schaltwerke

Erinnerung bi-stabile NAND-Kippstufe



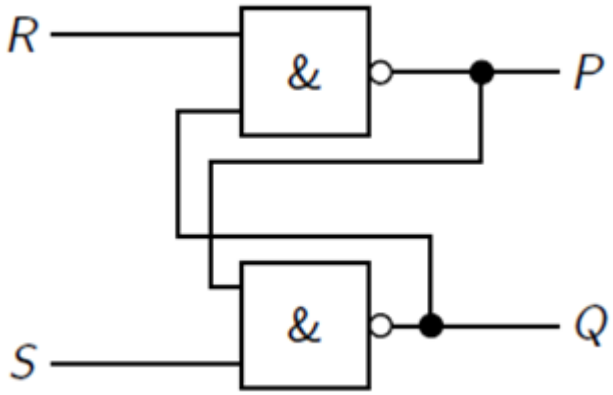
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	1	1	0
1	0	0	1
1	1	P_t	$\overline{P_t}$



$$Q = \{0, 1\},\ q_0 = 0$$
$$\sum = \{01, 10, 11\}$$
$$\Delta = \{0, 1\}$$

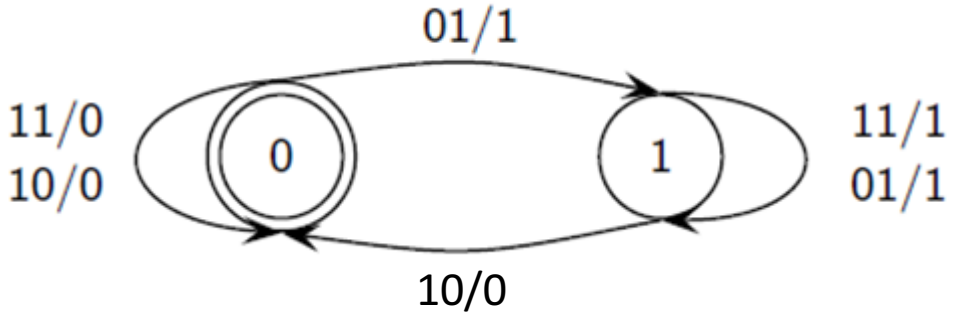
Vergleich Automat und NAND-Kippstufe

nicht getaktet



R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	1	1	0
1	0	0	1
1	1	P_t	$\overline{P_t}$

getaktet



$$Q = \{0, 1\},\ q_0 = 0$$
$$\sum = \{01, 10, 11\}$$
$$\Delta = \{0, 1\}$$

Synchrone Schaltwerke

ab jetzt getaktete Schaltwerke

also Führe Taktsignal T ein

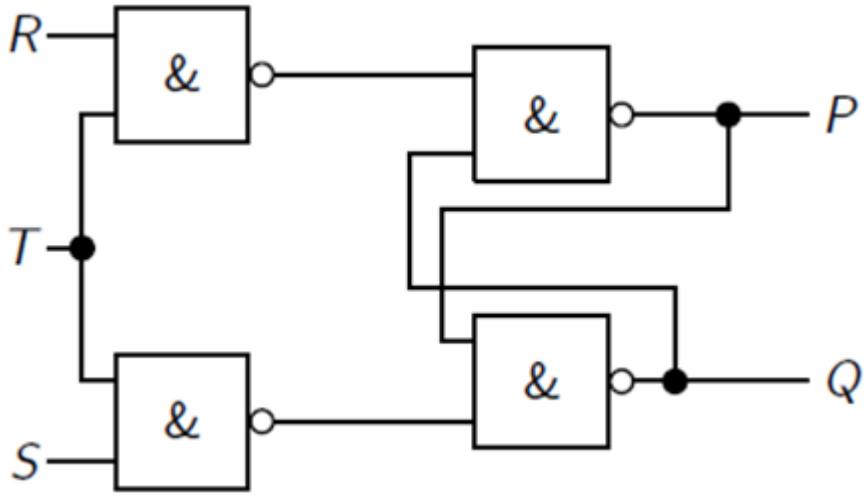
verschiedene technische Möglichkeiten

- Pegelsteuerung: aktiv, wenn 1 anliegt
- positive Flankensteuerung: aktiv, wenn Wechsel von 0 nach 1
- negative Flankensteuerung: aktiv, wenn Wechsel von 1 nach 0

digital-logische Ebene technisches Detail ignorieren

8.4 Synchrone Schaltwerke

RS-Flip-Flop



R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

Zustandstabelle NAND-Kippstufe

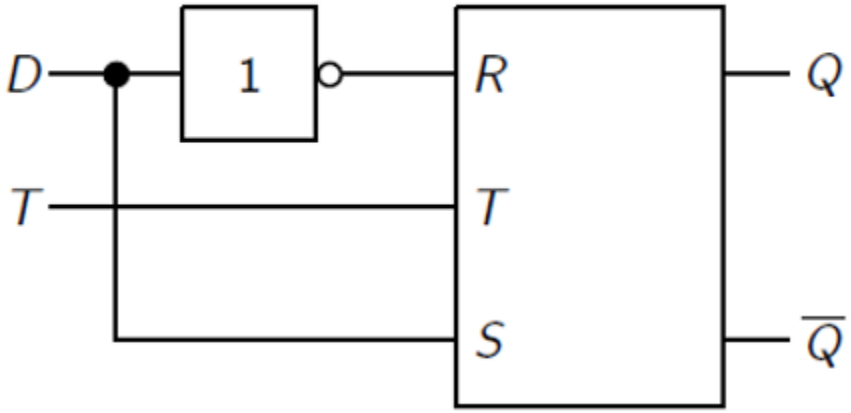
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	nicht erlaubt	
0	1	1	0
1	0	0	1
1	1	$\overline{Q_t}$	Q_t

Wertetabelle NAND

x	y	$\overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0

8.4 Synchrone Schaltwerke

D-Flip-Flop

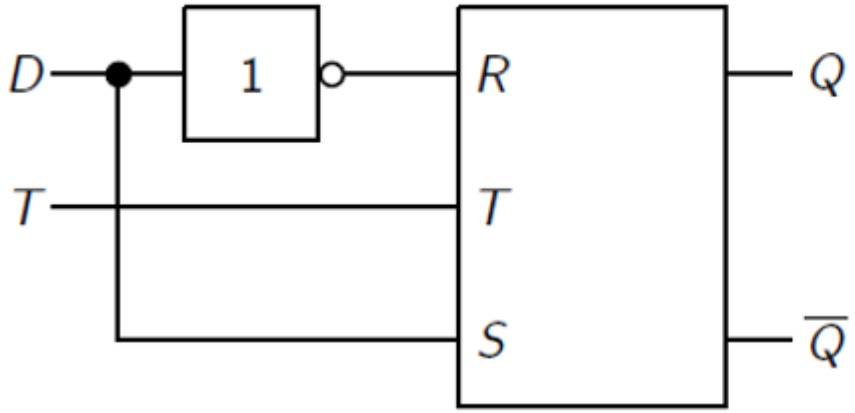


R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

D	Q
0	0
1	1

Sinnlos?

D-Flip-Flop



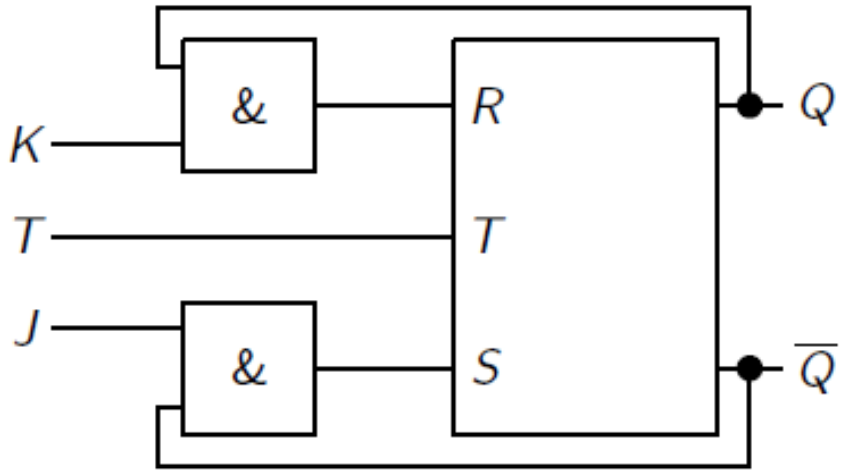
R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

D	Q
0	0
1	1

Sinnlos? Verzögerung um 1 Takt

Name Delay

JK-Flip-Flop

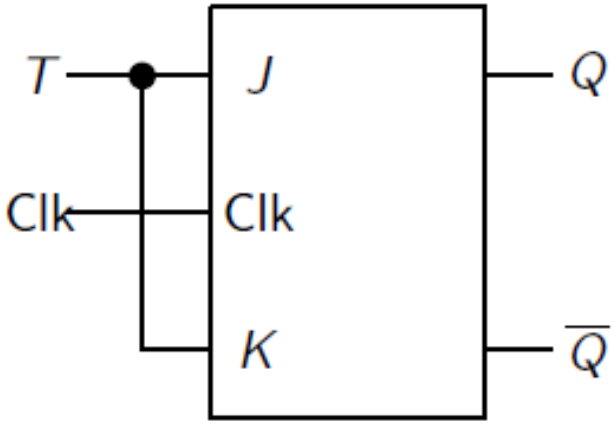


R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

J	K	R	S	Q
0	0	0	0	Q
0	1	Q	0	0
1	0	0	$\overline{Q}$	1
1	1	Q	$\overline{Q}$	$\overline{Q}$

positiv alle Eingaben erlaubt, sinnvolle Funktionalität, vielseitig

T-Flip-Flop



J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

T	Q
0	Q
1	$\overline{Q}$

Name Toggle

Flip-Flops

Wir haben also hier 4 verschiedene Flip-Flop-Typen

Wozu brauchen wir eigentlich Flip-Flops?

klar Realisierung von Speicher

Was müssen wir für den Einsatz als Speicher wissen?

klar gezielte Änderung von Speicherinhalten

also Zustandstabellen eigentlich nicht interessant

besser Ansteuertabellen

Flip-Flops

etwas präziser

Zustandstabelle	Ansteuertabelle
Eingabe ⇒ Zustand	Ist-Zustand, Soll-Zustand ⇒ Eingabe

Anmerkung Ansteuertabellen können „don’t care“ enthalten

Ansteuertabelle D-Flip-Flop

Zustandstabelle

D	Q
0	0
1	1

Ansteuertabelle

Q_{alt}	Q_{neu}	D
0	0	0
0	1	1
1	0	0
1	1	1

Beobachtung Ansteuerung D vom alten Zustand unabhängig

Ansteuertabelle T-Flip-Flop

Zustandstabelle

T	Q
0	Q
1	$\overline{Q}$

Ansteuertabelle

Q_{alt}	Q_{neu}	T
0	0	0
0	1	1
1	0	1
1	1	0

Beobachtung Kenntnis des „alten“ Zustands erforderlich

Ansteuertabelle RS-Flip-Flop

Zustandstabelle

R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

Ansteuertabelle

Q_{alt}	Q_{neu}	R	S
0	0	$*$	0
0	1	0	1
1	0	1	0
1	1	0	$*$

Beobachtung wenn Zustand nicht wechselt,

gibt es Freiheit in der Ansteuerung

Ansteuertabelle JK-Flip-Flop

Zustandstabelle

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Ansteuertabelle

Q_{alt}	Q_{neu}	J	K
0	0	0	$*$
0	1	1	$*$
1	0	$*$	1
1	1	$*$	0

Beobachtung immer Freiheit in der Ansteuerung

Unvollständig definierte Ansteuerfunktionen

Wir haben für RS-Flip-Flops und JK-Flip-Flops

nur partiell definierte Ansteuerfunktionen

Ist das ungünstig?

Nein!

Erinnerung Minimalpolynome für partiell definierte Funktionen

Erinnerung Realisierungen können wesentlich einfacher sein

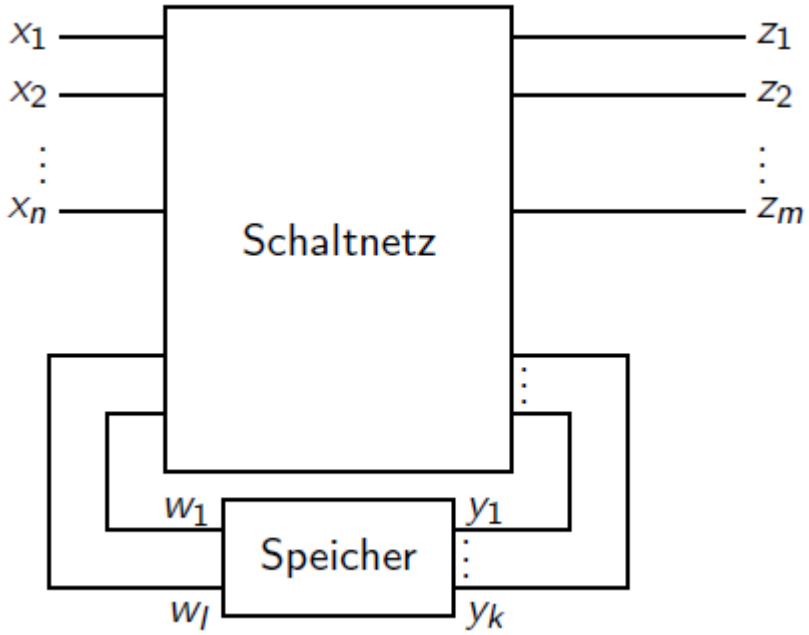
Erinnerung Minimalpolynom für partiell definiertes f

durch PI für f_1 und Überdeckung von f_0

8.4 Synchrone Schaltwerke

Huffmann Schaltwerk-Modell

ein allgemeines, formales Schaltwerkmodell

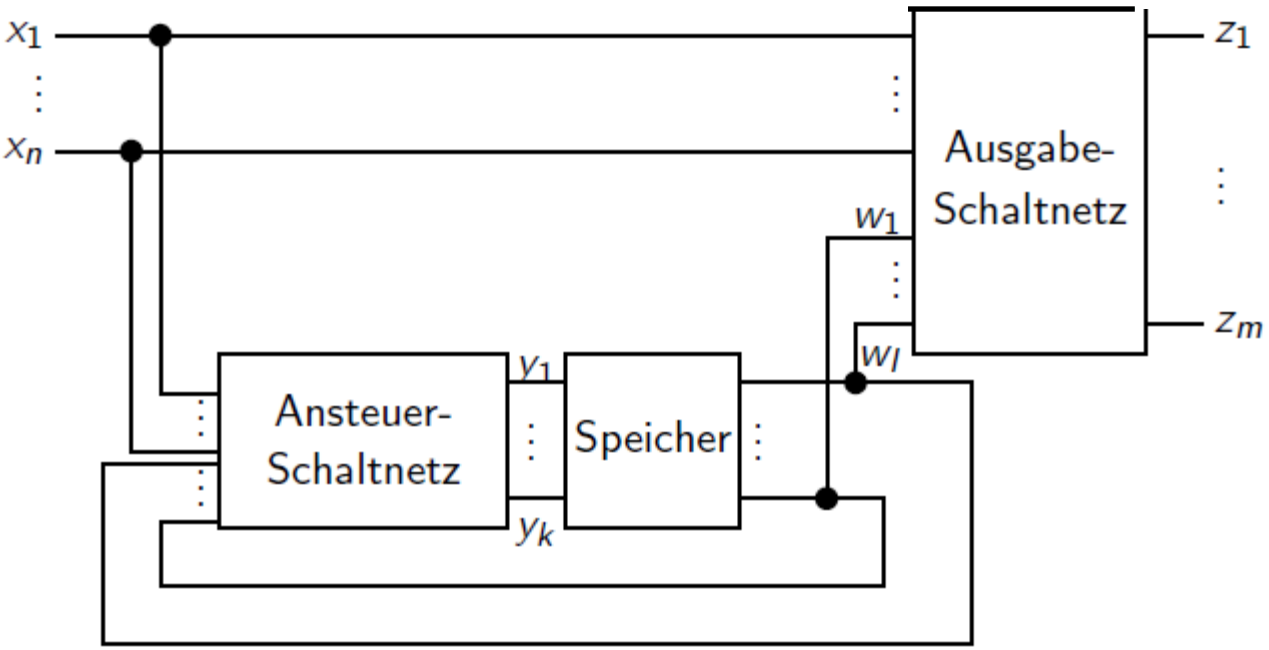


Beobachtung führt Gelerntes über Schaltnetze, Flip-Flops und Automaten sinnvoll zusammen

8.4 Synchrone Schaltwerke

Huffmann Schaltwerk-Modell

Etwas detaillierter



Schaltwerk-Entwurf

Wunsch strukturierter Schaltwerk-Entwurf

Was können wir überhaupt als Schaltwerk realisieren?

klar alles, was als Automat beschrieben werden kann

zum Beispiel

- Getränke-Automat
- Ampelsteuerung
- Waschmaschinen-Steuerung
- . . .

Anmerkung Heuristiken und Erfahrung sind wichtig

8. Synchrone Schaltwerke

1. Einleitung ✓
2. Bistabile Kippstufe ✓
3. Automaten ✓
4. Synchrone Schaltwerke ✓
5. **Serienaddierwerke**
6. Speicher & Schieberegister
7. Takt

8.5 Serienaddierwerke

Schritte beim Schaltwerk-Entwurf

0. **Verstehen** der Aufgabe
1. Spezifikation des **Verhaltens** (z.B. als Mealy-Automat)
2. Wahl der **Coderierung** von Eingaben, Zuständen, Ausgaben
3. **Wertetabelle** mit Eingaben, Zustand, Ausgaben, neuem Zustand
4. Wahl der **Flip-Flop-Typen**
5. Ergänzung der Wertetabelle um die **Flip-Flop-Ansteuerung**
6. Entwurf passender **boolescher Funktionen**
7. Entwurf passender **Schaltnetze**
8. Entwurf **vollständiges getaktetes Schaltwerk**

8.5 Serienaddierwerke

Beispiel zum Schaltwerk-Entwurf

zur Einführung ein „praktisches“ Beispiel

hilfreich besonders gut verstandenes Problem

Aufgabe Addition von zwei Betragszahlen

Erinnerung wir haben

Verfahren	Größe	Tiefe
• Schul-Methode	$\approx 5n$	$\approx 2n$
• Carry-Look-Ahead	$\approx n^2$	$\approx 2 \log_2 n$

jetzt klein und flach mit einem Schaltwerk

aber extrem langsam

Entwurf Addierwerk: Schritt 0

Schritt 0 **Verstehen** der Aufgabe

Wunsch Summanden bitweise eingeben

Summe bitweise erhalten

Offensichtlich geht nur von rechts nach links

Was muss man sich merken?

- nur den aktuellen Übertrag
- also 1 Bit

8.5 Serienaddierwerke

Entwurf Addierwerk: Schritt 1

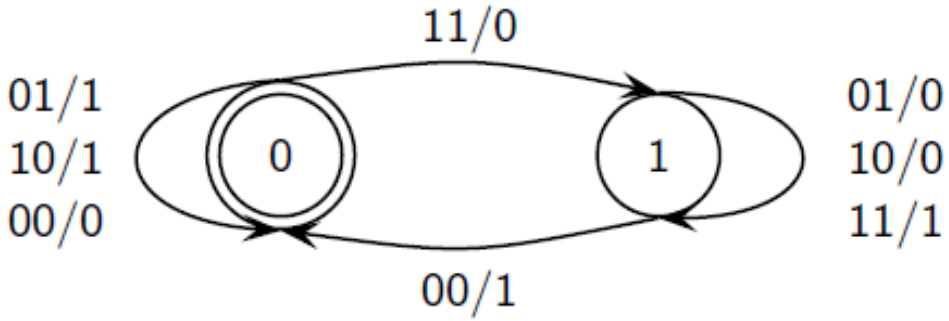
Schritt 1 Spezifikation des **Verhaltens**

Entscheidung Beschreibung durch Mealy-Automat

Eingabealphabet $\sum = \{00, 01, 10, 11\}$

Ausgabealphabet $\Delta = \{0, 1\}$ Summenbit

Zustandsmenge $Q = \{0, 1\}$ Übertragsbit



Entwurf Addierwerk: Schritt 2

Schritt 2 Wahl der Codierung: Eingaben, Zuständen, Ausgaben

Beobachtung im Allgemeinen (fast) jede Freiheit

hier kanonische Codierung naheliegend

Codierung von Q

$q \in Q$	Codierung
0	0
1	1

Codierung von Σ

$w \in \Sigma$	Codierung
00	0 0
01	0 1
10	1 0
11	1 1

Codierung von Δ

$w \in \Delta$	Codierung
0	0
1	1

Entwurf Addierwerk: Schritt 3

Schritt 3 **Wertetabelle** Eingaben, Zustand, neuer Zustand, Ausgaben

naheliegend Eingaben heißen *x, y;* alter Zustand heißt c_{alt}

neuer Zustand heißt c_{neu}; Ausgabe heißt *s*

c_{alt}	x	y	c_{neu}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

8.5 Serienaddierwerke

Entwurf Addierwerk: Schritt 4

Schritt 4 Wahl der **Flip-Flop-Typen**

Entscheidung JK-Flip-Flops

Anmerkung

- nicht viele überzeugende Gründe
- weil es besonders viele Freiheiten erlaubt
- weil der Zustandswechsel nichts nahelegt
- weil es oft benutzt wird

Entwurf Addierwerk: Schritt 5

Schritt 5 Ergänzung der Wertetabelle um **Flip-Flop-Ansteuerung**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

Ansteuertabelle JK-Flip-Flop

Q_{alt}	Q_{neu}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

Entwurf Addierwerk: Schritt 6

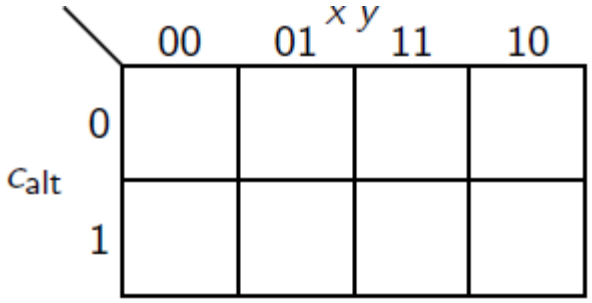
Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0



Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

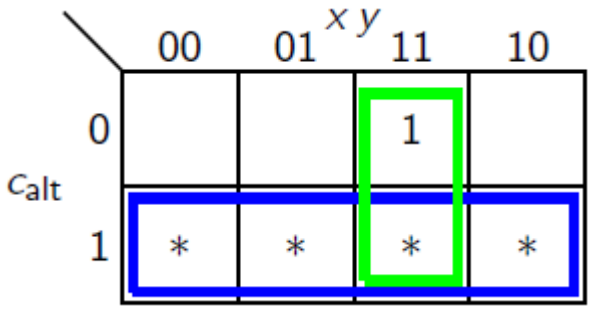
c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

c_{alt} \ $x\,y$	00	01	11	10
0			1	
1	*	*	*	*

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

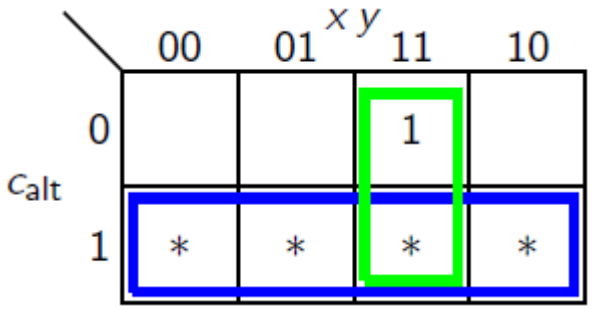
c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0



Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

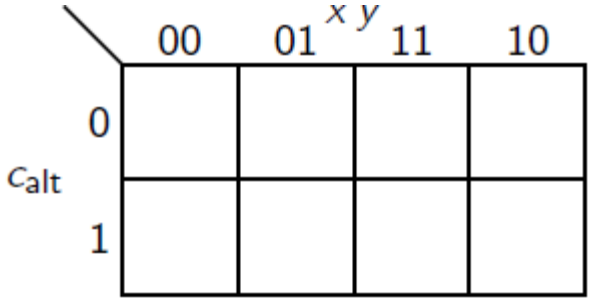


$$J(c_{alt}, x, y) = x\,y$$

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0



$$J(c_{alt}, x, y) = x\,y$$

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

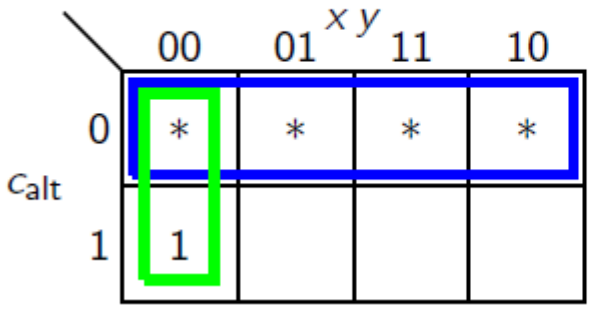
c_{alt} \ $x\,y$	00	01	11	10
0	*	*	*	*
1	1			

$$J(c_{alt}, x, y) = x\,y$$

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0



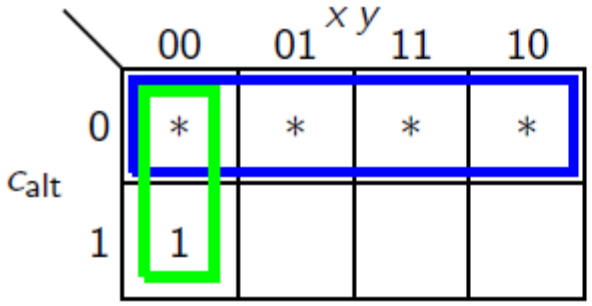
$$J(c_{alt}, x, y) = x\,y$$

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$

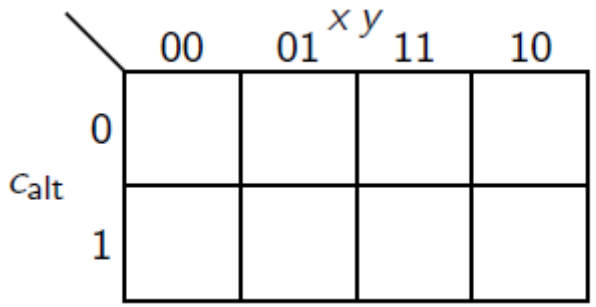


Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$



Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$

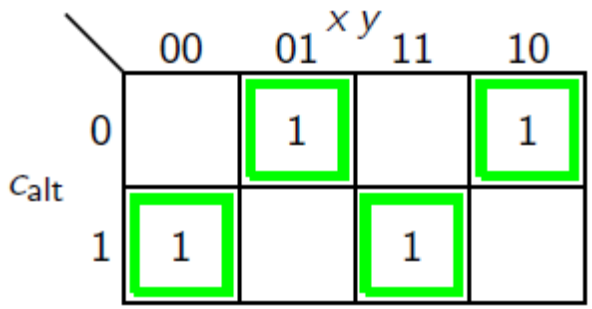
c_{alt} \ $x\,y$	00	01	11	10
0		1		1
1	1		1	

