Logik mit „Gedächtnis“: Sequentielle Logik

- Schaltwerke
- Grundkomponenten zur Informationspeicherung: Flip-Flops
- Typische Schaltwerke
- Entwurf eines Schaltwerks

asynchrone und synchrone Schaltwerke

- asynchrone Schaltwerke
  - gesteuert durch Veränderung der Eingangssignale
  - Zeitpunkt, an dem wieder stabile Ausgangssignale vorliegen, ist nur durch Gatterlaufzeit festgelegt
  - aufwendiger Entwurf (Zeit ist „Echtzeit“)
  - sehr schnelle Schaltwerke möglich
- synchrone Schaltwerke
  - gesteuert durch zentralen Takt
  - Übernahme der Änderung eines Eingangssignals erfolgt nur zu festen Zeitpunkten
  - einfacher, systematischer Entwurf (Zeit ist „Taktzeit“)
  - langsamste Komponente bestimmt maximale Taktfrequenz

Sequentielle Logik

Kombinatorische Logik wird durch Schaltnetze repräsentiert, die durch zyklenfreie Graphen dargestellt werden können ⇒ es gibt keine Rückkopplungen !

Eigenschaften:

1. Zustandslos: Ausgabe ist nur von der Eingabe abhängig;
2. One-Way: keine Rückkopplungen im Schaltnetz;
3. 0-Verzögerung: keine Berücksichtigung der Gatterlaufzeit.

Frage:

Können wir die bisherigen Konzepte und Techniken der logischen Schaltungen einsetzen, um wesentliche Elemente eines Rechners zu beschreiben wie z.B. :

- Ablaufsteuerungen
- Speicherelemente
- Takterzeuger

?

Welche Eigenschaften müssen hinzugefügt werden?

Logik Simulator "Digital Works"

http://www.electronics-lab.com/downloads/schematic/002/index.html

oder im Lehrbuch:

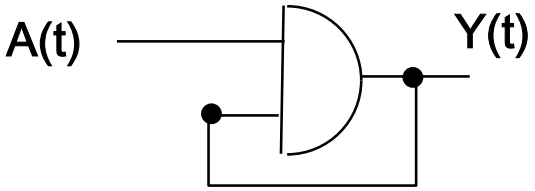
A. Clements: The Principles of Computer Hardware,
3. Auflage, Oxford University Press, 2000

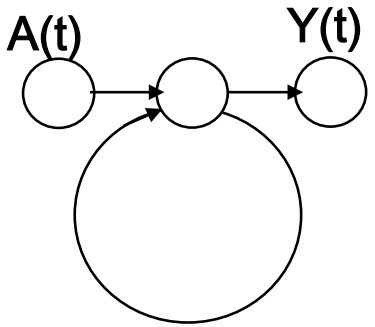
Schaltwerke

Was geschieht in einer digitalen Schaltung bei der Rückkopplung eines Gatterausganges ?

Beispiel 1:

rückgekoppeltes UND-Gatter





?

A(t)	Y(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	1	1	1	1
0	1	0	0	0
1	0	0	0	0
0	0	0	0	0

A(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	Y(t)	Y(t)	Y(t)
0	$\overline{Y}$(t)	$\overline{Y}$(t)	$\overline{Y}$(t)

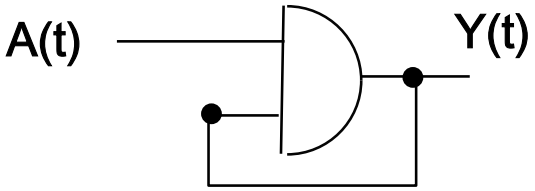
Berücksichtigung von ZEIT !!

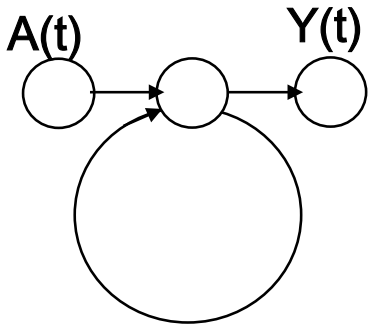
Schaltwerke

Was geschieht in einer digitalen Schaltung bei der Rückkopplung eines Gatterausganges ?

Beispiel 1:

rückgekoppeltes UND-Gatter





?

A(t)	Y(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	1	1	1	1
0	1	0	0	0
1	0	0	0	0
0	0	0	0	0

A(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	Y(t)	Y(t)	Y(t)
0	0	0	0

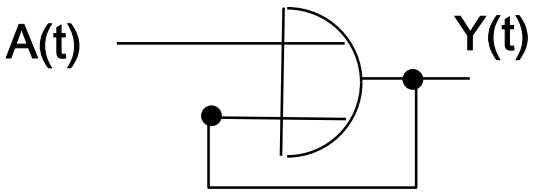
Berücksichtigung von ZEIT !!

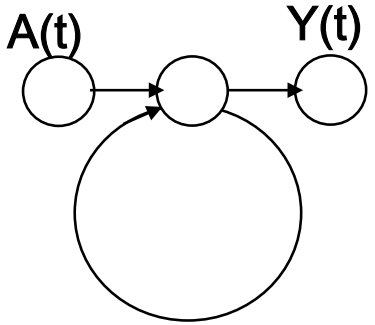
Schaltwerke

- Was geschieht in einer digitalen Schaltung bei der Rückkopplung eines Gatterausganges ?

- Beispiel 2:

rückgekoppeltes ODER-Gatter





?

A(t)	Y(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	1	1