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

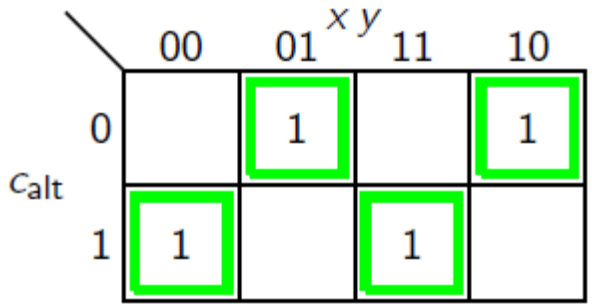
$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$



Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0



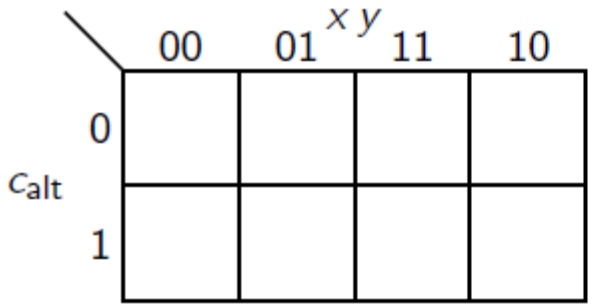
$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$
$$s(c_{alt}, x, y) = \overline{c_{alt}}\,\overline{x}\,y \vee \overline{c_{alt}}\,x\,\overline{y} \vee c_{alt}\,\overline{x}\,\overline{y} \vee c_{alt}\,x\,y$$

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender **boolescher Funktionen**

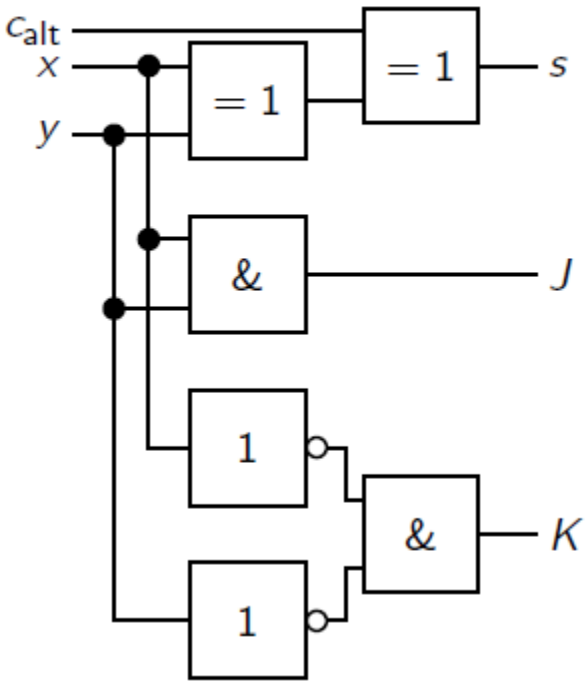
c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0

$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$
$$s(c_{alt}, x, y) = c_{alt} \oplus x \oplus y$$



Entwurf Addierwerk: Schritt 7

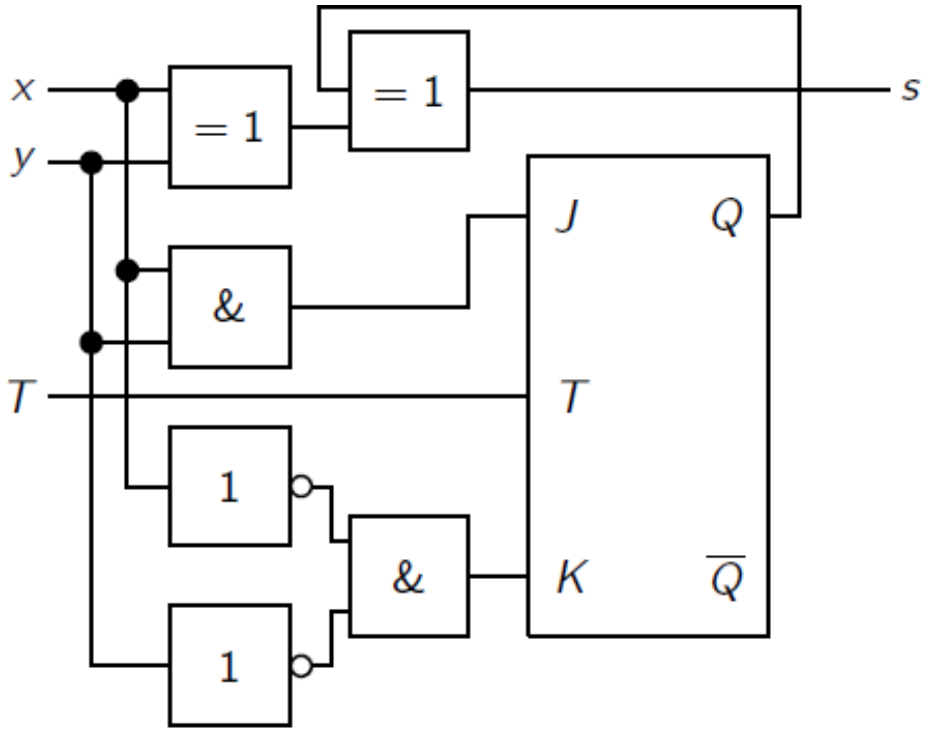
Schritt 7 Entwurf passender **Schaltnetze**



$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$
$$s(c_{alt}, x, y) = c_{alt} \oplus x \oplus y$$

Entwurf Addierwerk: Schritt 8

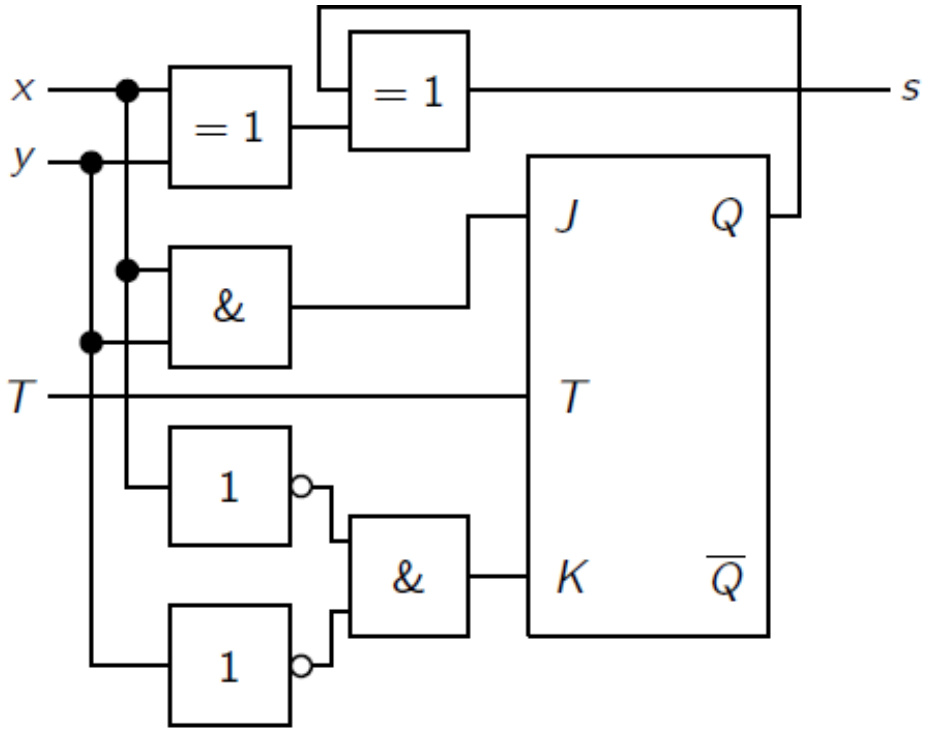
Schritt 8 Entwurf des **vollständigen Schaltwerks**



Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

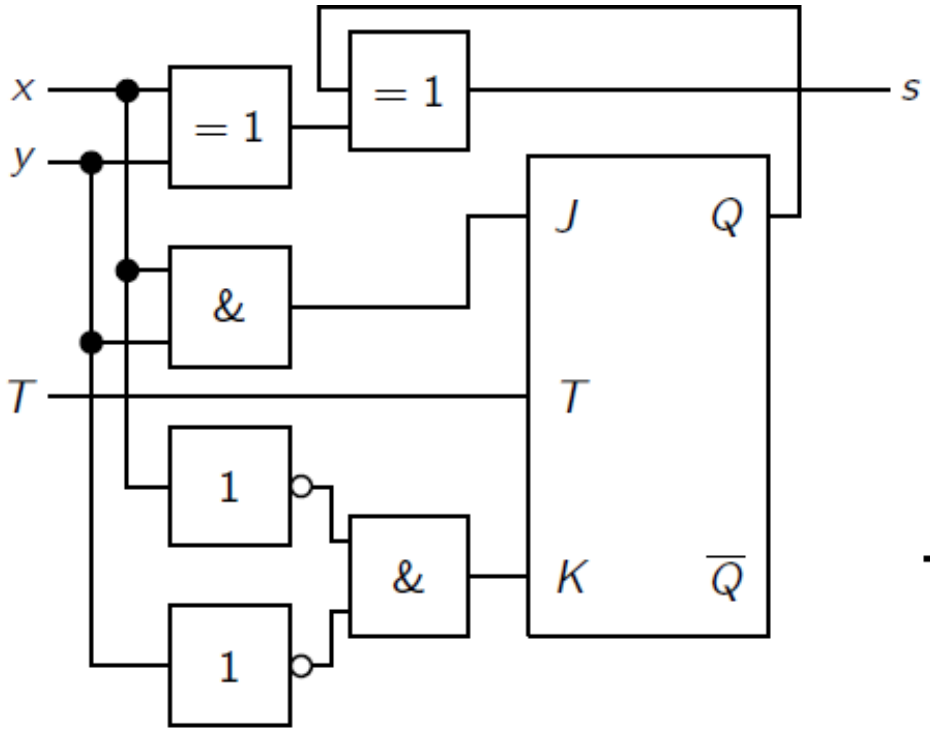


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

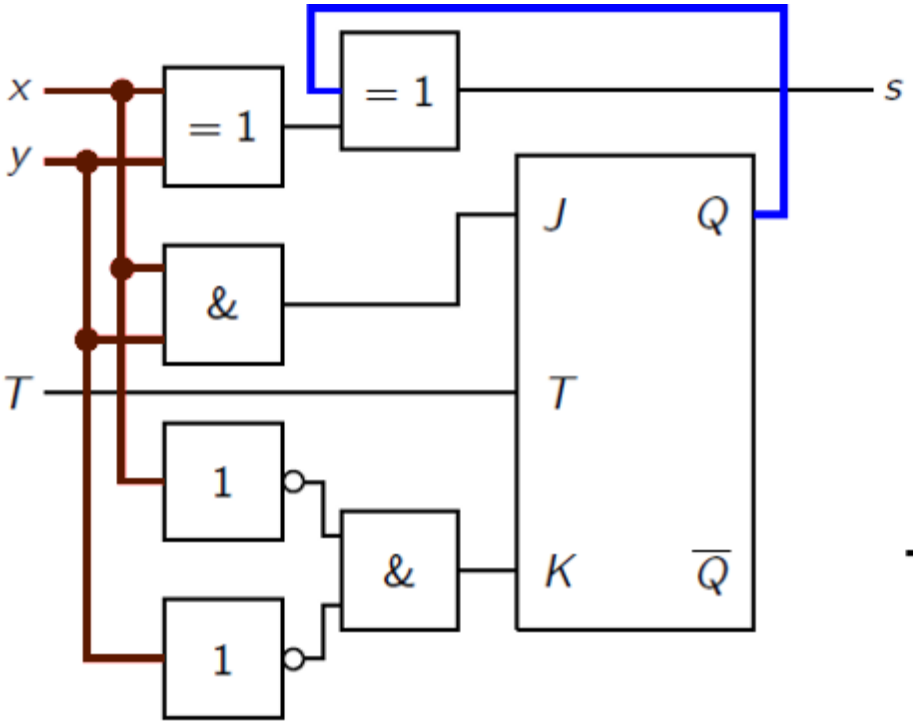


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

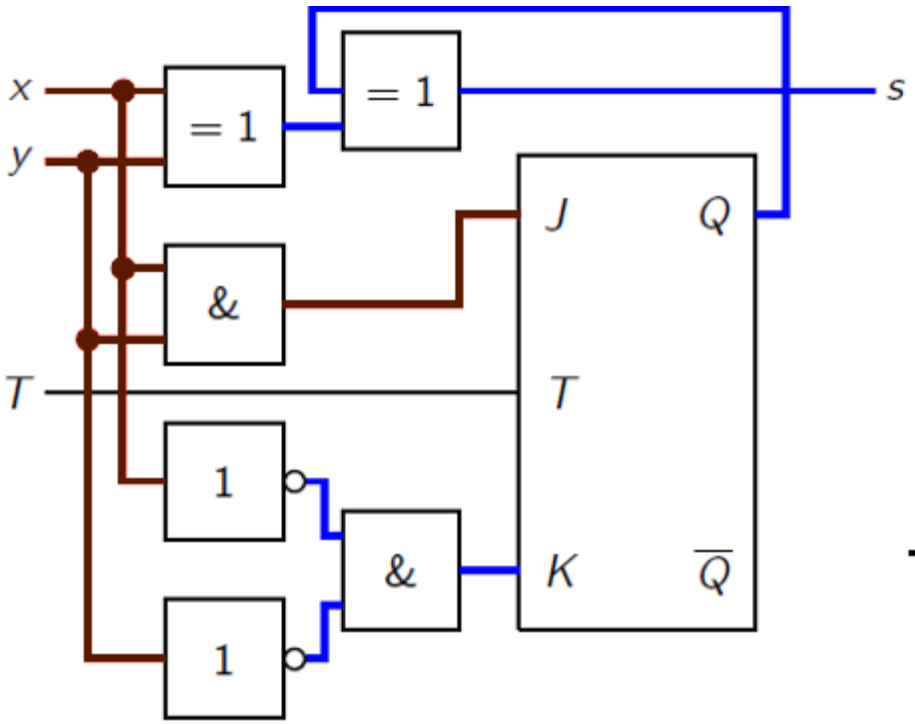


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

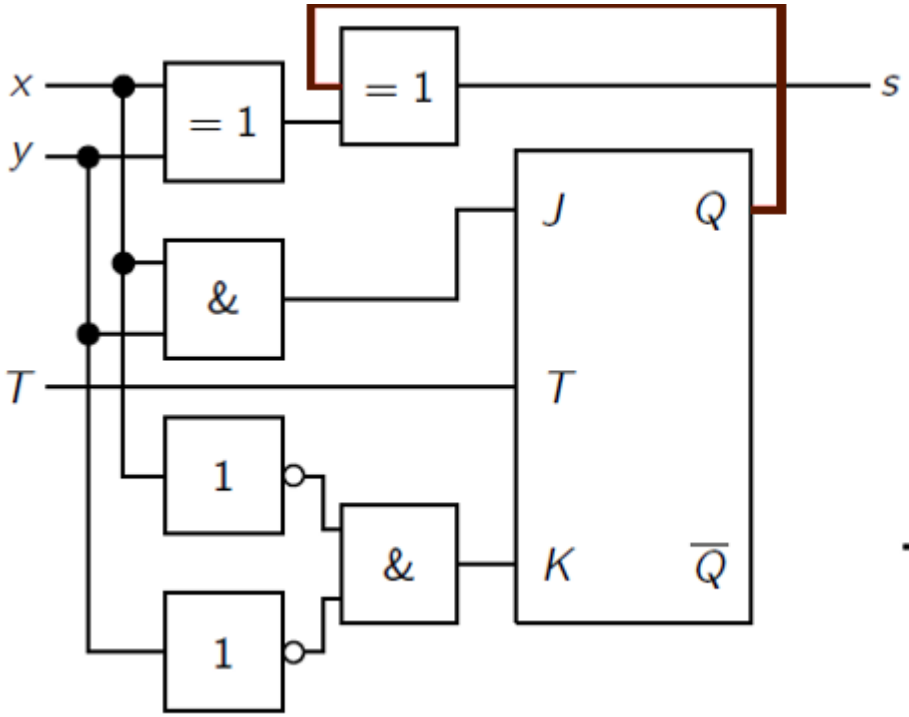


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

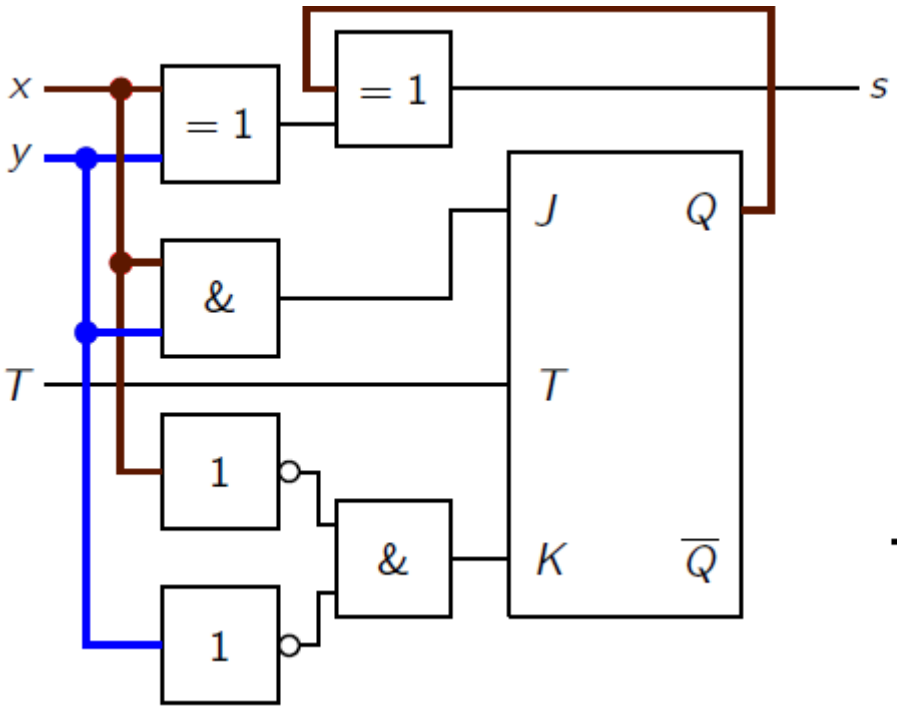


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline \qquad\qquad\ \ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

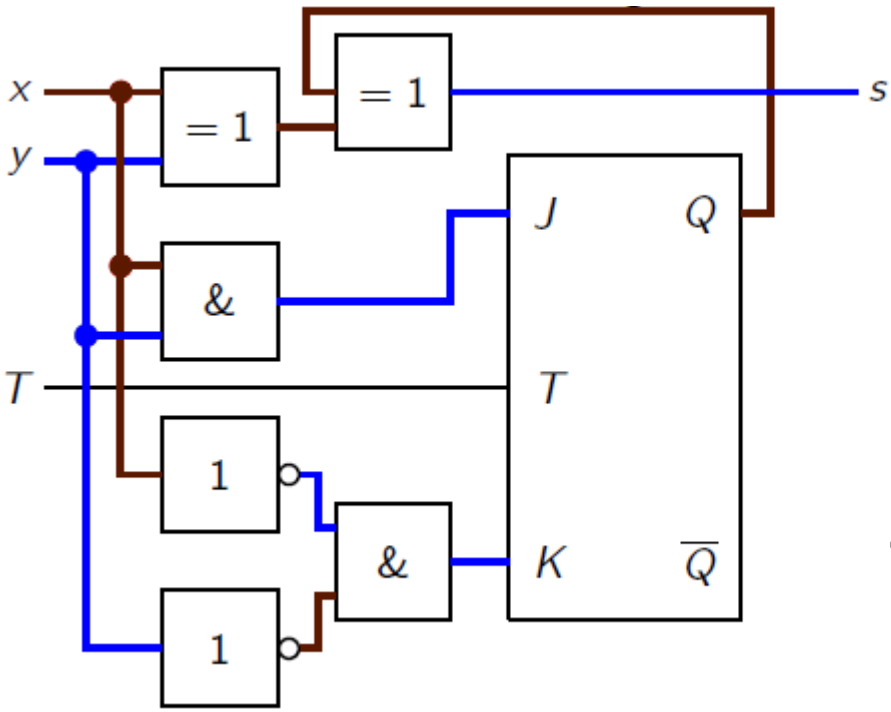


$$\begin{array}{rcccccc} & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & & & & & & 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

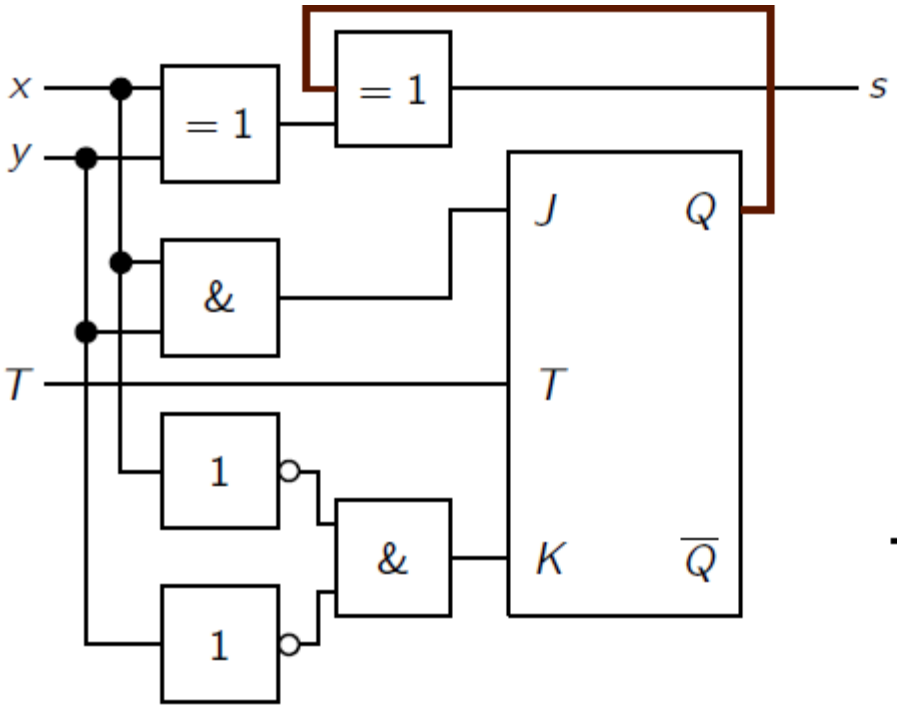


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline \qquad\qquad\ \ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

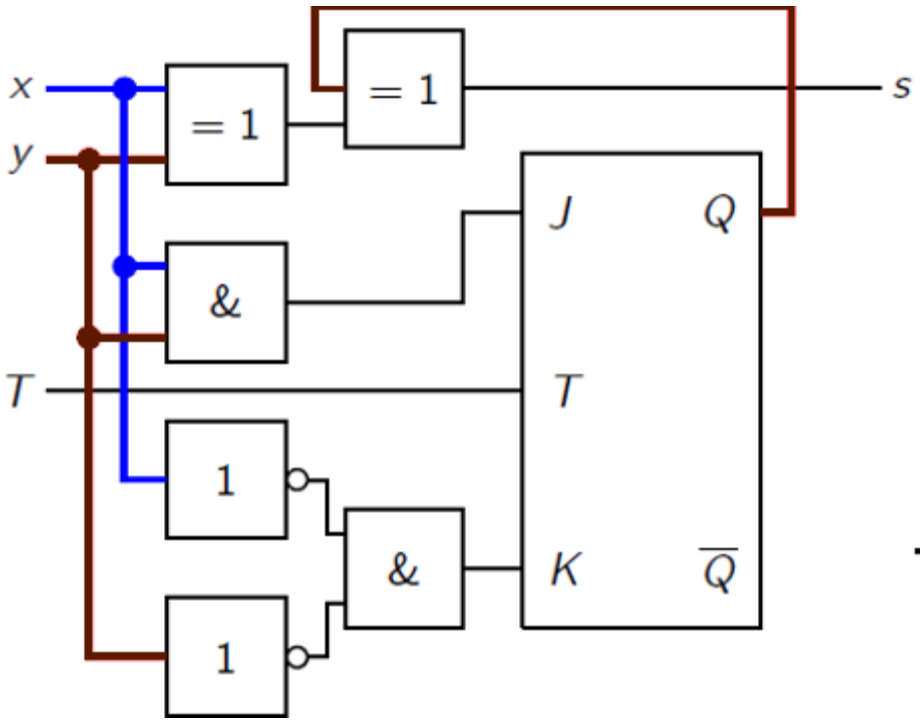


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 0\ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

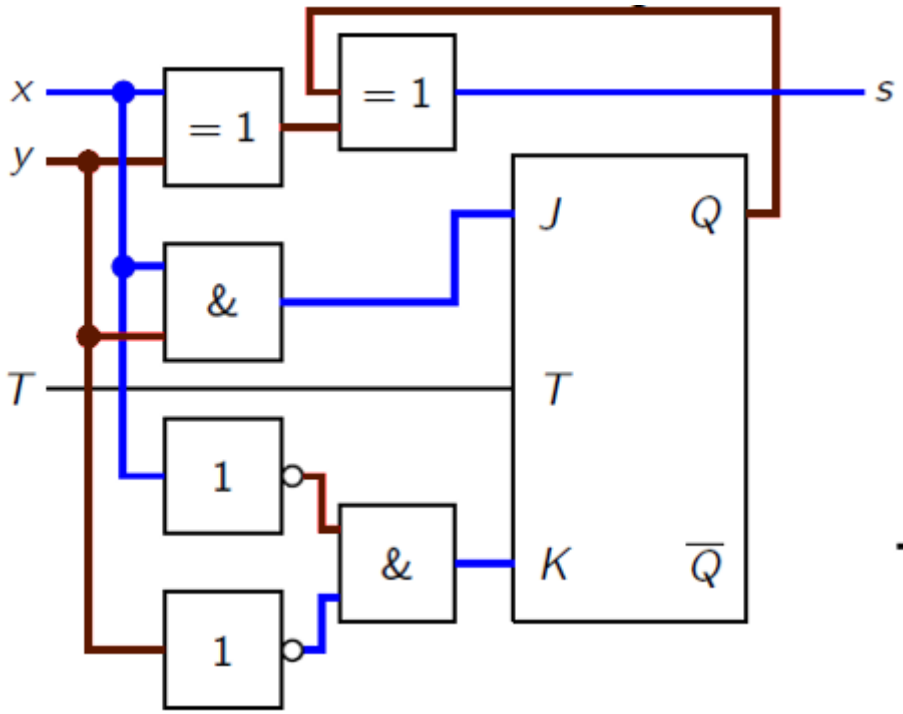


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline \quad\quad\ \ 0\ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

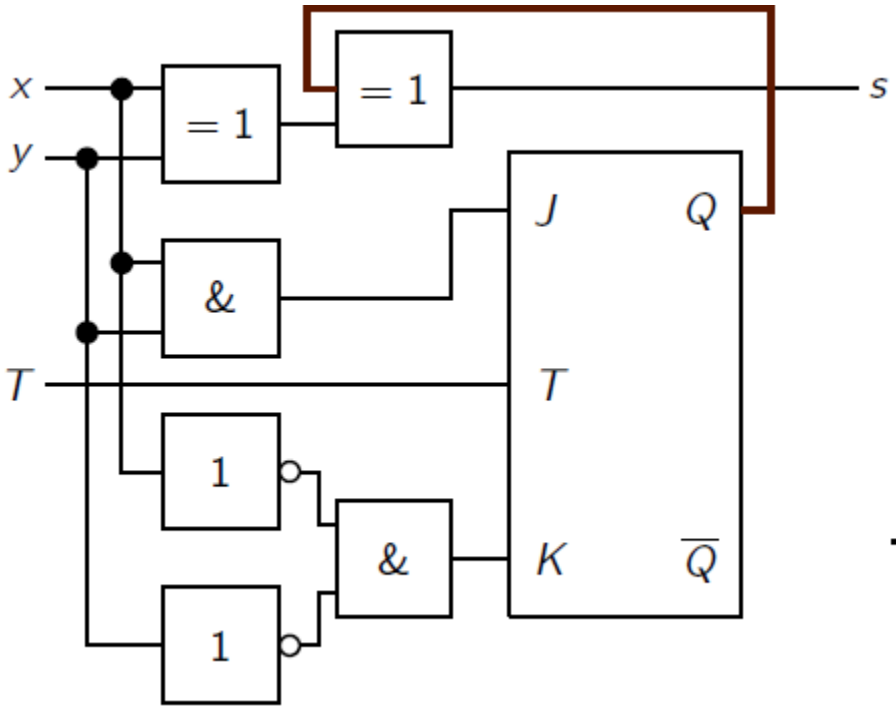


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 0\ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

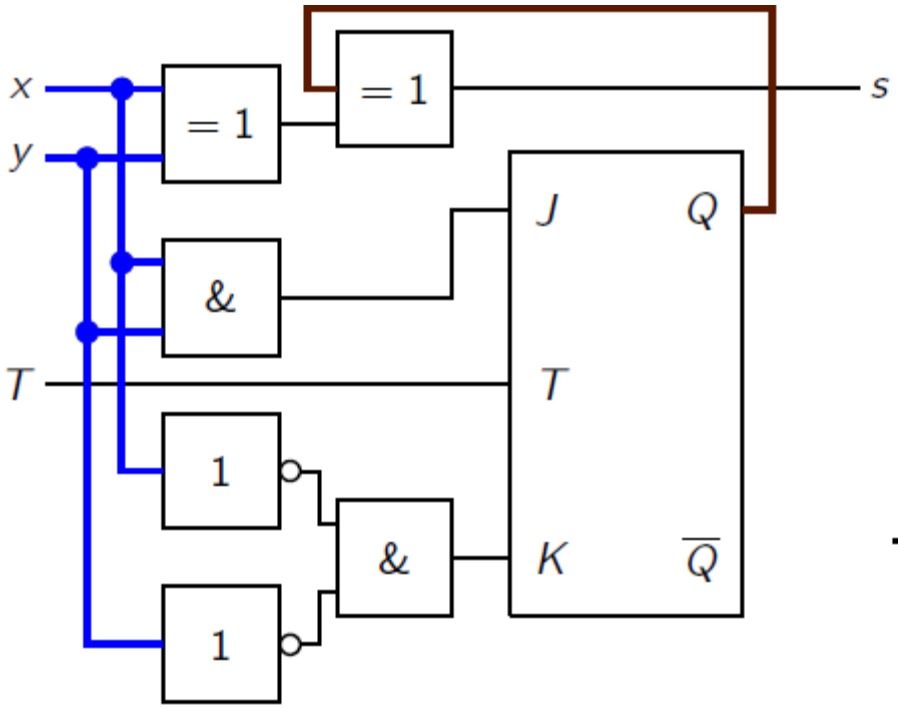


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 0\ 0\ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

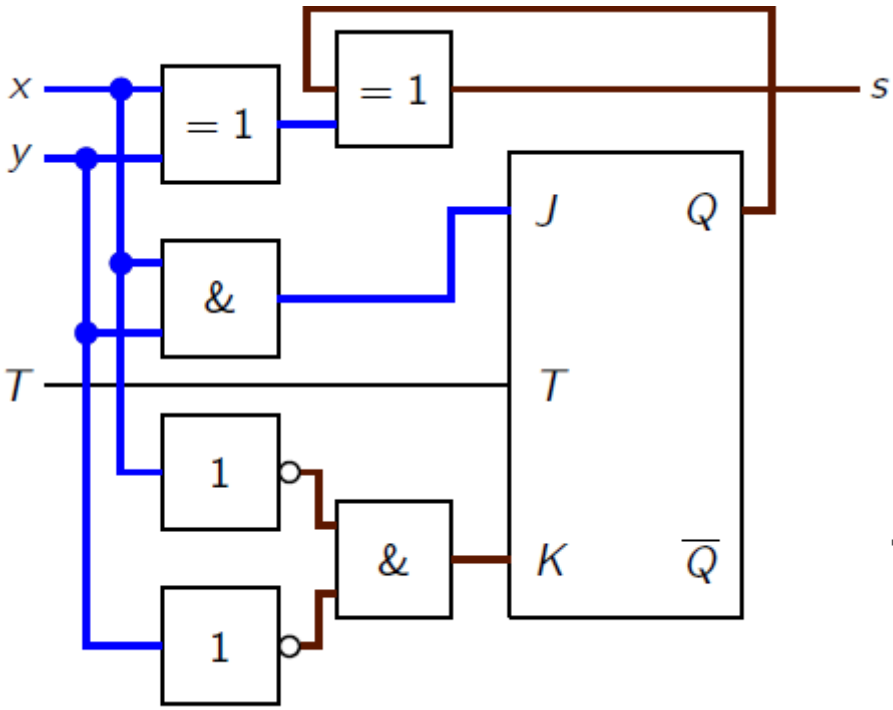


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 0\ 0\ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

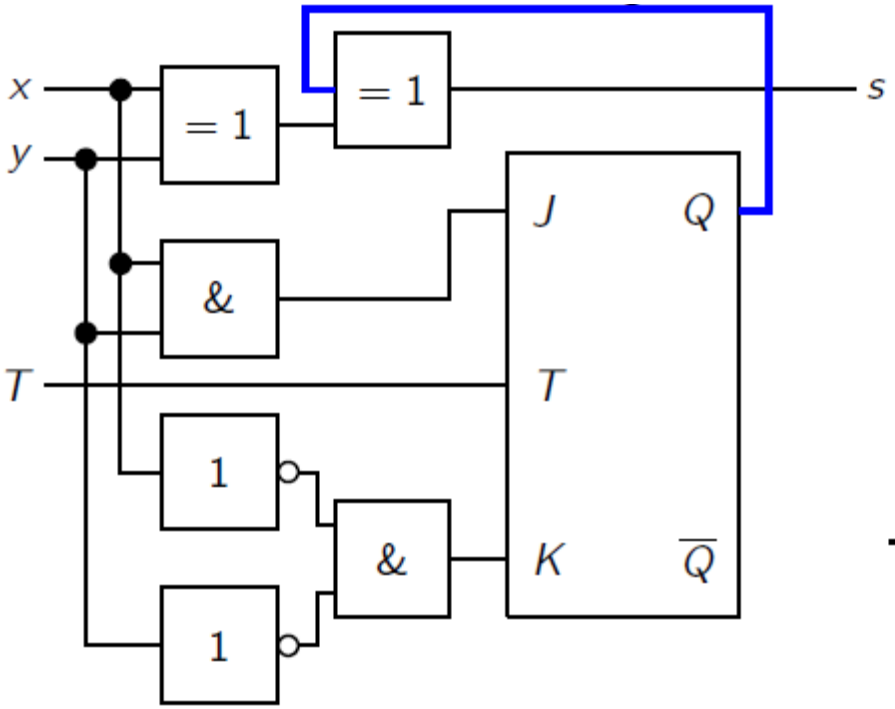


$$\begin{array}{rcccccc} & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & & & & 0 & 0 & 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

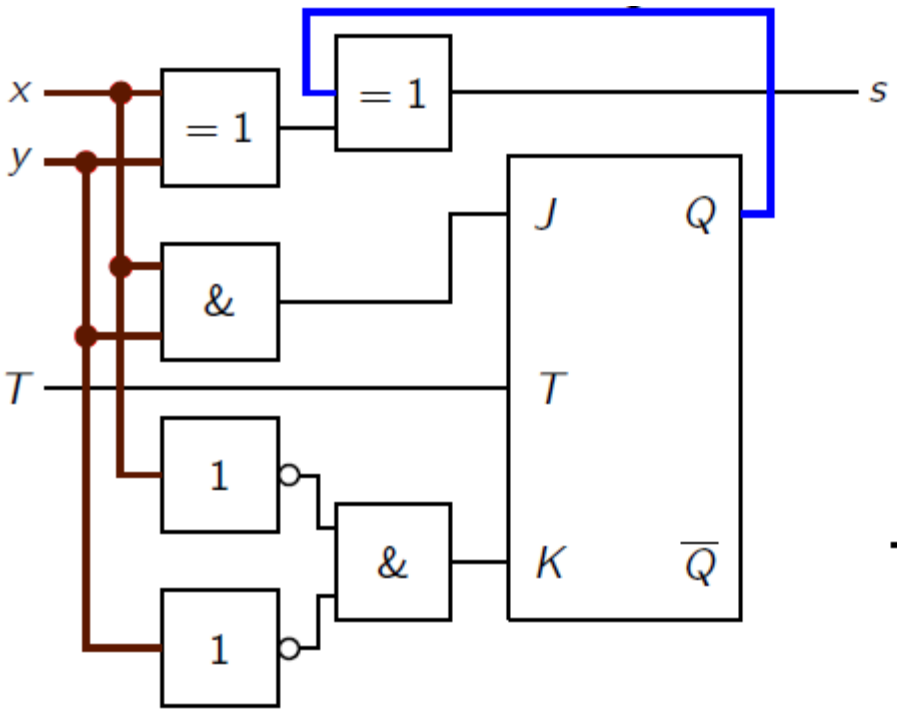


$$\begin{array}{rcccccc} & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & & & 1 & 0 & 0 & 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

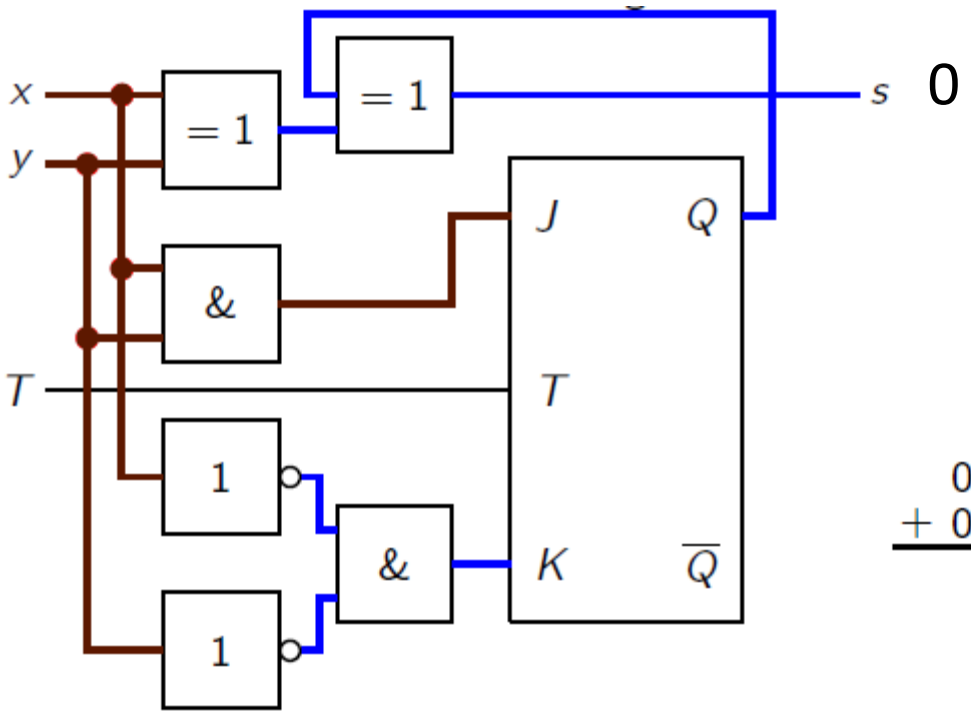


$$\begin{array}{rcccccc} & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & & & 1 & 0 & 0 & 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

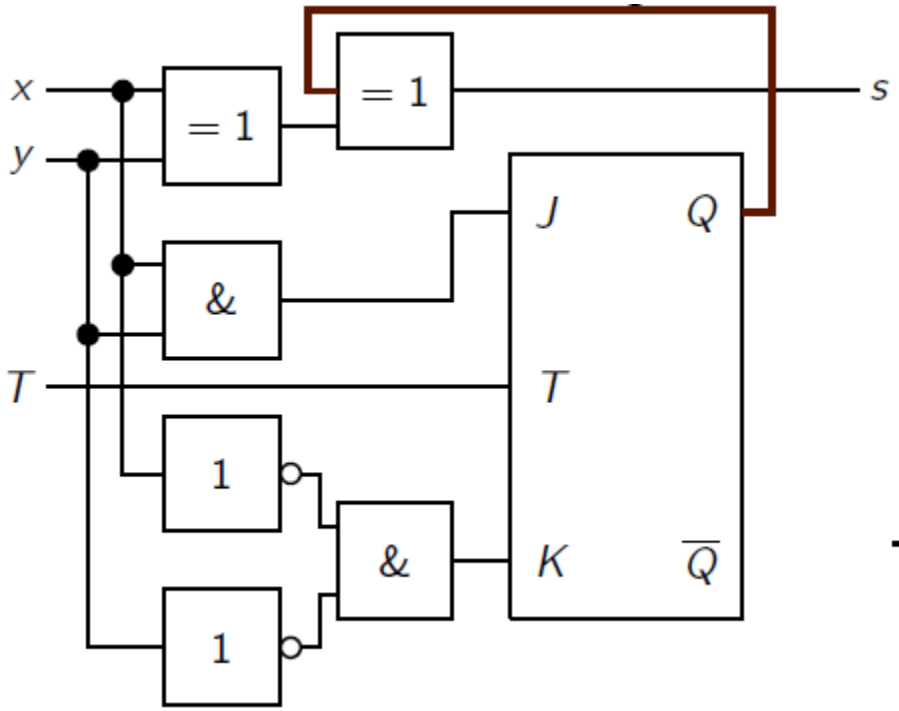


$$\begin{array}{rcccccc} & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & & & 1 & 0 & 0 & 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

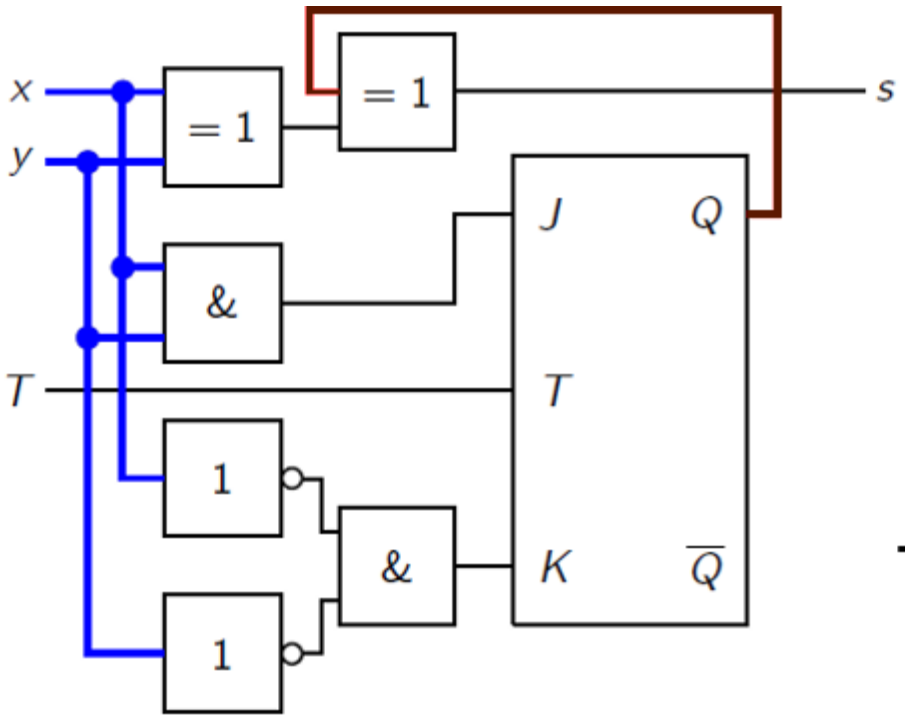


$$\begin{array}{rcccccc} & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & & 0 & 1 & 0 & 0 & 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

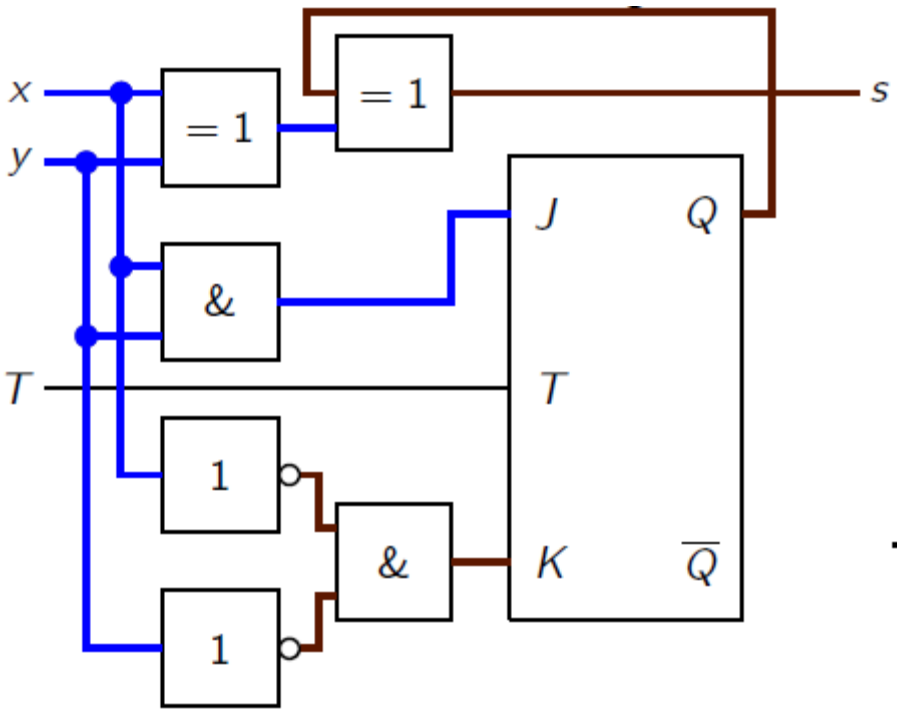


$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 0\ 1\ 0\ 0\ 0 \end{array}$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

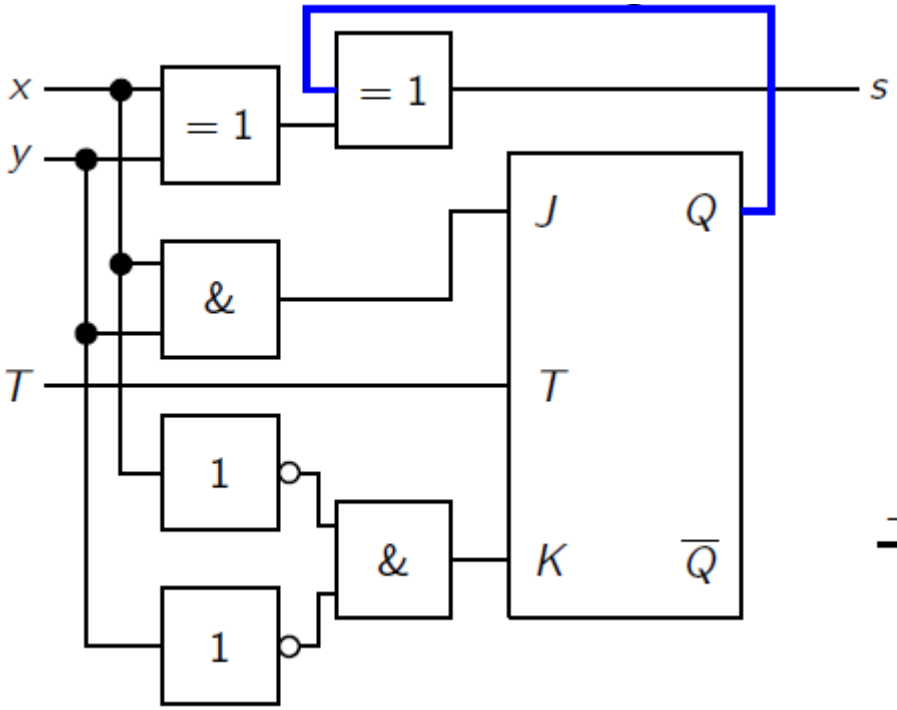


	0	1	0	0	1	1
+	0	1	0	1	0	1
		0	1	0	0	0

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**

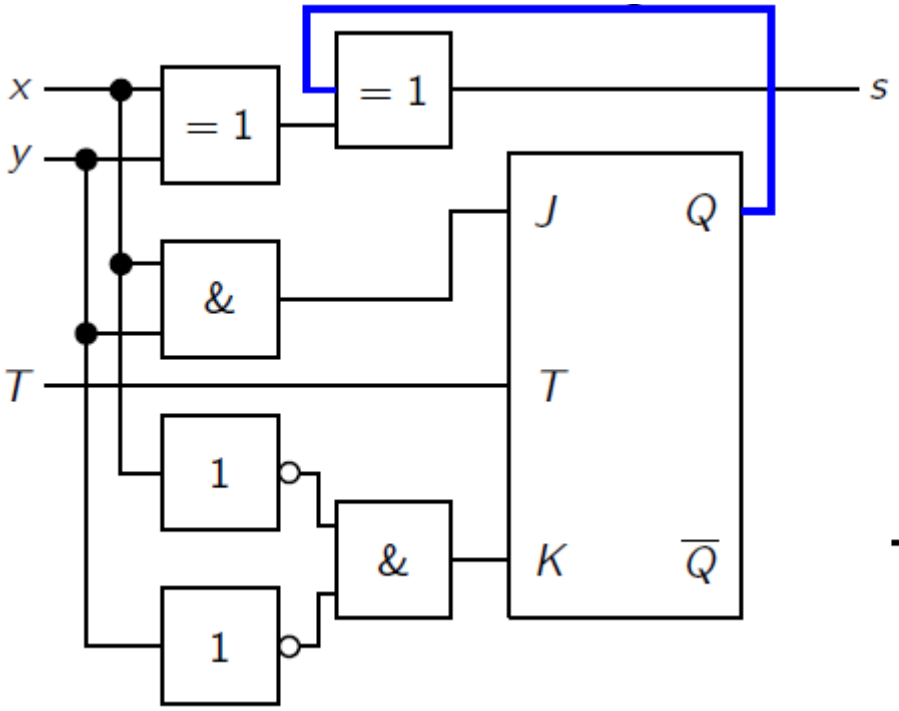
J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$



	0	1	0	0	1	1
+	0	1	0	1	0	1
	1	0	1	0	0	0

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**



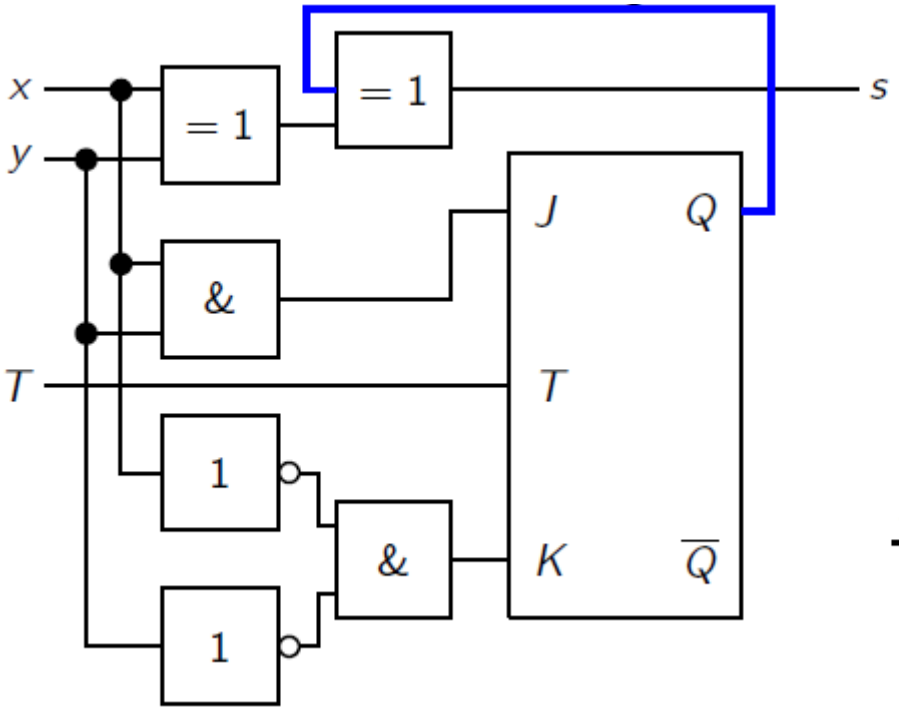
$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Beobachtung Eingabe (0,0) im letzten Takt wichtig

8.5 Serienaddierwerke

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**



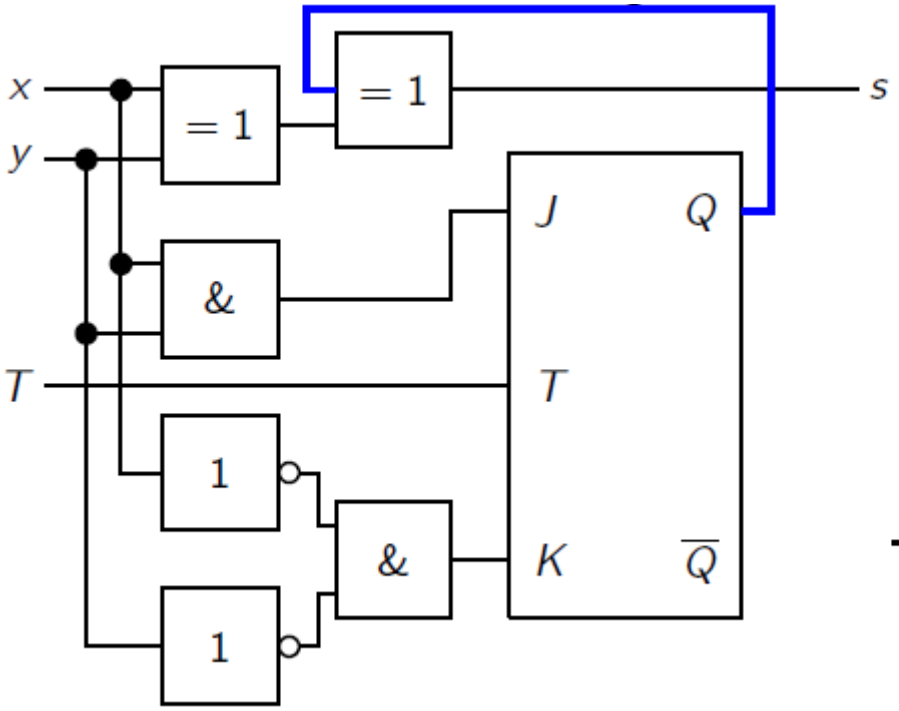
$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Beobachtung Eingabe (0,0) im letzten Takt wichtig

Initialisierung?

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des **vollständigen Schaltwerks**



$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Beobachtung Eingabe (0,0) im letzten Takt wichtig

Initialisierung Eingabe (0,0)

8.5 Serienaddierwerke

Initialisierung Eingabe (0,0)

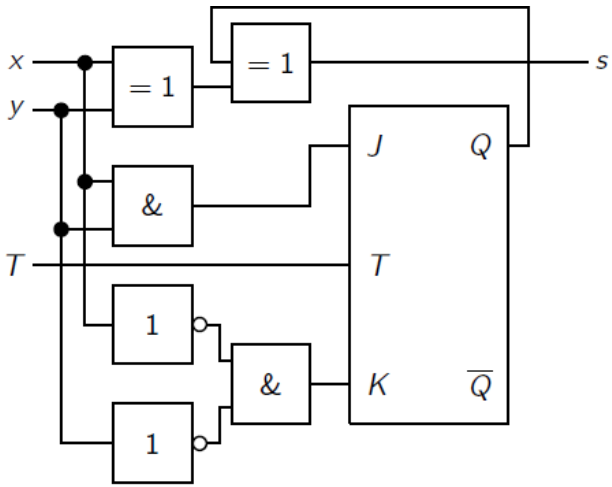
Ziel: Löschen des Übertrags Q_{neu}

- $Q_{neu} = 0$ für (J, K) = (0,*) oder (J, K)=(*,1)
  → (J,K) = (0,1)

- (J,K) = (0,1)
  → (x,y) = (0,0)

Q_{alt}	Q_{neu}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	*
0	0	1	0	1	0	*
0	1	0	0	1	0	*
0	1	1	1	0	1	*
1	0	0	0	1	*	1
1	0	1	1	0	*	0
1	1	0	1	0	*	0
1	1	1	1	1	*	0



8.5 Serienaddierwerke

Unser Addierwerk: Serien-Addierer

Fazit

- addiert beliebig lange Betragszahlen
- ist sehr klein und flach
- ist sehr leicht zu initialisieren
- ist extrem langsam

Warum ist das Schaltwerk so langsam?

bereits bekannt wegen der Überträge

Müssen Überträge so lange dauern?

bekannt im Allgemeinen nicht (siehe Addierer)

aber Bei bit-weiser Eingabe schon!

8.5 Serienaddierwerke

Über unsere Modelle

Wir wissen schon — Überträge werden nur manchmal lange weitergereicht.

Kann man ein Schaltwerk bauen, dass nur manchmal langsam ist?

Problem — unsere Automaten können das zunächst nicht

Beobachtung — bei Mealy- und Moore-Automat bestimmt Länge der Eingabe die Anzahl der Rechentakte

darum Modifikation des Automatenmodells

8.5 Serienaddierwerke

Über unsere Modelle

darum Modifikation des Automatenmodells

neu Erlaube leere Eingabe ε und

signalisiere Ende der Rechnung durch Rechenende-Zeichen

Beobachtung Das ist fundamental neu für uns.

Rechenzeit kann von der Eingabe abhängen

(nicht nur von der Eingabelänge).

8.5 Serienaddierwerke

Auf dem Weg zum besseren Addierwerk

Wie wollen wir vorgehen?

Beobachtung Eingaben müssen sofort ganz zur Verfügung stehen

sonst kann man nicht schneller sein

also Eingabealphabet $\sum = \{0, 1\}^{2n}$

Eingabe $x_{n-1}x_{n-2}\cdots x_1x_0y_{n-1}y_{n-2}\cdots y_1y_0 \in \{0,1\}^{2n}$

interpretieren als

$$\begin{array}{cccccc} & x_{n-1} & x_{n-2} & \cdots & x_0 \\ + & y_{n-1} & y_{n-2} & \cdots & y_0 \\ \hline \end{array}$$

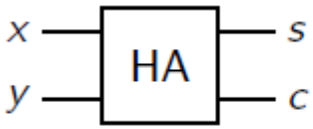
bekannte Idee zur Lösung Ersetze x und y durch x' und y' mit

x + y = x' + y' so lange, bis y' = 0 gilt.

8.5 Serienaddierwerke

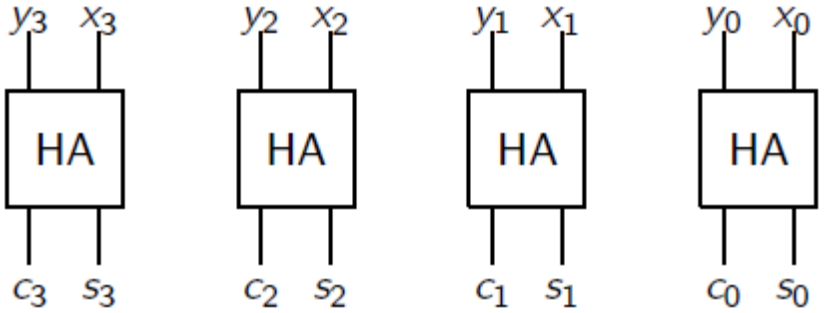
Eine gute Idee verallgemeinern

Erinnerung Wir kennen das schon vom Halbaddierer...



mit $x + y = 2c + s = (c0)_2 + s$

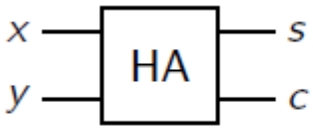
Klar Das funktioniert auch für alle *n* Stellen parallel.



8.5 Serienaddierwerke

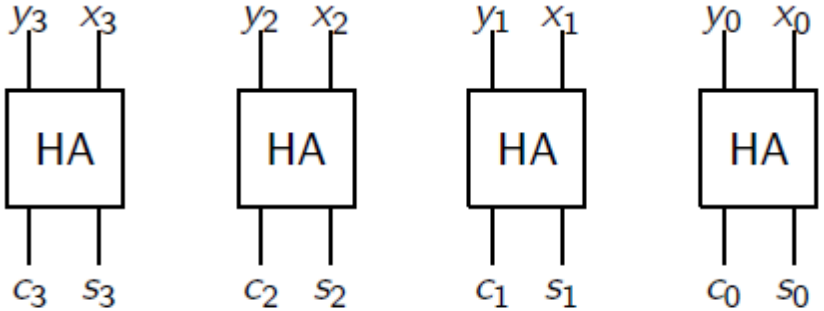
Eine gute Idee verallgemeinern

Erinnerung Wir kennen das schon vom Halbaddierer. . .



mit $x + y = 2c + s = (c0)_2 + s$

Klar Das funktioniert auch für alle n Stellen parallel.



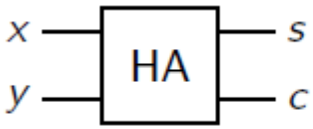
$$x'_3\, x'_2\, x'_1\, x'_0 \;=\; s_3\, s_2\, s_1\, s_0$$
$$\text{Ü}\; y'_3\, y'_2\, y'_1\, y'_0 \;=\; c_3\, c_2\, c_1\, c_0\, 0$$

Fortschritt?

8.5 Serienaddierwerke

Eine gute Idee verallgemeinern

Erinnerung Wir kennen das schon vom Halbaddierer. . .



mit $x + y = 2c + s = (c0)_2 + s$

Klar Das funktioniert auch für alle n Stellen parallel.

Eine gute Idee verallgemeinern

Erinnerung Wir kennen das schon vom Halbaddierer. . .

Klar Das funktioniert auch für alle n Stellen parallel.

Fortschritt? in y hinten „neue“ 0 also nach $\leq n$ Takten $y = 0$

Fortschritt? in y hinten „neue“ 0 also nach $\leq n$ Takten $y = 0$

8.5 Serienaddierwerke

Noch offene Fragen

Was ist mit dem Ü?

klar potenziell kann in jedem Takt vorne ein Überlauf entstehen

Also bis zu n Überläufe speichern?

zum Glück nein

Wir wissen höchstens 1 Überlauf insgesamt

Wann ist die Rechnung fertig?

klar Rechnung fertig , y = 0

Also „done“ d = $\overline{y_0 \vee y_1 \vee \ldots \vee y_{n-1}} = \neg \bigvee_{i=0}^{n-1} y_i$

jetzt unsere Ideen zusammensetzen

8.5 Serienaddierwerke

Das von Neumann-Addierwerk nach John von Neumann (1903–1957)

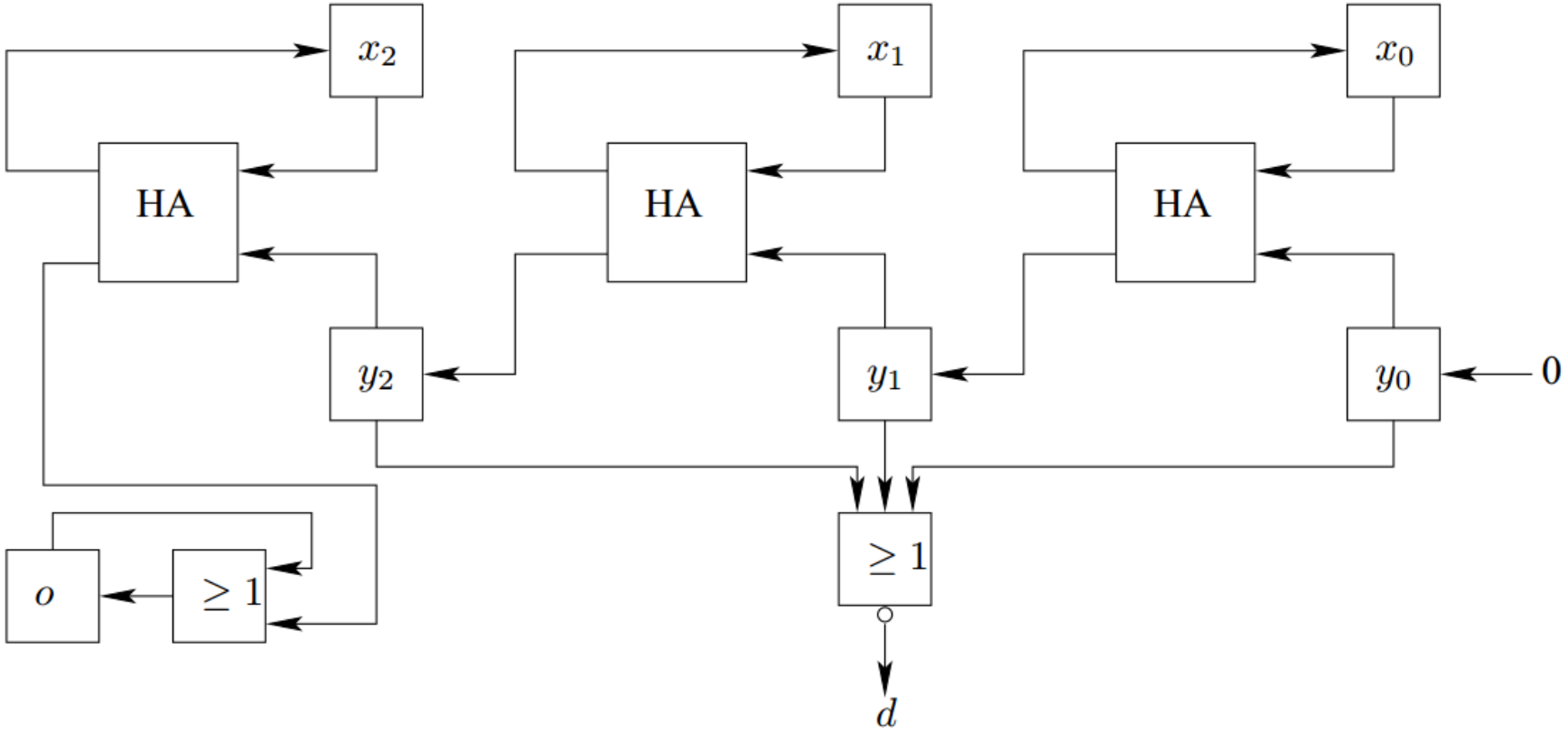
Ideen

- Schaltwerk bekommt direkt Summanden komplett
- Eingabe ε wird erlaubt
- Rechenende-Zeichen signalisiert Ende der Rechnung
- in jedem Takt ersetze x, y durch x', y' mit $x + y = x' + y'$
- Ersetzung stellenweise mit Halbaddierern

Hoffnung Addierwerk ist oft schnell

8.5 Serienaddierwerke

Schematische Darstellung von Neumann-Addierwerk für $n = 3$



Die x_i und y_i und Ü (in der Abbildung mit o bezeichnet) sind Speicher, die mit Flip-Flops realisiert werden

8.5 Serienaddierwerke

Ist das Addierwerk denn jetzt wirklich schnell?

Frage Welche Eingaben dauern lange?

1. Versuch viele Überträge exemplarisch für $n = 6$: 111111 + 111111

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	1	1	1	1	1	1
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

Ist das Addierwerk denn jetzt wirklich schnell?

Frage Welche Eingaben dauern lange?

1. Versuch viele Überträge exemplarisch für $n = 6$: 111111 + 111111

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	1	1	1	1	1	1
	y_5	y_4	y_3	y_2	y_1	y_0

$\Rightarrow$

Ü	x_5	x_4	x_3	x_2	x_1	x_0
1	0	0	0	0	0	0
	1	1	1	1	1	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

Ist das Addierwerk denn jetzt wirklich schnell?

Frage Welche Eingaben dauern lange?

1. Versuch viele Überträge exemplarisch für $n = 6$: 111111 + 111111

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	1	1	1	1	1	1
	y_5	y_4	y_3	y_2	y_1	y_0

$\Rightarrow$

Ü	x_5	x_4	x_3	x_2	x_1	x_0
1	0	0	0	0	0	0
	1	1	1	1	1	0
	y_5	y_4	y_3	y_2	y_1	y_0

$\Rightarrow$

Ü	x_5	x_4	x_3	x_2	x_1	x_0
1	1	1	1	1	1	0
	0	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

Ist das Addierwerk denn jetzt wirklich schnell?

Frage Welche Eingaben dauern lange?

1. Versuch viele Überträge exemplarisch für $n = 6$: 111111 + 111111

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	1	1	1	1	1	1
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
1	0	0	0	0	0	0
	1	1	1	1	1	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
1	1	1	1	1	1	0
	0	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

Beobachtung nach nur zwei Takten fertig → **schnell**

Erinnerung bzgl. Übertrag schwierig: 0 + 1, 1 + 0

8.5 Serienaddierwerke

Ist das Addierwerk denn jetzt wirklich schnell?

Frage Welche Eingaben dauern lange?

2. Versuch 0+1, 1+0 exemplarisch für $n = 6$: 000000 + 111111

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	0	0	0	0	0	0
	1	1	1	1	1	1
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

Ist das Addierwerk denn jetzt wirklich schnell?

Frage Welche Eingaben dauern lange?

2. Versuch 0+1, 1+0 exemplarisch für $n = 6$: 000000 + 111111

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	0	0	0	0	0	0
	1	1	1	1	1	1
	y_5	y_4	y_3	y_2	y_1	y_0

$\Rightarrow$

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

Ist das Addierwerk denn jetzt wirklich schnell?

Frage Welche Eingaben dauern lange?

2. Versuch 0+1, 1+0 exemplarisch für $n = 6$: 000000 + 111111

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	0	0	0	0	0	0
	1	1	1	1	1	1
	y_5	y_4	y_3	y_2	y_1	y_0

$\Rightarrow$

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

Beobachtung nach nur einem Takten fertig → **sehr schnell**

Offensichtlich 111111 + 000000 wäre sofort fertig

Vermutung Überträge mit langen Wegen dauern lange

Frage Was ist die "**worst case**" Eingabe?

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	0
	0	0	0	0	**1**	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	0
	0	0	0	0	**1**	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	0	0
	0	0	0	**1**	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	0
	0	0	0	0	**1**	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	0	0
	0	0	0	**1**	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	0	0	0
	0	0	**1**	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	0
	0	0	0	0	**1**	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	0	0
	0	0	0	**1**	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	0	0	0
	0	0	**1**	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	0	0	0	0
	0	**1**	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	0
	0	0	0	0	**1**	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	0	0
	0	0	0	**1**	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	0	0	0
	0	0	**1**	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	0	0	0	0
	0	**1**	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	0	0	0	0	0
	1	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	0
	0	0	0	0	**1**	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	0	0
	0	0	0	**1**	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	0	0	0
	0	0	**1**	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	0	0	0	0
	0	**1**	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	0	0	0	0	0
	1	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
1	0	0	0	0	0	0
	0	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

8.5 Serienaddierwerke

"**worst case**" Eingabe für das von Neumann Addierwerk

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	1
	0	0	0	0	0	**1**
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	1	0
	0	0	0	0	**1**	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	1	0	0
	0	0	0	**1**	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	1	0	0	0
	0	0	**1**	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	1	0	0	0	0
	0	**1**	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
0	1	0	0	0	0	0
	1	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

⇒

Ü	x_5	x_4	x_3	x_2	x_1	x_0
1	0	0	0	0	0	0
	0	0	0	0	0	0
	y_5	y_4	y_3	y_2	y_1	y_0

extrem langsam

8.5 Serienaddierwerke

Analyse des von Neumann-Addierwerks

Beobachtung $111 \cdots 111 + 000 \cdots 000 \Longrightarrow$ 0 Takte **extrem schnell**
Beobachtung $111 \cdots 111 + 000 \cdots 001 \Longrightarrow$ n Takte extrem langsam

Ist das von Neumann-Addierwerk schnell oder langsam?

- Best Case schnell (0 Takte)
- Worst Case langsam (n Takte)
- 0 Takte Glücklicher Zufall?
- n Takte Seltenes Unglück?

Was ist typisch?

- Antwort Average-Case-Analyse
- d. h. durchschnittliche Rechenzeit
- Problem: Welche Verteilung über die Eingaben nehmen wir an?

8.5 Serienaddierwerke

Average-Case-Analyse

Average-Case-Rechenzeit **durchschnittliche Rechenzeit**

- bei Gewichtung der Eingaben
- nach ihrer Wahrscheinlichkeit bei gegebener
- Wahrscheinlichkeitsverteilung über den Eingaben

Problem Welche Verteilung ist angemessen?

- meist betrachtet **Gleichverteilung**
- **Vorsicht:** reale Daten fast nie gleichverteilt
- trotzdem hier Annahme Gleichverteilung

Gleichverteilung bedeutet

- alle Eingaben gleichwahrscheinlich
- an jeder Position 0 und 1 mit Wahrscheinlichkeit ½
- alle Positionen unabhängig

8.5 Serienaddierwerke

Average-Case-Analyse (**ungenau!**)

Start *Typische* Eingabe der Länge *2n* bei Gleichverteilung:

- je *50%* 0- und 1-Stellen
- d.h. *n/2* 1-en pro Summand

Verarbeitung: nach 1 Takt ...

- **Überträge (y)**: Wir erwarten 1 an y_{i+1}, wenn vorher 1 sowohl an x_i als auch an y_i, d.h. mit Wahrscheinlichkeit *1/4*
  $\Rightarrow$ **Erwarten nach 1 Takt *n/4* 1-en in *y***
- **Ergebnis (x)**: Wir erwarten 1 in x_i, wenn vorher 01 oder 10 in x_i, y_i, d.h. in 50% der Fälle
  $\Rightarrow$ **Erwarten nach 1 Takt weiterhin *n/2* 1-en in x**
- $\Rightarrow$ **Erwarten**: **Anzahl 1-en in y halbiert sich je Takt (in x unverändert)**

Fazit: Erwarten Rechenende nach $\log_2(n)$ Takten

8.5 Serienaddierwerke

Moore-Automat zum von Neumann-Addierwerk

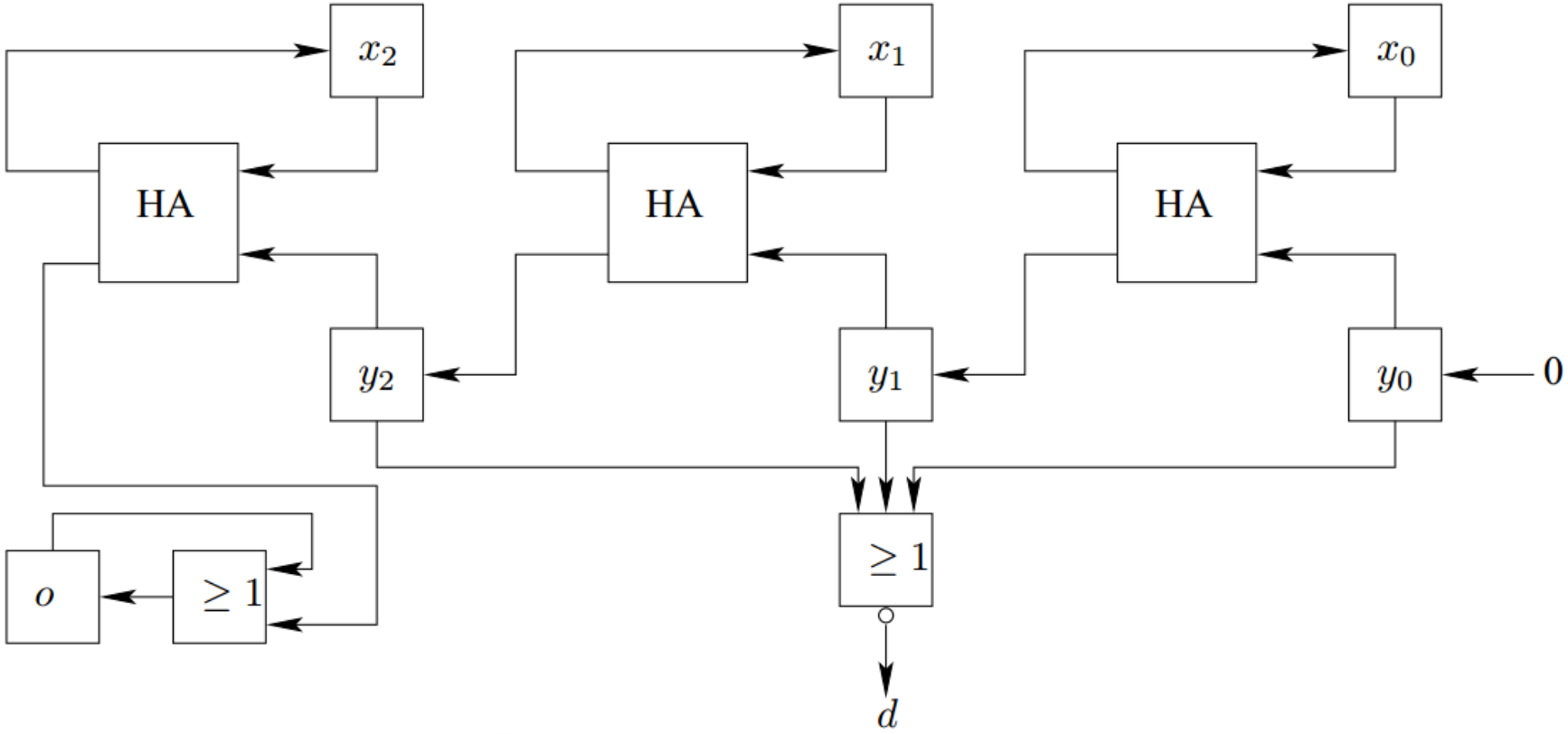
Ziel Beschreibung eines Moore-Automaten zum von Neumann-Addierwerk zur Addition von zwei n-Bit-Zahlen

Erweiterung des Automatenmodells um **ε-Eingaben** im folgenden Sinn

- bei Eingabe von xy wird aktuelle **Rechnung unterbrochen** und mit **Berechnung** von $x + y$ **begonnen**
- bei Eingabe von ε wird aktuelle **Rechnung fortgesetzt**
- Ausgabe ist "–", falls Rechnung noch **nicht beendet**
- Ausgabe ist Summe der Eingabezahlen, wenn **fertig berechnet**

8.5 Serienaddierwerke

Schematische Darstellung von Neumann-Addierwerk für $n = 3$



8. Synchrone Schaltwerke

1. Einleitung ✓
2. Bistabile Kippstufe ✓
3. Automaten ✓
4. Synchrone Schaltwerke ✓
5. **Serienaddierwerke** ✓
6. Speicher & Schieberegister
7. Takt

8.6 Speicher & Schieberegister

Speicher

Erinnerung Wir können in einem Flip-Flop ein Bit speichern.

Beobachtung

- 1-Bit-Speicher reichen uns nicht.
- Wir können in *k* Flips-Flops *k* Bits speichern.

Wie **organisieren** wir das so, dass der Zugriff bequem ist?

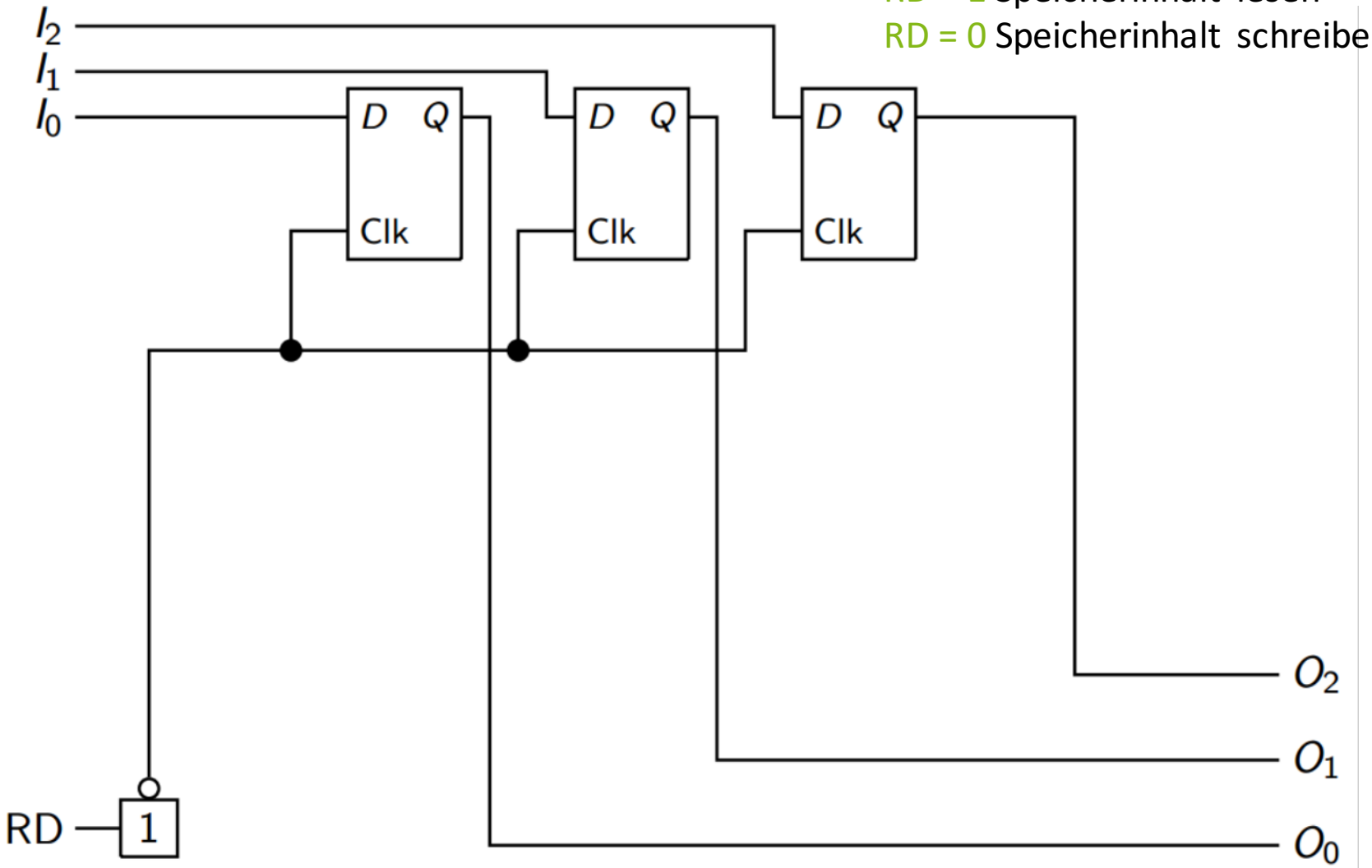
- exemplarisch Speicher für 3-Bit-Wörter
- Warum 3? Passt bequem auf eine Folie

Anmerkung Auch 3-Bit-Speicher reicht in der Praxis nicht aus.

8.6 Speicher & Schieberegister

Speichern eines 3-Bit-Wortes

RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben

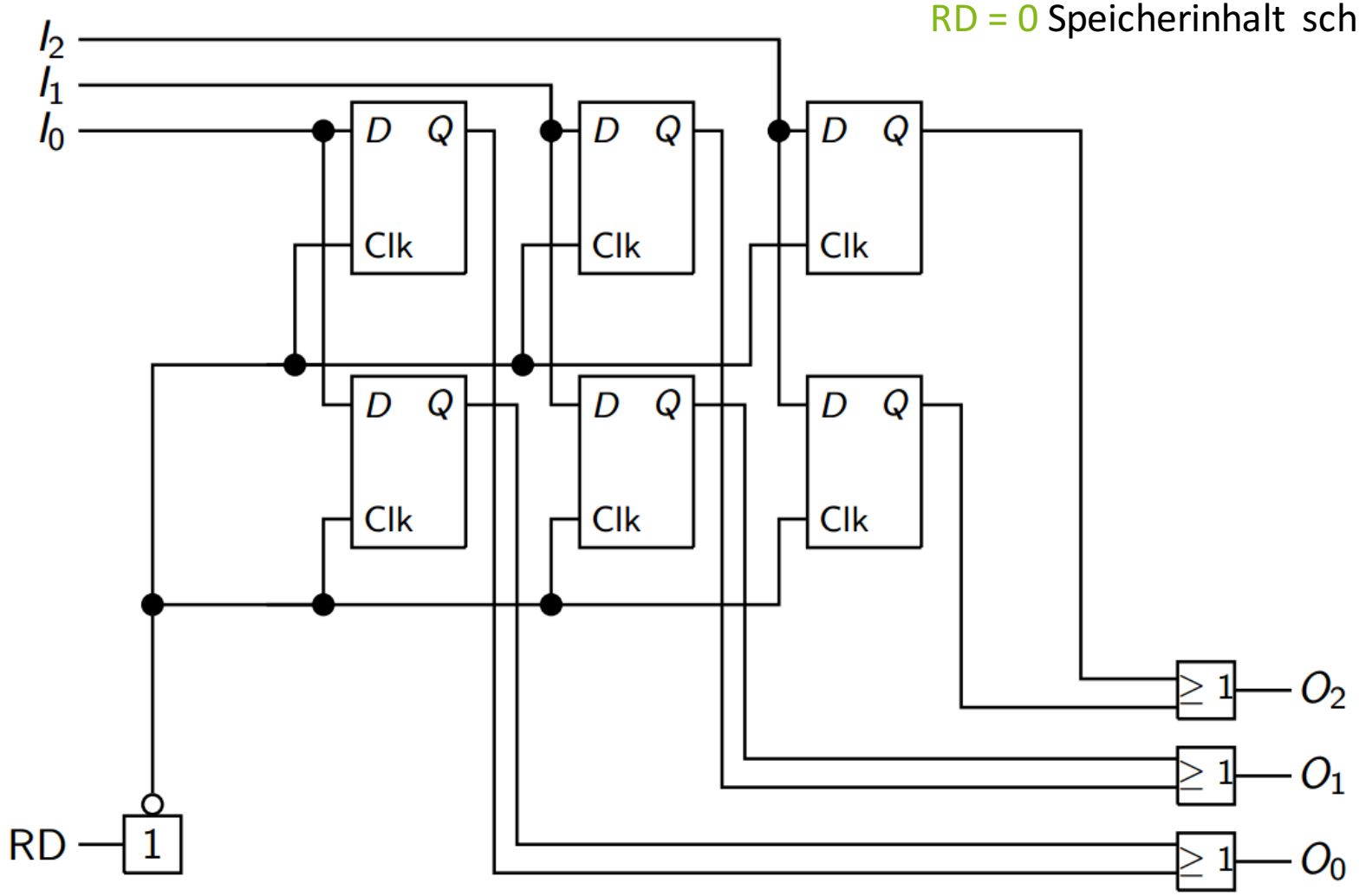


Wie können wir mehr als nur ein Wort speichern?

8.6 Speicher & Schieberegister

Speichern von zwei 3-Bit-Worten

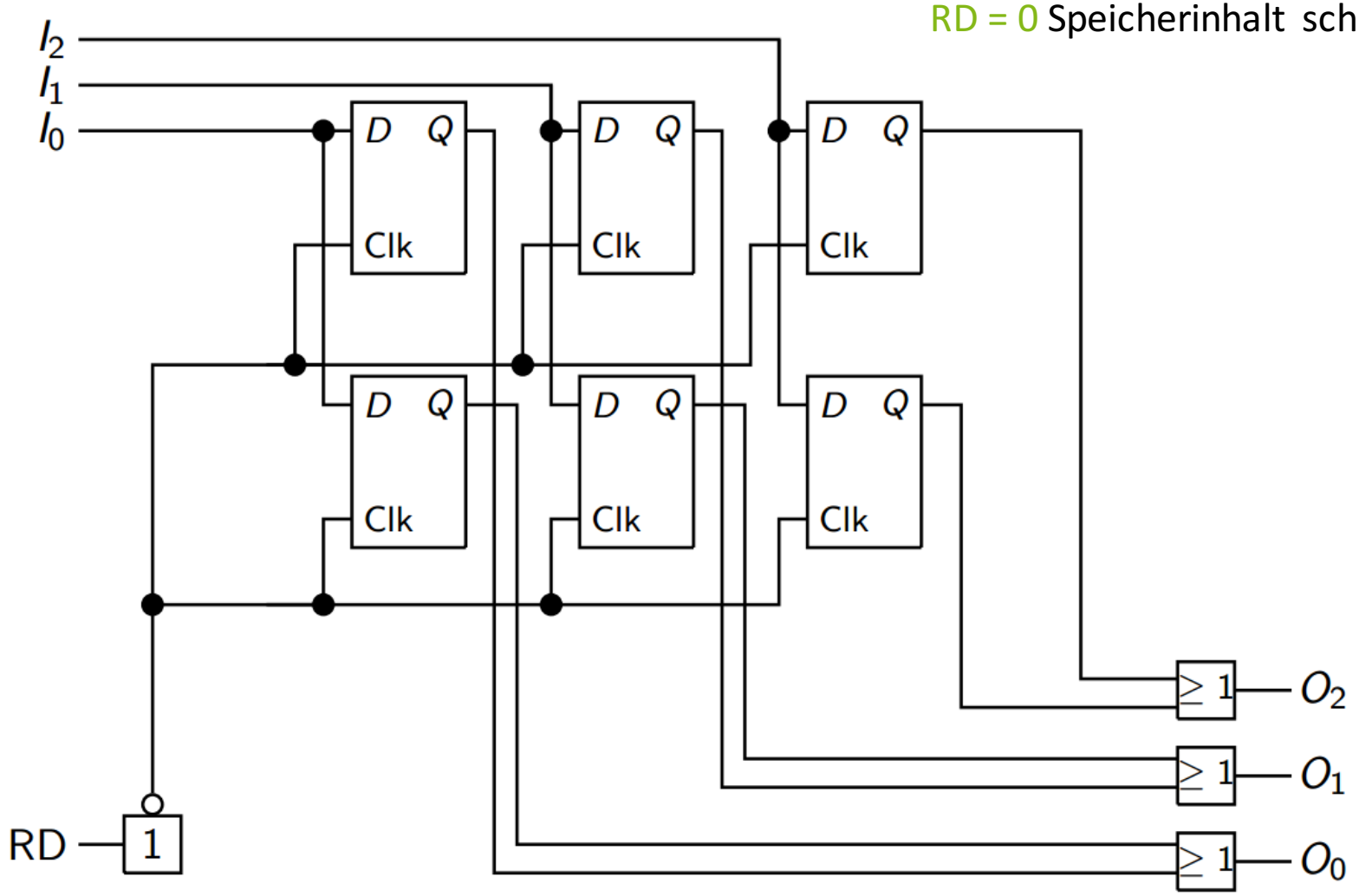
RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



8.6 Speicher & Schieberegister

Speichern von zwei 3-Bit-Worten

RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



Beobachtung: Oberes und unteres Wort stets identisch

8.6 Speicher & Schieberegister

Speichern von zwei 3-Bit-Worten

Offensichtlich Die Speicherwörter müssen getrennt ansprechbar sein

- wir wollen ein Wort wahlfrei adressieren.
- wir brauchen eine Adressleitung A0.

Interpretation

- A0 = 0 Wort w0 adressiert zum Lesen oder Schreiben
- A0 = 1 Wort w1 adressiert zum Lesen oder Schreiben

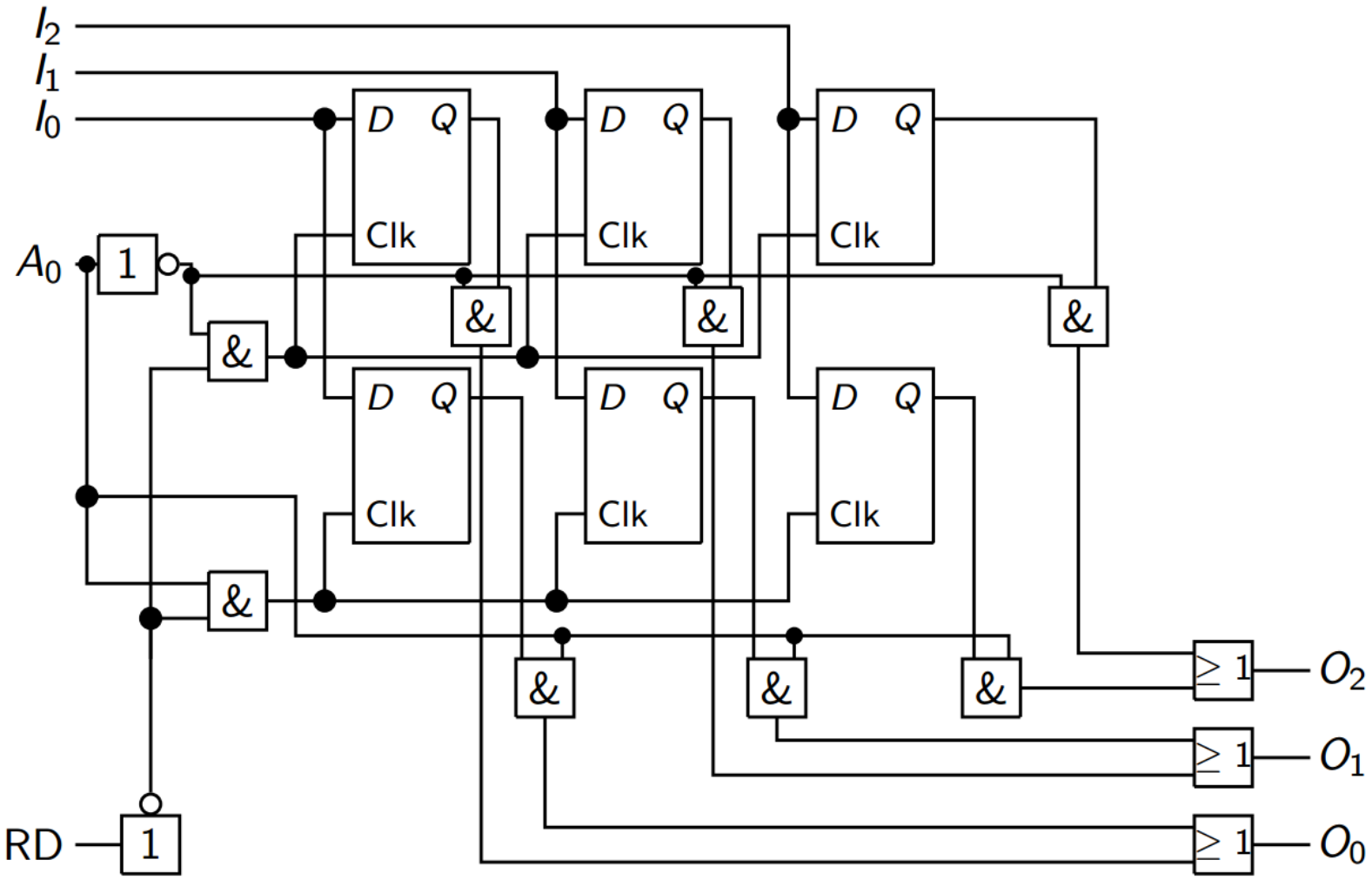
RD bestimmt, ob gelesen oder geschrieben wird

- RD = 1 Speicherinhalt lesen
- RD = 0 Speicherinhalt schreiben

8.6 Speicher & Schieberegister

Speichern von zwei 3-Bit-Worten

RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



8.6 Speicher & Schieberegister

Speichererweiterungen

angenommen Schaltung zur Speicherung von zwei 3-Bit-Wörtern existiert

angenommen Speicher reicht im Betrieb nicht aus

Wunsch Speichererweiterung

Beachte Speicher**erweiterung** bedeutet

- weiteren Speicherbaustein (gleicher Art) hinzufügen
- nicht vorhandenen Speicherbaustein durch größeren Speicherbaustein ersetzen

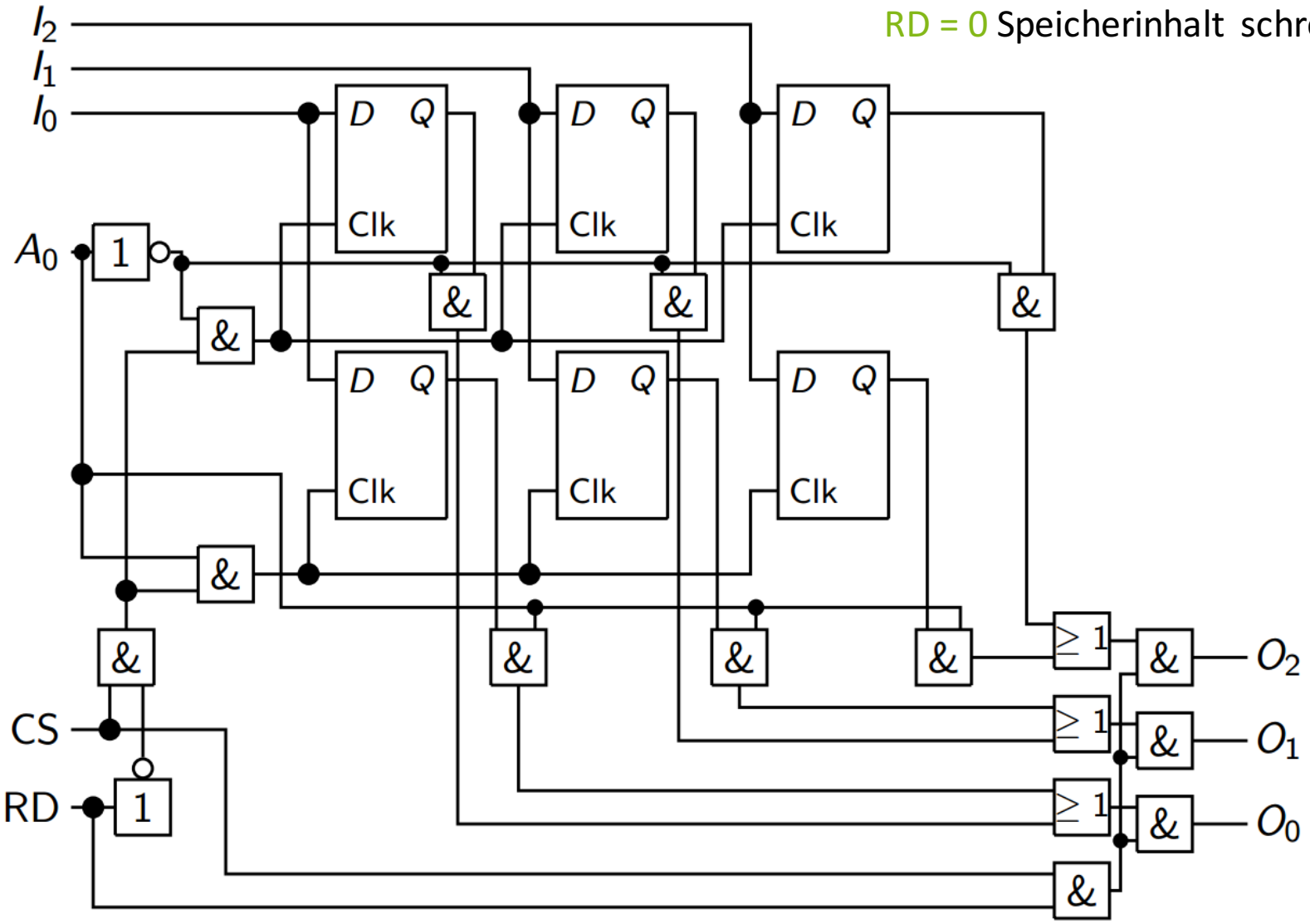
Wie können wir das unterstützen?

- Idee zusätzliche Eingabe
- Chip Selected (CS)

8.6 Speicher & Schieberegister

Speichern von zwei 3-Bit-Worten

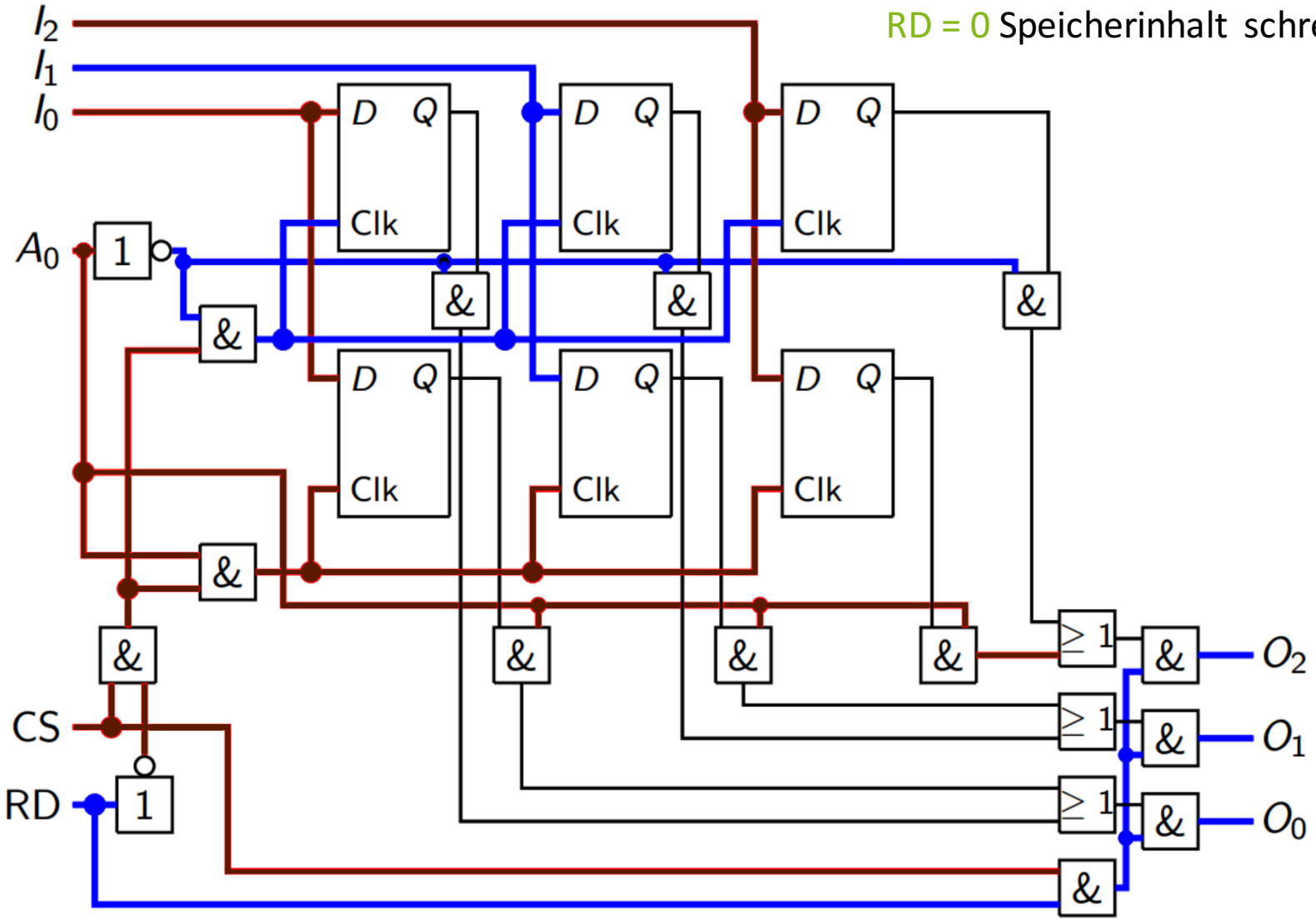
RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



8.6 Speicher & Schieberegister

Beispiel w1 := 1 0 1

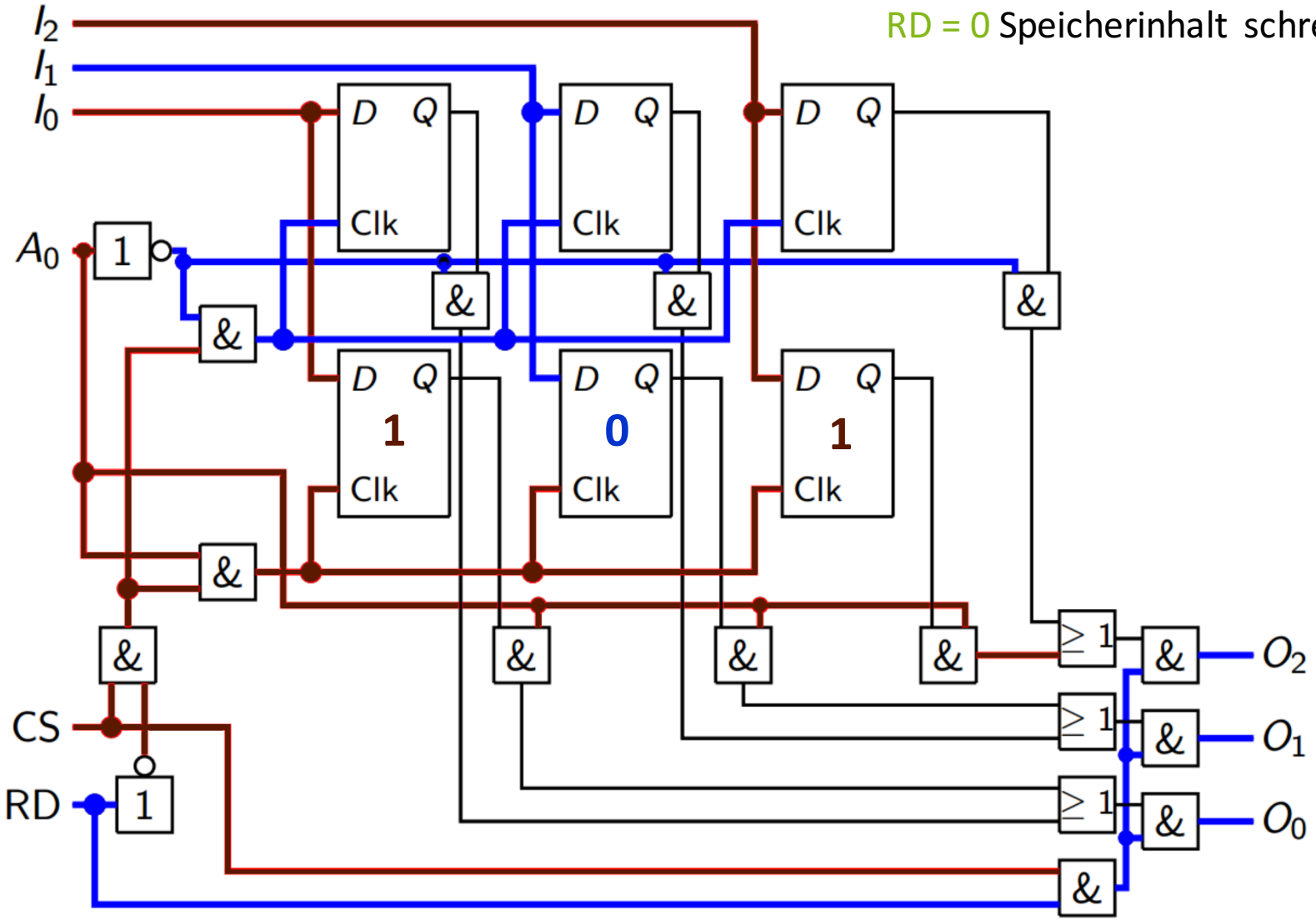
RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



8.6 Speicher & Schieberegister

Beispiel w1 := 1 0 1

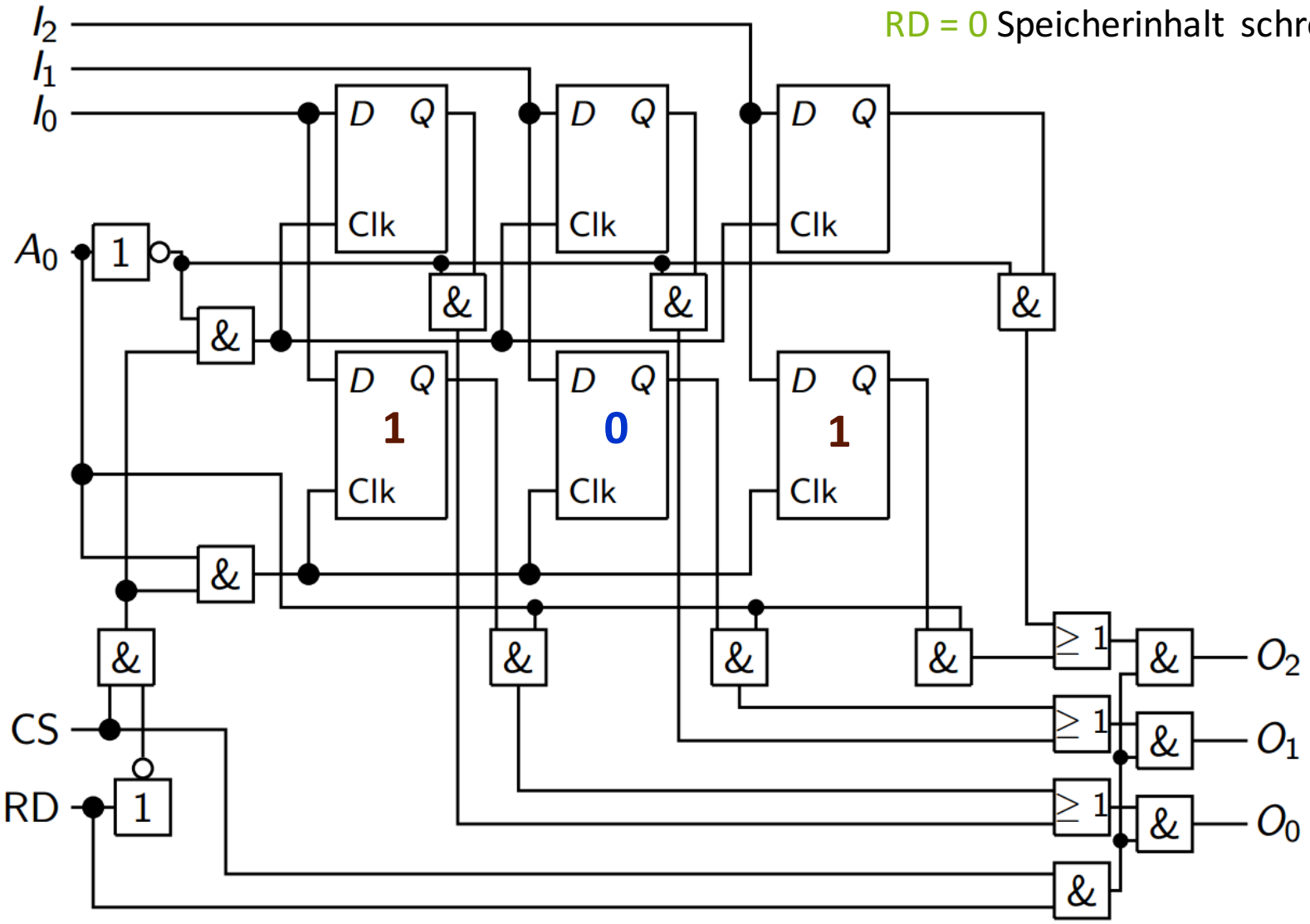
RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



8.6 Speicher & Schieberegister

Beispiel Lies w1

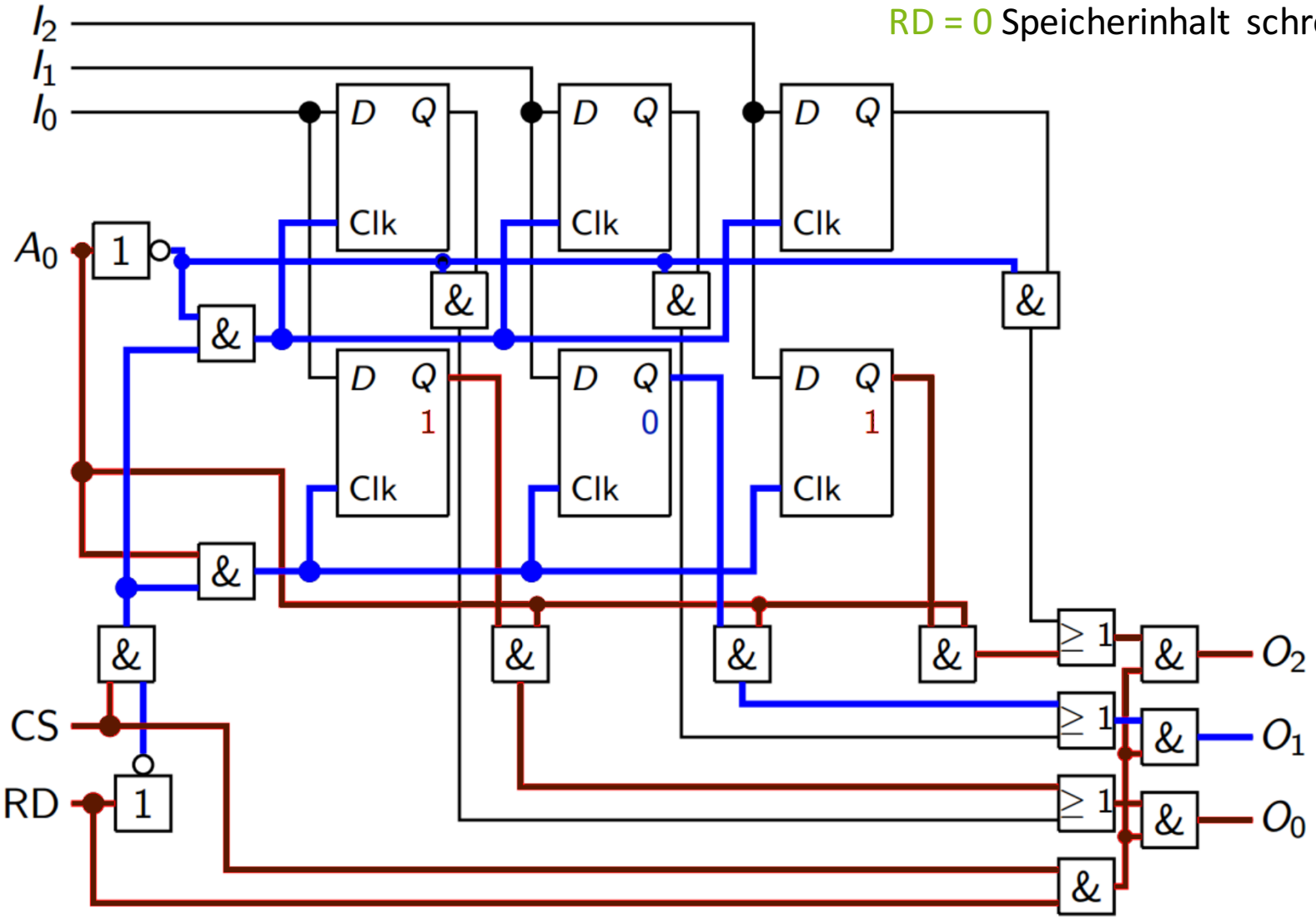
RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



8.6 Speicher & Schieberegister

Beispiel Lies w1

RD = 1 Speicherinhalt lesen
RD = 0 Speicherinhalt schreiben



8.6 Speicher & Schieberegister

Realistischere Speichergrößen

Beobachtung Speichergröße "2 Wörter" ist nicht realistisch

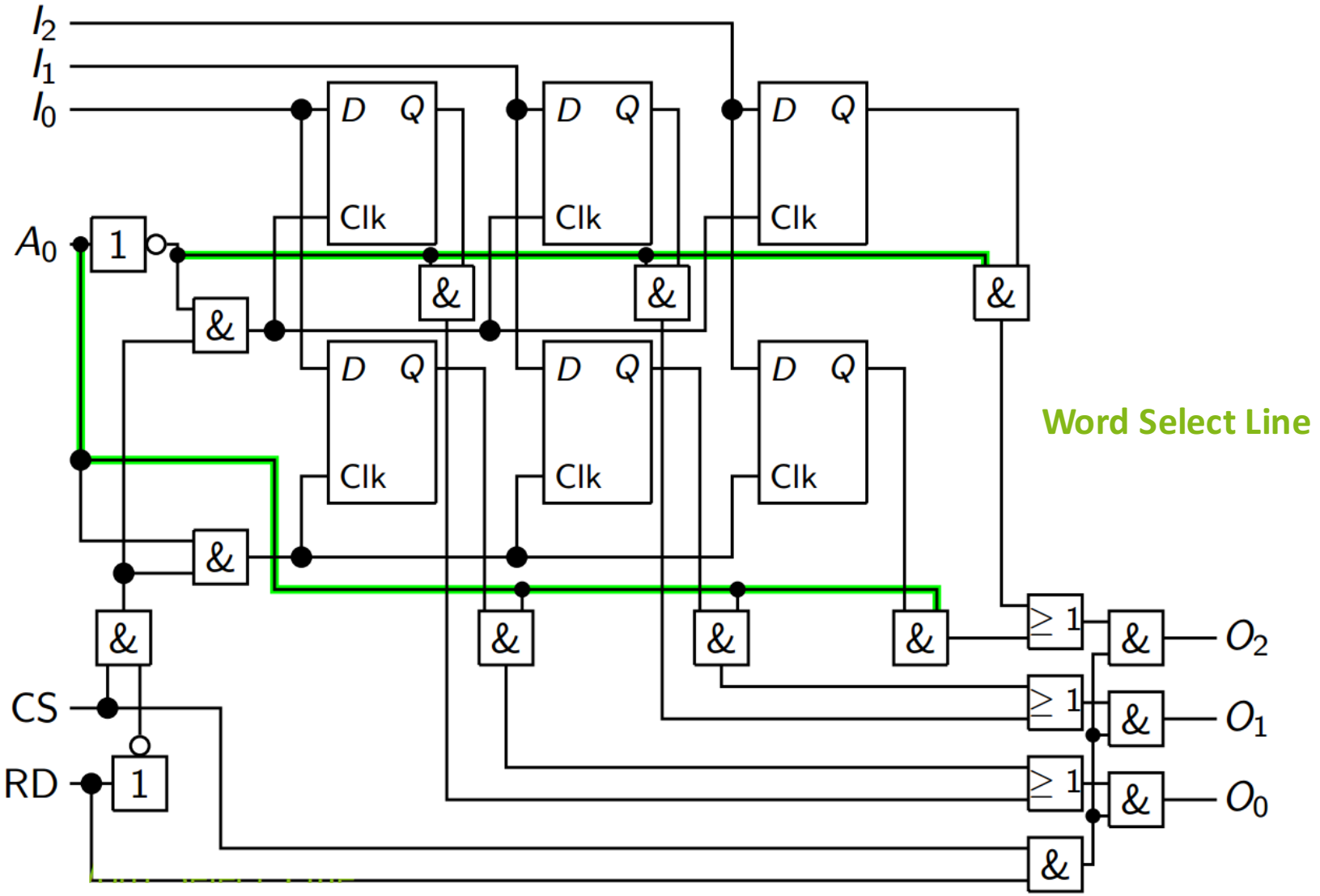
auch nicht bei Benutzung mehrerer Bausteine

Wie kommen wir zu **realistischen Speichergrößen**?

- größere Wortlänge funktioniert im Prinzip gleich
- Wie verallgemeinern wir auf > 2 Wörter?

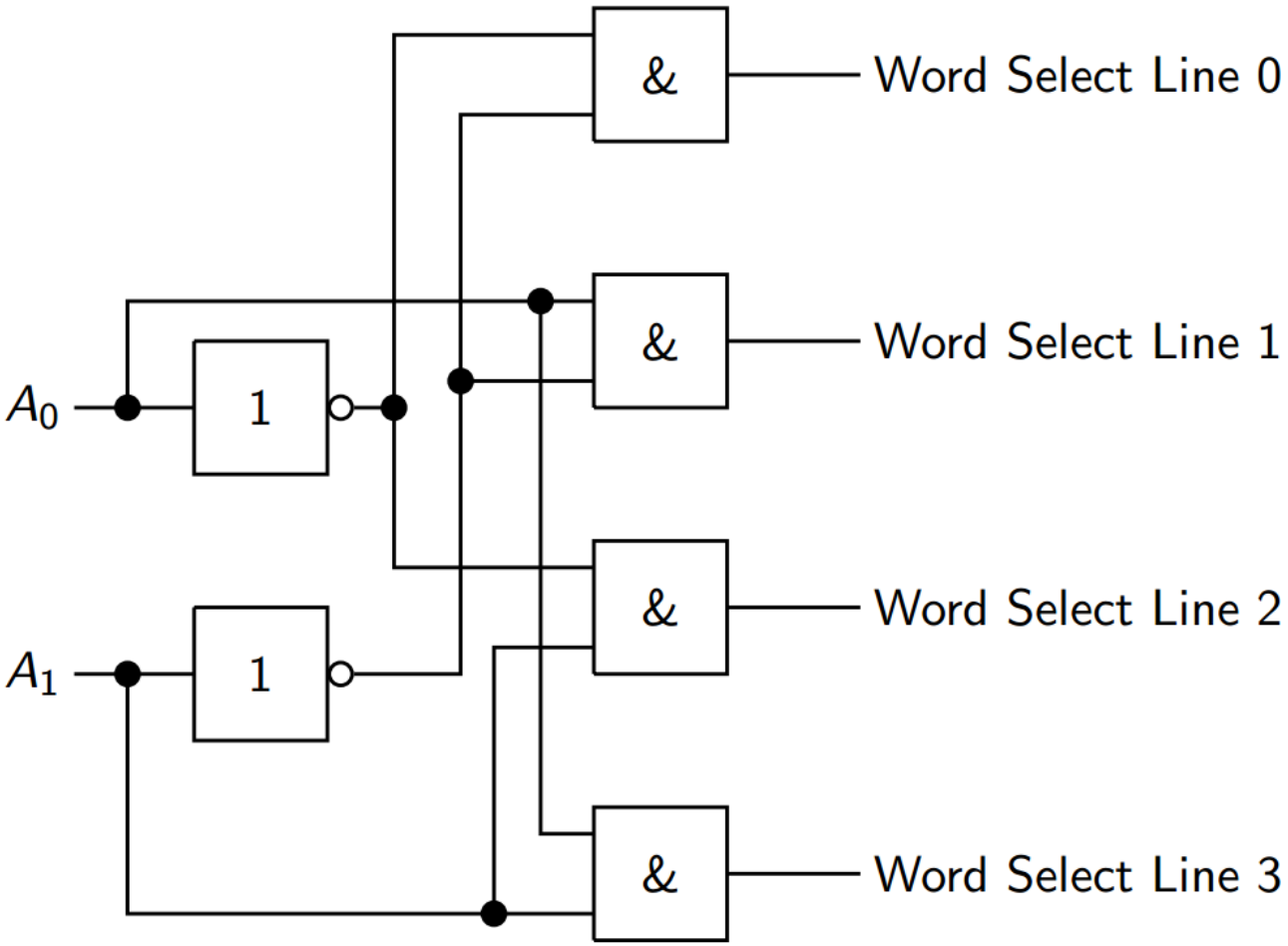
8.6 Speicher & Schieberegister

Speicher für zwei 3-Bit Wörter



8.6 Speicher & Schieberegister

Word Select Lines für vier Wörter



8.6 Speicher & Schieberegister

Speicher für viele Wörter

allgemein 2^k Wörter speichern

Offensichtlich k Adressleitungen benötigt

zunächst formale Beschreibung als boolesche Funktion

$$f: \{0,1\}^k \rightarrow \{0,1\}^{2^k}$$

mit $f\ (A_0, A_1, \ldots, A_{k-1}) = \left(s_0, s_1, \ldots, s_{2^k-1}\right)$

$$\text{mit } s_i = \begin{cases} 1 & \text{falls} \quad (A_{k-1}, A_{k-2}, \ldots, A_0)_2 = i \\ 0 & \text{sonst} \end{cases}$$

Name der Funktion Decoder

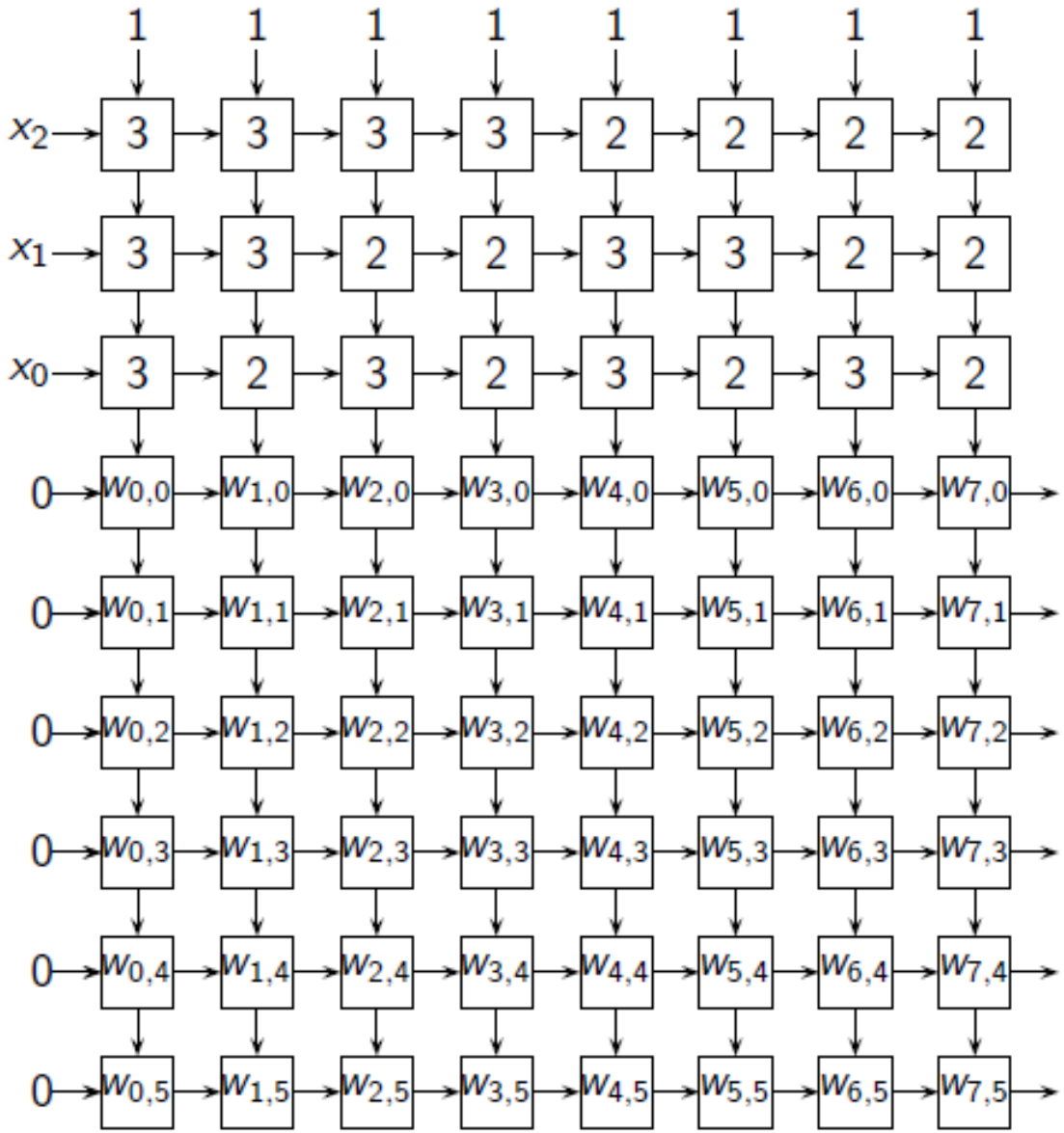
genauer $k \times 2^k$-Decoder

8.6 Speicher & Schieberegister

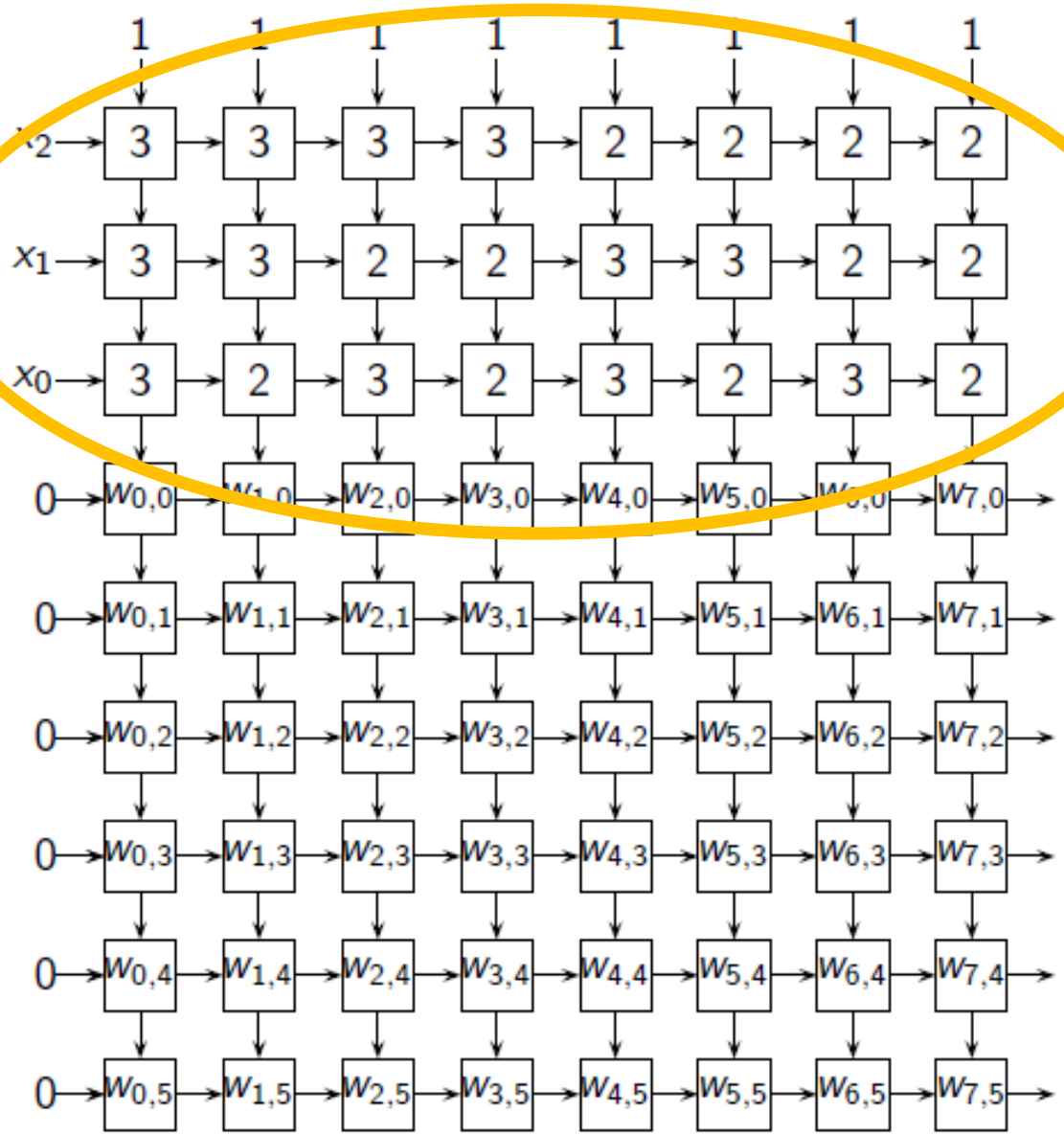
Decoder

Nachdenken Kommt uns das nicht bekannt vor?

8.6 Speicher & Schieberegister



8.6 Speicher & Schieberegister



Und-Teil realisiert einen Decoder

Demultiplexer

Erinnerung

- $MUX_d: \{0,1\}^{d+2^d} \rightarrow \{0,1\}$
- $MUX_d(y_1, y_2, \ldots, y_d, x_0, x_1, \ldots, x_{2^d-1}) = x_{(y_1, y_2, \ldots, y_d)_2}$

Erinnerung

- $DECODE_k: \{0,1\}^k \rightarrow \{0,1\}^{2^k}$ mit
- $DECODE_k(A_0, A_1, \ldots, A_{k-1}) = (s_0, s_1, \ldots, s_{2^k-1})$ mit
- $s_i = \begin{cases} 1 & \text{falls} \quad (A_{k-1}, A_{k-2}, \ldots, A_0)_2 = i \\ 0 & \text{sonst} \end{cases}$

Demultiplexer

- $DEMUX_d: \{0,1\}^{d+1} \rightarrow \{0,1\}^{2^d}$ mit
- $DEMUX_d(x, A_0, A_1, \ldots, A_{d-1}) =$

$$(x \wedge (DECODE_d(A_0, A_1, \ldots, A_{d-1}))_0,$$
$$x \wedge (DECODE_d(A_0, A_1, \ldots, A_{d-1}))_1, \ldots,$$
$$x \wedge (DECODE_d(A_0, A_1, \ldots, A_{d-1}))_{2^d-1})$$

8.6 Speicher & Schieberegister

Speicher

Realisiert man Speicher wirklich so?

- nicht für Hauptspeicher von Rechnern
- kompaktere Lösung mittels DRAM (**d**ynamic **r**andom **a**ccess **m**emory) möglich
- tatsächlich typische SRAM-Realisierung (SRAM = static RAM)

Eigenschaften

- dauerhaft
- schnell
- Zugriffszeit von Daten unabhängig
- teuer
- hoher Stromverbrauch

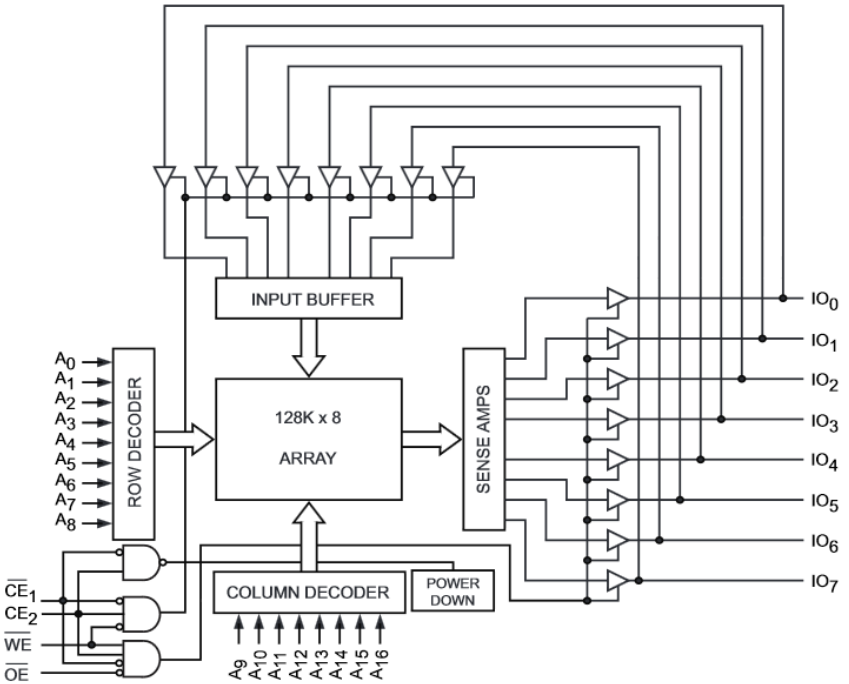
Bemerkung typisch für Cache- oder Register-Speicher

8.6 Speicher & Schieberegister

1-Mbit (128 K × 8) Static RAM

Features

- Pin- and function-compatible with CY7C109B/CY7C1009B
- High speed
  - t_{AA} = 10 ns
- Low active power
  - I_{CC} = 80 mA at 10 ns
- Low CMOS standby power
  - I_{SB2} = 3 mA
- 2.0 V Data Retention
- Automatic power-down when deselected
- TTL-compatible inputs and outputs
- Easy memory expansion with $\overline{CE}_1$, CE_2 and $\overline{OE}$ options
- CY7C109D available in Pb-free 32-pin 400-Mil wide Molded SOJ and 32-pin TSOP I packages. CY7C1009D available in Pb-free 32-pin 300-Mil wide Molded SOJ package



Rechnungen

- Anzahl der Flip-Flops: 128 · 1024 · 8 = 1.048.576
- 1 MByte → 8 ICs — 1 GByte → 8.096 ICs
- 8 ICs → 27,12 € — 8.096 ICs → 13.358,40 € (Mengenrabatt!)
- 8 ICs → 640 mA — 8.096 ICs → 647,68 A (Stromaufnahme)

8.6 Speicher & Schieberegister

Spezielle Speicher: Register

Beobachtung

- in Computern meist **keine direkte** Speichermanipulation
- stattdessen spezielle, direkt der CPU zugehörige Speicherzellen
- **Register**

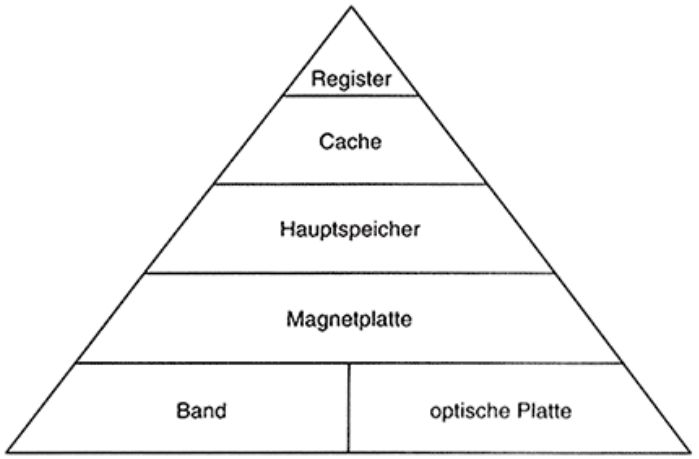
Register

- Register sind auch nur Flip-Flops
- aber spezielle Funktionalität gesonderte Betrachtung
- wir betrachten Schieberegister

8.6 Speicher & Schieberegister

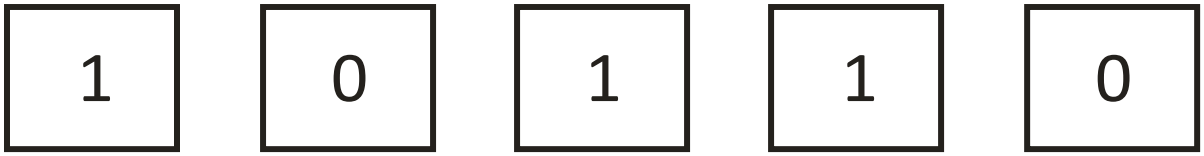
Warum sind Register wichtig? Latenzzeiten für Speicherzugriffe:

- **Register**: 0 Takte
- **primärer Cache**: 2 – 3 Takte
- **sekundärer Cache**: 8 – 10 Takte
- **Arbeitsspeicher (Seite in TLB)**: 75 – 200 Takte
- **Arbeitsspeicher (Seite nicht in TLB, aber im RAM)**: > 2000 Takte
- **Massenspeicher (Seite ausgelagert)**: einige 100 Millionen Takte

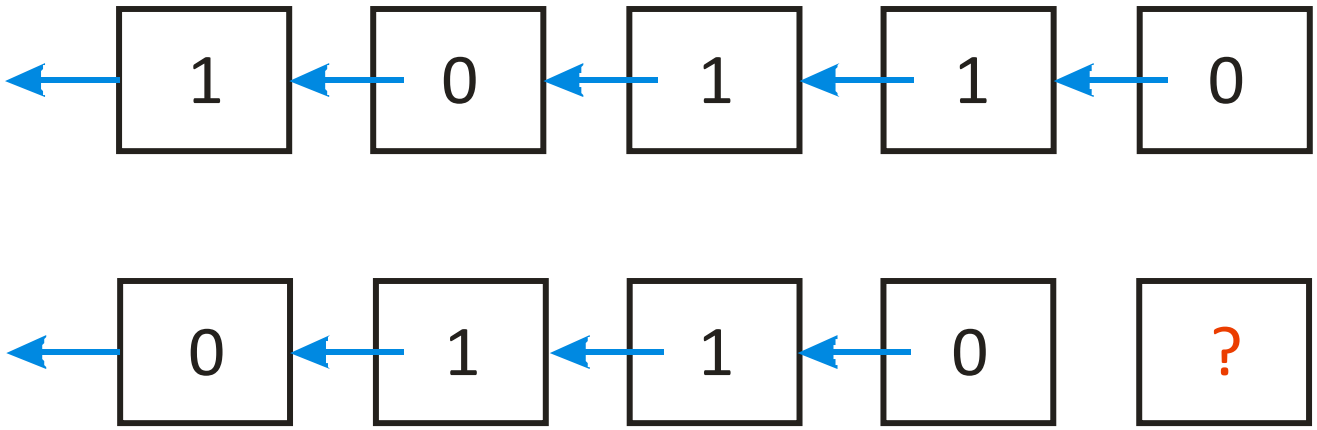


8.6 Speicher & Schieberegister

Schieberegister



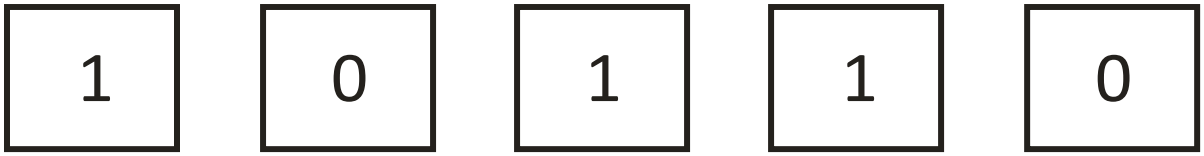
Funktion Speicherinhalte nach links oder rechts verschieben



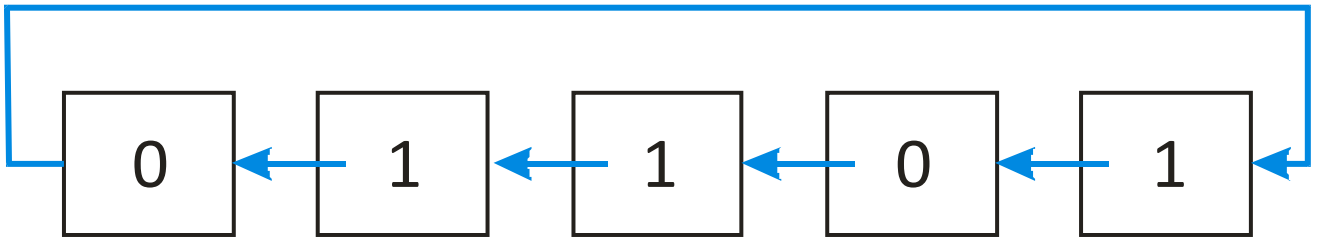
Frage Was machen wir an den Rändern?

8.6 Speicher & Schieberegister

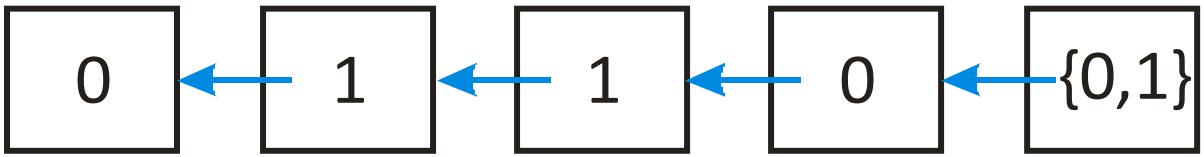
Schieberegister



Möglichkeit 1: zyklisch verschieben



Möglichkeit 2: Wert verbrauchen und auffüllen



8.6 Speicher & Schieberegister

Mealy-Automat zum nicht-zyklischen Schieberegister.

Warum gerade drei Bits?

- klein genug, um auf die Folie zu passen
- groß genug, um alles Wichtige zu enthalten
  - Bit am linken Rand
  - Bit am rechten Rand
  - mittleres Bit

Eingänge

- *d* (direction) Schieberichtung (d = 0 $\stackrel{\text{def}}{=}$ links, d = 1 $\stackrel{\text{def}}{=}$ rechts)
- *x* aufzufüllender Wert

Vereinfachung

- betrachten nicht Ein-/Ausgänge zum
- Schreiben/Lesen des Registers als Ganzem

8.6 Speicher & Schieberegister

Mealy-Automat zum nicht-zyklischen Schieberegister

Eingabealphabet $\Sigma = \{00,01,10,11\}$

Interpretation: $dx \in \Sigma$

Ausgabealphabet $\Delta = \emptyset$

d. h.: $\forall q \in Q, w \in \Sigma : \ \lambda(q,w) = \varepsilon$

Zustände $Q = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Interpretation: Registerinhalte

Startzustand $q_0 = 000$

8.6 Speicher & Schieberegister

Mealy-Automat zum nicht-zyklischen Schieberegister

Zustandsübergangsfunktion

$\delta: Q \times \Sigma \rightarrow Q$

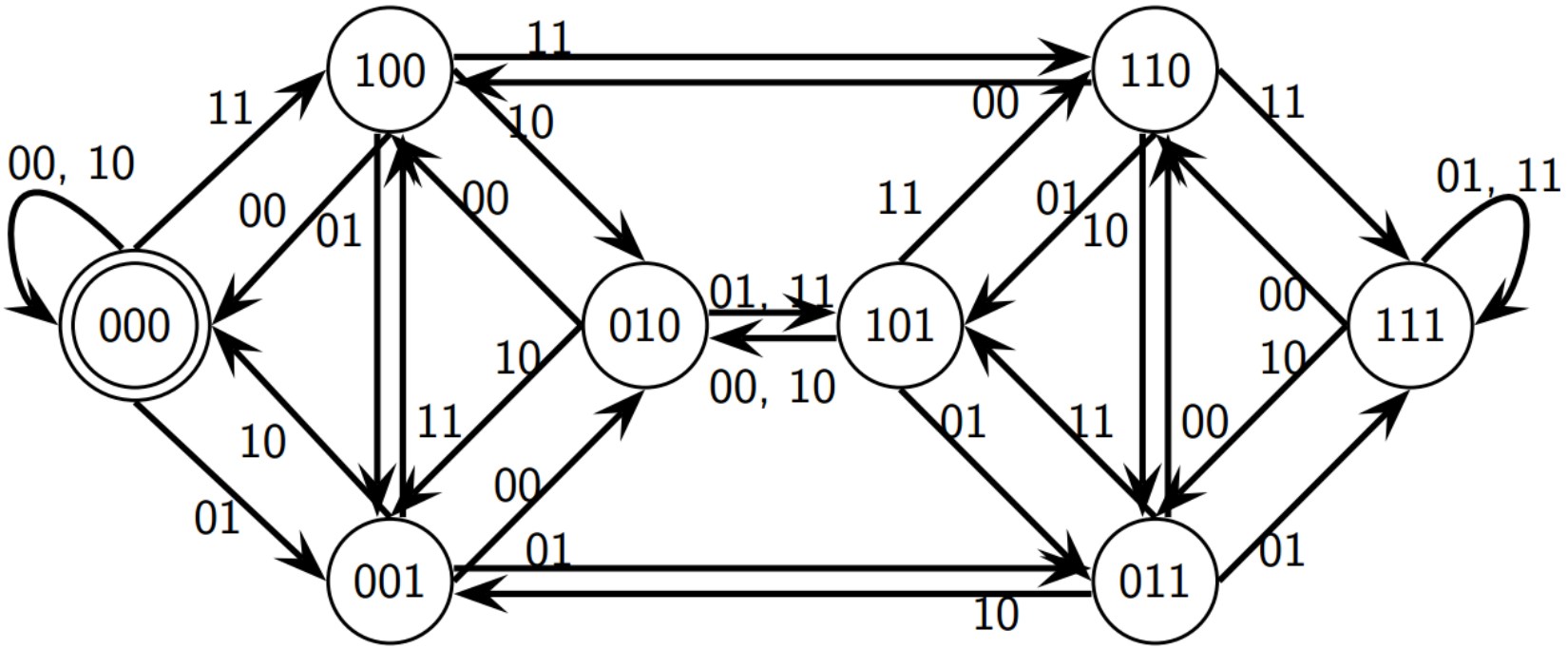
$q \in Q$	$w \in \Sigma$	$\delta(q, w) \in Q$
000	00	000
000	01	001
000	10	000
000	11	100
001	01	011
...	...	...
111	11	111

8.6 Speicher & Schieberegister

Mealy-Automat zum nicht-zyklischen Schieberegister

$\Sigma = \{00,01,10,11\}$, $dx \in \Sigma$, (d = 0 $\stackrel{\text{def}}{=}$ links, d = 1 $\stackrel{\text{def}}{=}$ rechts)

$\Delta = \emptyset$, $\lambda(q,w) = \varepsilon$, $Q = \{0,1\}^3$, $q_0 = 000$



8.6 Speicher & Schieberegister

Schaltwerk-Synthese abgekürzt

Codierung

- $w = dx$ durch d und x
- $q = q_l q_m q_r \in \{0,1\}^3$ durch q_l, q_m und q_r

Müssen wir jetzt Tabellen aufstellen? Nein, wir dürfen auch nachdenken.

Strukturverständnis zum **linken Bit**

- 1. Fall: d = 0 $\stackrel{\text{def}}{=}$ links

  $q_{l(neu)} = q_m$
- 2. Fall: d = 1 $\stackrel{\text{def}}{=}$ rechts

  $q_{l(neu)} = x$

→ $q_{l(neu)} = \overline{d} q_m \vee dx$

8.6 Speicher & Schieberegister

Schaltwerk-Synthese abgekürzt

Strukturverständnis zum **rechten Bit**

- 1. Fall: d = 0 $\stackrel{\text{def}}{=}$ links

  $q_{r(neu)} = x$
- 2. Fall: d = 1 $\stackrel{\text{def}}{=}$ rechts

  $q_{r(neu)} = q_m$

→ $q_{r(neu)} = \overline{d}x \vee dq_m$

Strukturverständnis zum **mittleren Bit**

- 1. Fall: d = 0 $\stackrel{\text{def}}{=}$ links

  $q_{m(neu)} = q_r$
- 2. Fall: d = 1 $\stackrel{\text{def}}{=}$ rechts

  $q_{m(neu)} = q_l$

→ $q_{m(neu)} = \overline{d}q_r \vee dq_l$

8.6 Speicher & Schieberegister

Schaltwerk-Synthese abgekürzt

Moment! Hätten wir nicht Flip-Flops wählen und Ansteuerfunktionen berechnen müssen?

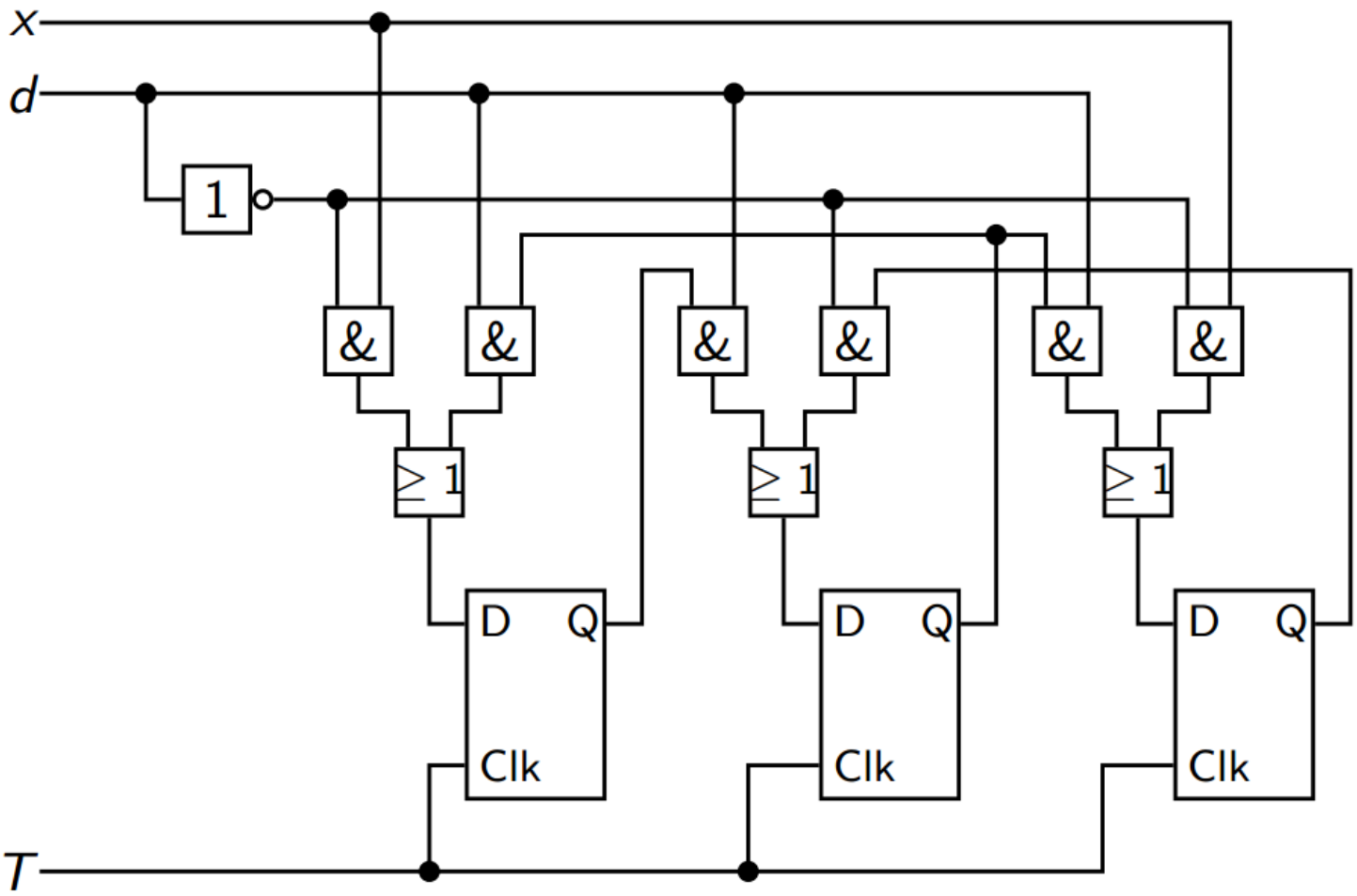
Einsicht nicht, wenn man D-Flip-Flops verwendet, da wir nur speichern müssen

Ansteuerfunktionen

- $q_{l(neu)} = \overline{d} q_m \vee dx$
- $q_{m(neu)} = \overline{d} q_r \vee dq_l$
- $q_{r(neu)} = \overline{d} x \vee dq_m$

8.6 Speicher & Schieberegister

Schaltwerk nicht-zyklisches Schieberegister



8.6 Speicher & Schieberegister

Zyklisches Schieberegister

Wir betrachten **zyklisches Schieberegister**

- wieder für drei Bits
- wieder ohne Daten-Eingabe

Mealy-Automat

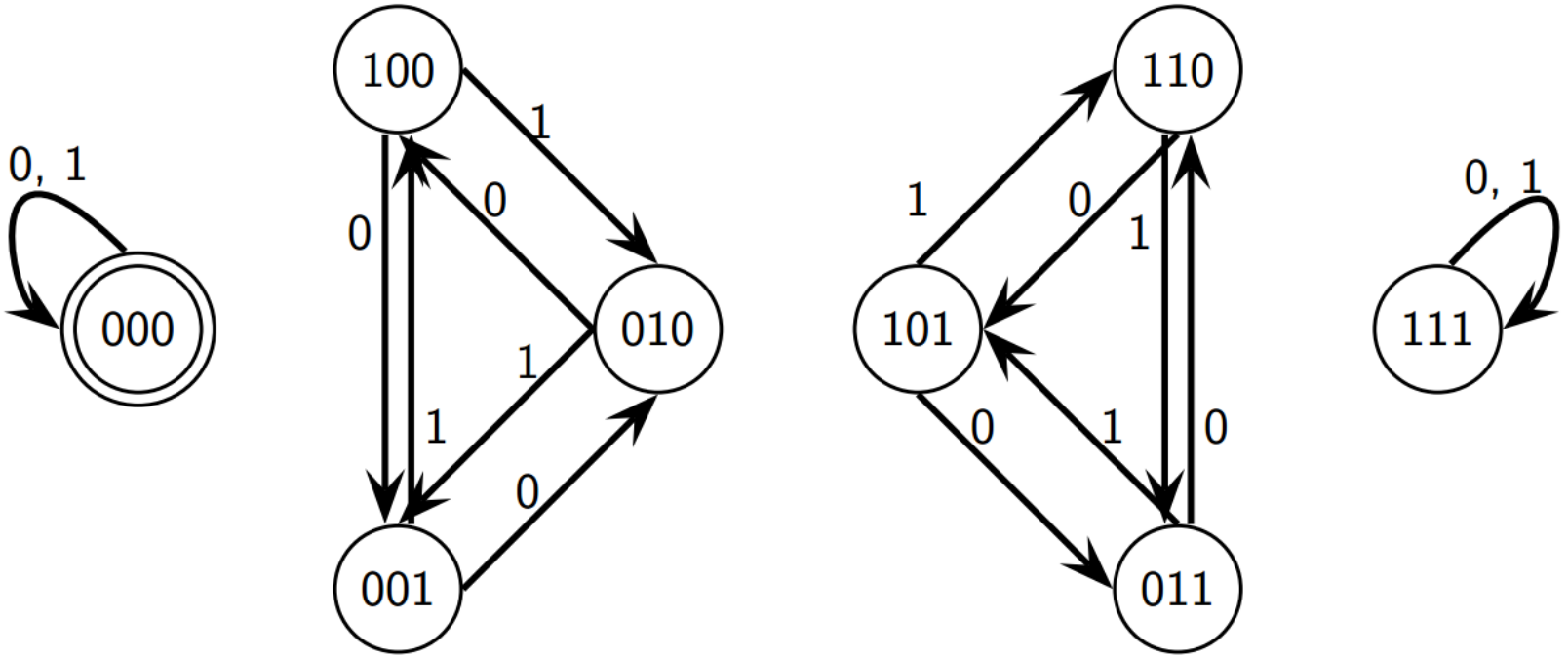
- **wie gehabt** $Q = \{0,1\}^3$, $\Delta = \emptyset$, $\forall$ q $\in$ Q, w $\in$ Σ: $\lambda(q, w) = \varepsilon$
- **anders** $\Sigma = \{0,1\}$ (weil *x* fehlt)
- **wie gehabt** Interpretation d = 0 $\stackrel{\text{def}}{=}$ links, d = 1 $\stackrel{\text{def}}{=}$ rechts

8.6 Speicher & Schieberegister

Mealy-Automat zyklisches Schieberegister

$\Sigma = \{0,1\}$, $w = d$, $d = 0 \stackrel{\text{def}}{=}$ links, $d = 1 \stackrel{\text{def}}{=}$ rechts

$\Delta = \emptyset$, $Q = \{0,1\}^3$, $q_0 = 000$, $\lambda(q, w) = \varepsilon$



8.6 Speicher & Schieberegister

Schaltwerk-Synthese abgekürzt

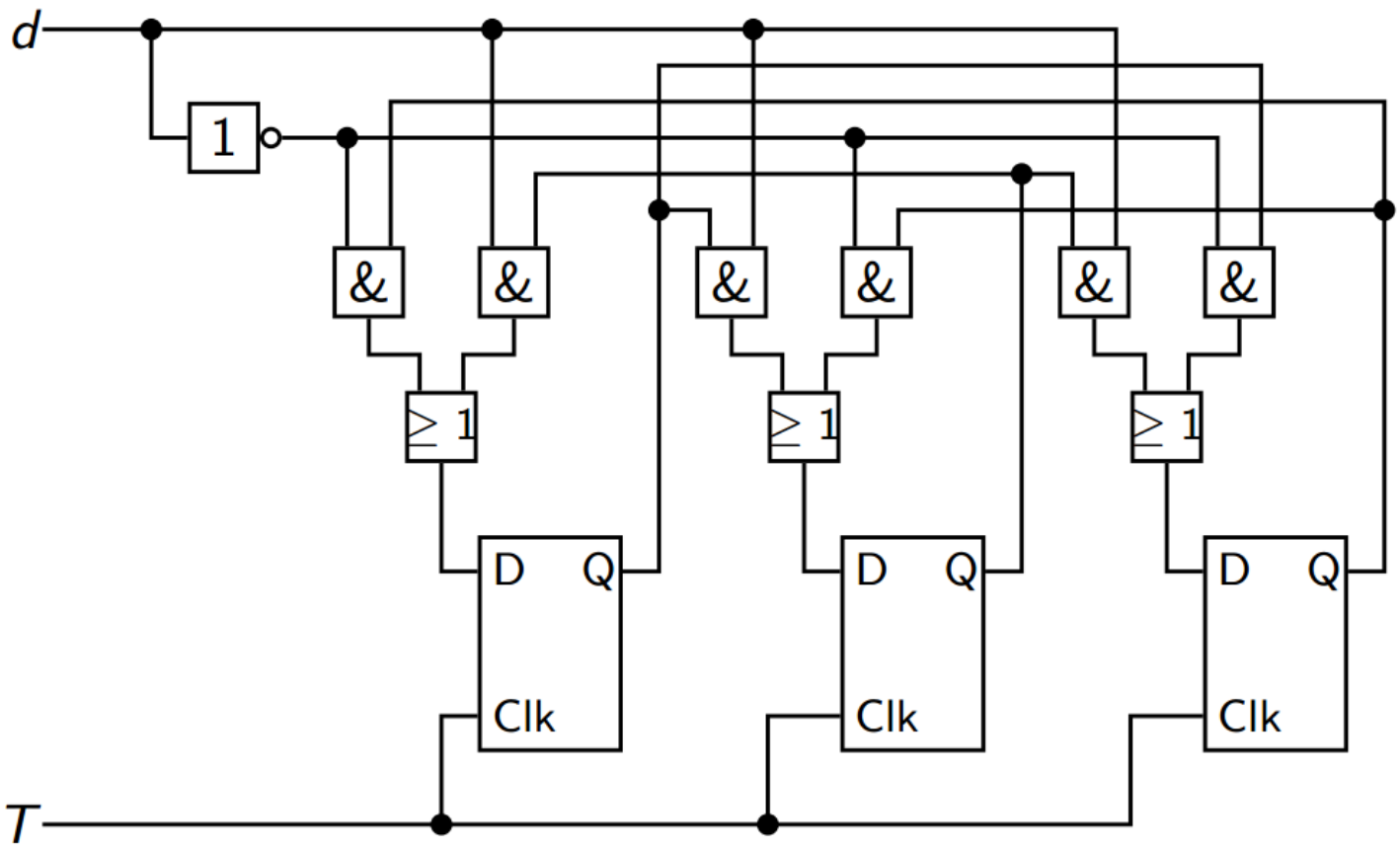
Ansteuerfunktionen direkt ableiten

- $q_{l(neu)} = \overline{d} q_m \vee d q_r$
- $q_{m(neu)} = \overline{d} q_r \vee d q_l$
- $q_{r(neu)} = \overline{d} q_l \vee d q_m$

Schaltwerk kann bei **Verwendung von D-Flip-Flops** direkt angegeben werden

8.6 Speicher & Schieberegister

Schaltwerk zyklisches Schieberegister



8.6 Speicher & Schieberegister

Schieberegister allgemein

Können wir die Funktionen für die Bits auch allgemein angeben?

Notation Bits $q_{n-1}, q_{n-2}, \ldots, q_1, q_0$

q_{n-1} am linken Rand, q_0 am rechten Rand

zyklisches Schieberegister

$q_i = \overline{d} q_{(i-1) \bmod n} \vee d q_{(i+1) \bmod n}$

nicht-zyklisches Schieberegister

$q_i = \overline{d} q_{i-1} \vee d q_{i+1}$ mit $q_{-1} = x$ und $q_n = x$ für den Rand

Welche Operationen können Schieberegister realisieren?

8.6 Speicher & Schieberegister

Welche Operationen können Schieberegister realisieren?
Multiplikation bzw. Division

direkt mit Binärzahlen. . .

1	0	1	0	1	1	0	0	·	1	1	1	0	1	0
														0
						1	0	1	0	1	1	0	0	
												0		
				1	0	1	0	1	1	0	0			
			1	0	1	0	1	1	0	0				
		1	0	1	0	1	1	0	0					
	1	0	0	1	1	0	1	1	1	1	1	0	0	0

Multiplizieren heißt

- Nullen passend schreiben
- Zahlen passend ***verschoben*** kopieren
- viele Zahlen addieren

8.6 Speicher & Schieberegister

Datentransfer

Daten müssen gelegentlich zwischen Registern **transferiert** werden, um z. B. Zwischenergebnisse zu **speichern**

Wie kann man Daten von einem Register in ein anderes transferieren?

- auf Leitungen zwischen den Registern
- wie verbindet man Register geschickt?

Kriterien

- Einfachheit (der technischen Realisierung)
- Sparsamkeit (im Sinne des Schaltungsaufwands)
- Möglichkeit zum effizienten Datenaustausch

8.6 Speicher & Schieberegister

Register verbinden

eine Möglichkeit: **paarweise verbinden**

Vorteile

- direkter Datenaustausch, also schnell
- bis $^{n}/_{2}$ Registerpaare gleichzeitig aktiv, also gute Parallelisierung

Nachteil

- $\binom{n}{2}$ Verbindungen
- Beispiel $\binom{16}{2} = \frac{16 \cdot 15}{2} = 120$ Verbindungen
- meist nicht realisiert

8.6 Speicher & Schieberegister

Register sparsam verbinden

andere Möglichkeit: **alle Register mit einer gemeinsamen Leitung verbinden**

Vorteile

- nur eine Leitung, also einfach
- Datenaustausch direkt, also schnell

Nachteile

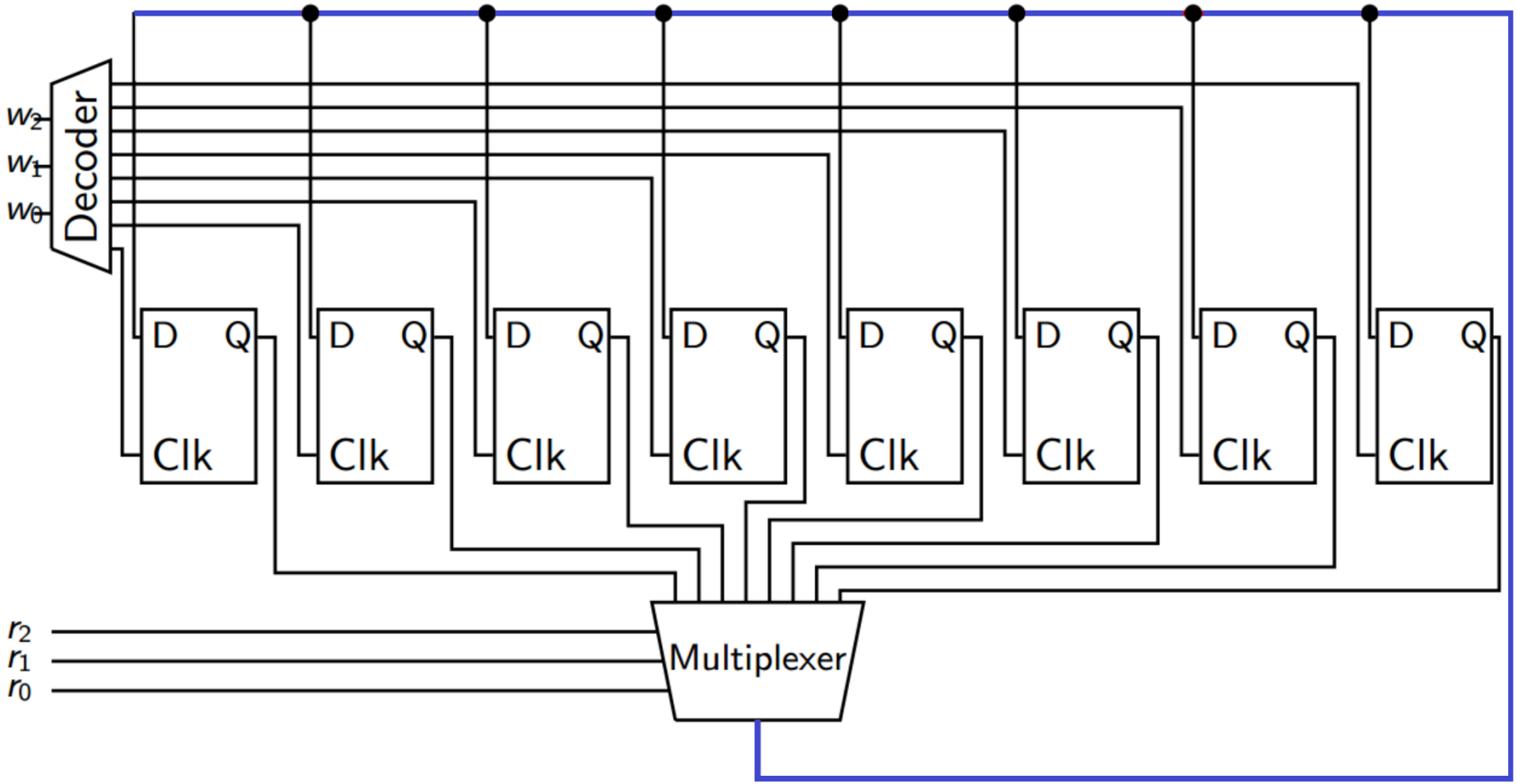
- immer nur ein Paar aktiv, also langsam
- Steuerlogik erforderlich, nicht ganz einfach

Name dieser Variante: Bus

- häufig verwendet
- Engpass durch mehrere Busse entschärfbar

8.6 Speicher & Schieberegister

Bus für acht 1-Bit-Register



8.6 Speicher & Schieberegister

Verallgemeinerungen

Verallgemeinerung auf **mehr Register**

- mehr Adressleitungen zum Adressieren des Quell- und Zielregisters
- größere Multiplexer und Decoder

Verallgemeinerung auf breitere Register

- je Register entsprechend mehr Flip-Flops (je Bit ein Flip-Flop)
- entsprechend mehr Multiplexer (je Bit ein Multiplexer)
- entsprechend mehr Bus-Leitungen (je Bit eine Bus-Leitung)

8. Synchrone Schaltwerke

1. Einleitung ✓
2. Bistabile Kippstufe ✓
3. Automaten ✓
4. Synchrone Schaltwerke ✓
5. Serienaddierwerke ✓
6. Speicher & Schieberegister ✓
7. **Takt**

8.7 Takt

Taktung von Digitalrechnern

Gibt es in einem Rechner eigentlich **den Takt**? JEIN

- oft mehrere Takte
- aber nur eine "Uhr" zur Erzeugung der Takte

Wie funktioniert das?

Einsicht schneller geht nicht, verschoben oder langsamer schon

Langsamer – Wie geht das?

- Takt(e) auslassen
- dazu Zähler ausreichend
- Zähler wie jedes Schaltwerk realisierbar

8.7 Takt

Taktreduktion mit Zählern

Idee: Zähle Taktimpulse → erzeuge in jedem n-ten Takt einen neuen Impuls für den **langsameren** abgeleiteten Takt

Erforderlich: Zähler modulo n, d.h. $z \leftarrow (z+1) \bmod n$

Realisierung als synchrones Schaltwerk

- Zustand $\stackrel{\text{def}}{=}$ Zählerstand, d.h.: $\boldsymbol{Q} = (\mathbf{0}, \mathbf{1}, \ldots, \boldsymbol{n}-\mathbf{1})$
  speicherbar in $\log_2(n)$ Bits / Flip-Flops
- Ausgabe, z.B. wenn Zählerstand 0 erreicht
  → nicht weiter betrachtet
- Eingabe? ... eigentlich nicht zwingend erforderlich, aber Umschaltung der Zählrichtung möglich!
  $\Rightarrow \boldsymbol{\Sigma} = \{\mathbf{0}, \mathbf{1}\}$ (d.h. $1 \stackrel{\text{def}}{=}$ vorwärts, $0 \stackrel{\text{def}}{=}$ rückwärts zählen)

8.7 Takt

Beispiel: Zähler modulo 8

Zustandsüberführungsfunktion: $\delta(q,d) = \left(q + (-1)^{1-d}\right) \bmod 8$

d	q			q_{neu}			R_l	S_l	R_m	S_m	R_r	S_r
0	0	0	0	1	1	1	0	1	0	1	0	1
0	0	0	1	0	0	0	*	0	*	0	1	0
0	0	1	0	0	0	1	*	0	1	0	0	1
0	0	1	1	0	1	0	*	0	0	*	1	0
0	1	0	0	0	1	1	1	0	0	1	0	1
0	1	0	1	1	0	0	0	*	*	0	1	0
0	1	1	0	1	0	1	0	*	1	0	0	1
0	1	1	1	1	1	0	0	*	0	*	1	0
1	0	0	0	0	0	1	*	0	*	0	0	1
1	0	0	1	0	1	0	*	0	0	1	1	0
1	0	1	0	0	1	1	*	0	0	*	0	1
1	0	1	1	1	0	0	0	1	1	0	1	0
1	1	0	0	1	0	1	0	*	*	0	0	1
1	1	0	1	1	1	0	0	*	0	1	1	0
1	1	1	0	1	1	1	0	*	0	*	0	1
1	1	1	1	0	0	0	1	0	1	0	1	0

Ansteuertabelle

Q_{alt}	Q_{neu}	R	S
0	0	*	0
0	1	0	1
1	0	1	0
1	1	0	*

8.7 Takt

Beispiel: Zähler modulo 8

Flip-Flop l

- $R_l = \overline{d}\ q_l \overline{q_m}\ \overline{q_r} \vee d\ q_l\ q_m\ q_r$
- $S_l = \overline{d}\ \overline{q_l}\ \overline{q_m}\ \overline{q_r} \vee d\ \overline{q_l}\ q_m\ q_r$

Flip-Flop m

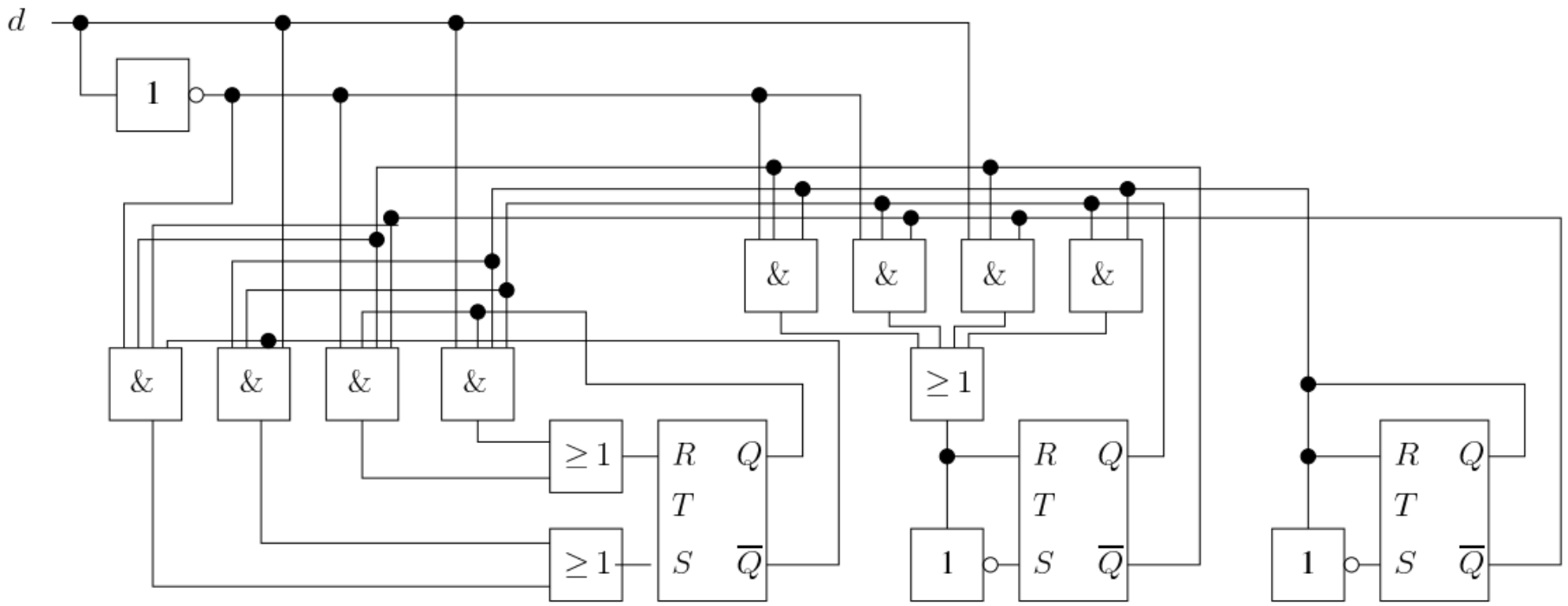
- $R_m = \overline{d}\ \overline{q_m}\ q_r \vee \overline{d}\ q_m\ \overline{q_r} \vee d\ \overline{q_m}\ \overline{q_r} \vee q_m\ q_r$
- $S_m = \overline{R_m}$

Flip-Flop r

- $R_r = q_r$
- $S_r = \overline{R_r}$

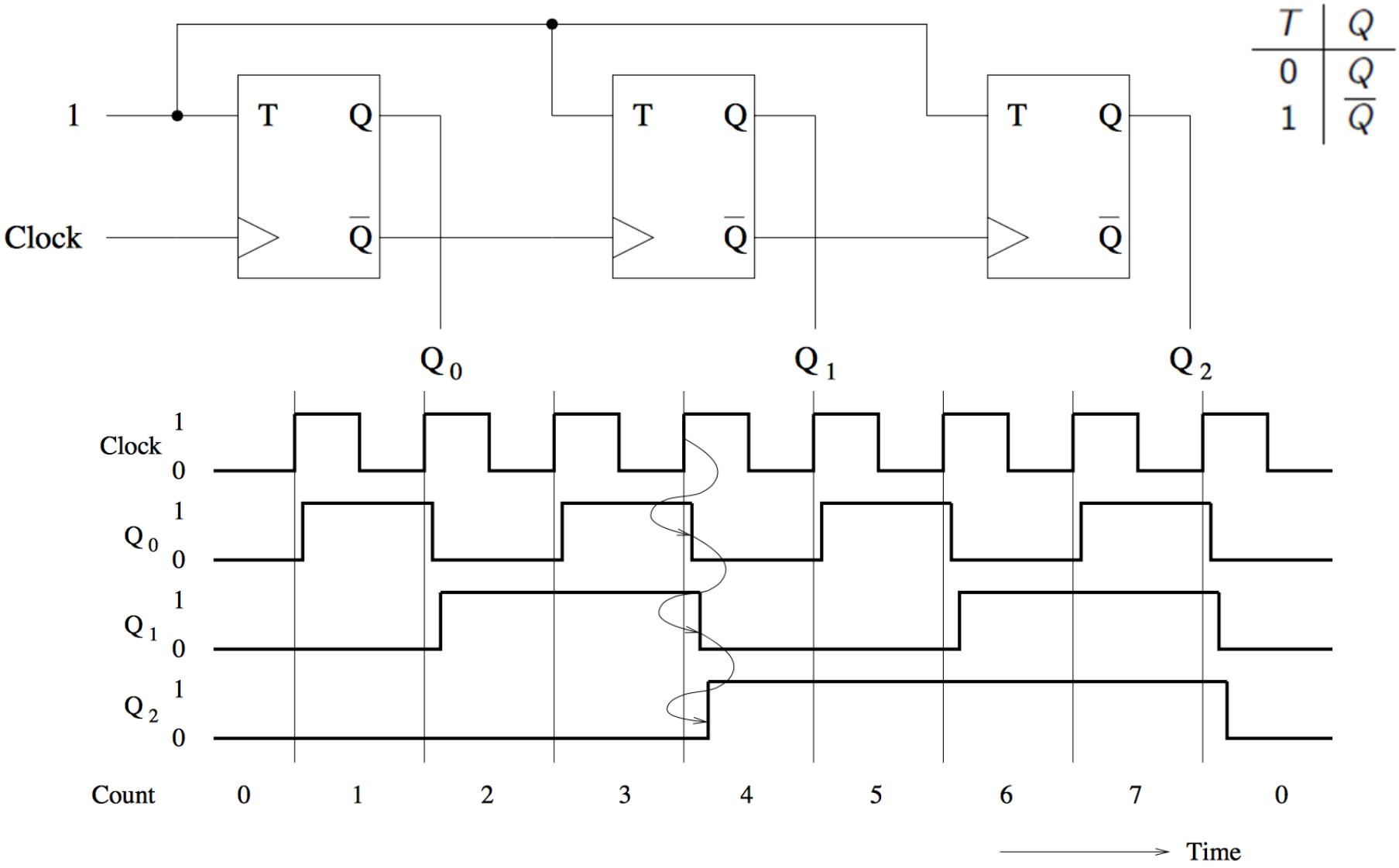
8.7 Takt

Schaltwerk des Zählers modulo 8



8.7 Takt

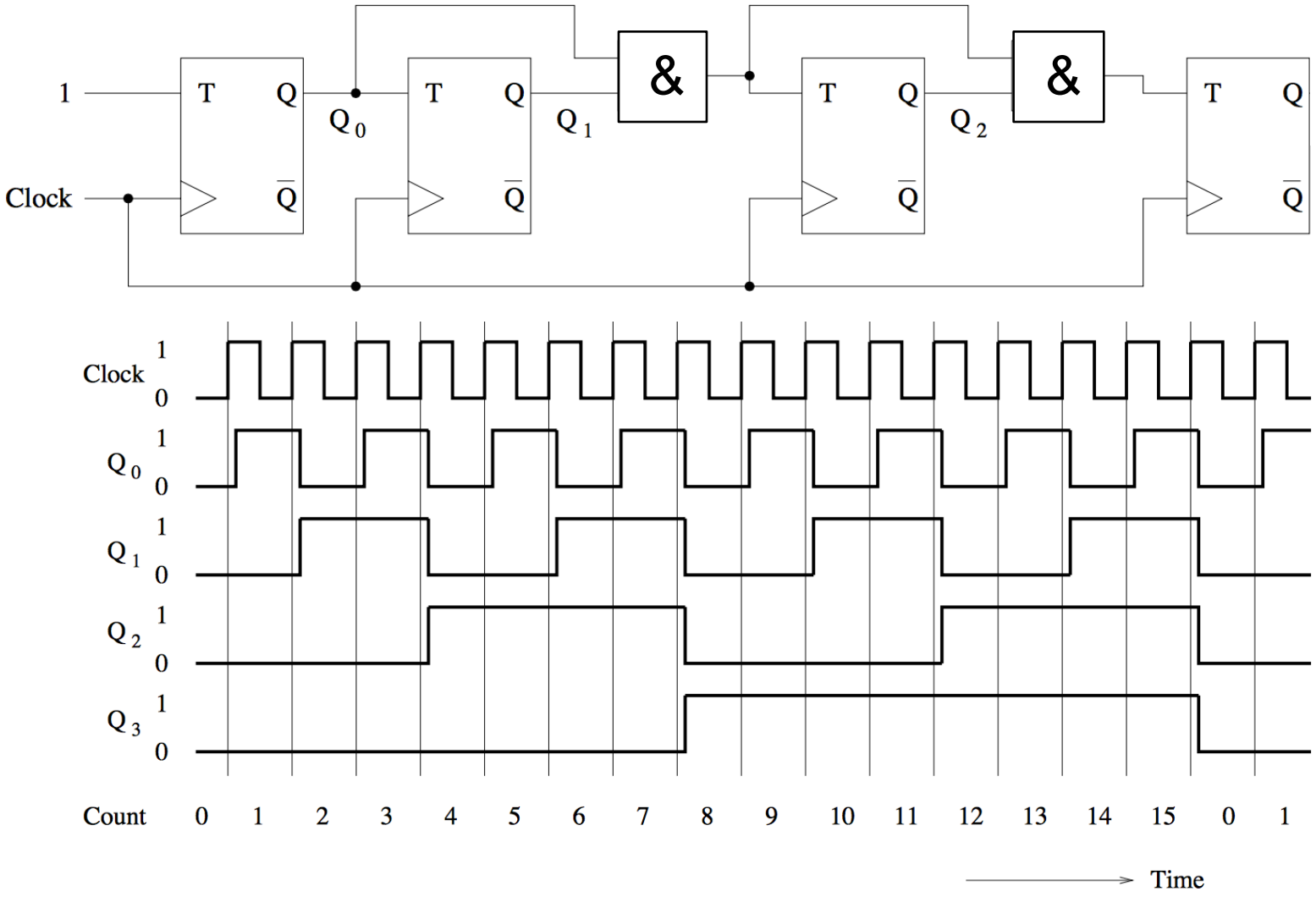
Taktteilung Betrachte Kette von T-Flip-Flops in asynchroner Schaltung



8.7 Takt

Taktteilung Betrachte Kette von T-Flip-Flops in synchroner Schaltung

T	Q
0	Q
1	$\overline{Q}$

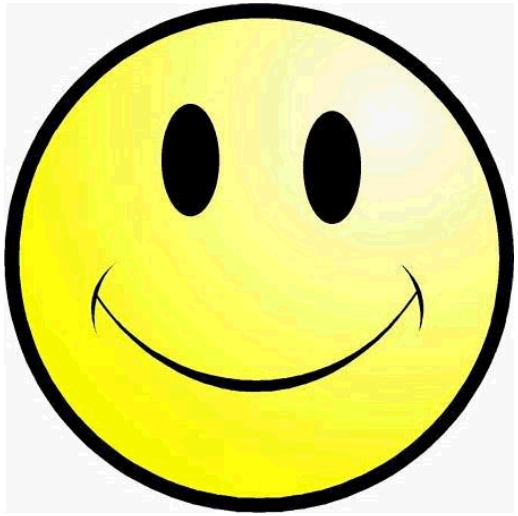


8. Synchrone Schaltwerke

1. Einleitung ✓
2. Bistabile Kippstufe ✓
3. Automaten ✓
4. Synchrone Schaltwerke ✓
5. Serienaddierwerke ✓
6. Speicher & Schieberegister ✓
7. Takt ✓

Übersicht

1. Organisatorisches ✓
2. Einleitung ✓
3. Repräsentation von Daten ✓
4. Boolesche Funktionen und Schaltnetze ✓
5. Rechnerarithmetik ✓
6. Optimierung von Schaltnetzen ✓
7. Programmierbare Bausteine ✓
8. Synchrone Schaltwerke ✓



Appendix

8.5 Serienaddierwerke

Moore-Automat zum von Neumann-Addierwerk

Eingabealphabet

alle möglichen Paare von *n*-Bit-Zahlen, außerdem ε

$\Sigma := \{0,1\}^{2n} \cup \{\varepsilon\}$

Ausgabealphabet

alle möglichen Ergebnisse der Addition, außerdem "–"

$\Delta := \{0,1\}^{n+1} \cup \{-\}$

Zustandsmenge

alle möglichen Zwischenergebnisse einschließlich Übertrag

$Q := \{0,1\}^{2n+1}$

Startzustand

willkürlich $q_0 = \underbrace{000 \ldots 000}_{2n+1 \text{ Nullen}} = 0^{2n+1}$

8.5 Serienaddierwerke

Zustandsüberführungsfunktion δ

$$\Sigma = \{0,1\}^{2n} \cup \{\varepsilon\},\ \Delta = \{0,1\}^{n+1} \cup \{-\},\ \mathrm{Q} = \{0,1\}^{2n+1},\ q_0 = 0^{2n+1}$$

1. Fall Eingabe $w = xy \neq \varepsilon$

$\delta(q, w) = 0w = 0xy$

2. Fall Eingabe $w = \varepsilon$

Notation aktueller Zustand $q = \ddot{u}xy$

$$\delta(q, \varepsilon) = \mathrm{q}' = \ddot{u}'x'y' = \ddot{u}'x'_{n-1}x'_{n-2} \ldots x'_0\, y'_{n-1}y'_{n-2} \ldots y'_0$$

mit

- $y'_0 = 0$
- $y'_i = x_{i-1} \wedge y_{i-1}$ für $i \in \{1,2,\ldots,n-1\}$
- $x'_i = x_i \oplus y_i$ für $i \in \{0,1,\ldots,n-1\}$
- $\ddot{u}' = \ddot{u} \vee (x_{n-1} \wedge y_{n-1})$

8.5 Serienaddierwerke

Ausgabefunktion λ

$\Sigma = \{0,1\}^{2n} \cup \{\varepsilon\}, \Delta = \{0,1\}^{n+1} \cup \{-\}, \mathrm{Q} = \{0,1\}^{2n+1}, q_0 = 0^{2n+1}$

Notation neuer Zustand $\delta(q,w) = \mathrm{q}' = \ddot{\mathrm{u}}'x'y'$

1. Fall $y' \neq 0^n$

$\lambda\big(\delta(q,w)\big) = -$

2. Fall $y' = 0^n$

$\lambda\big(\delta(q,w)\big) = \ddot{\mathrm{u}}'x'$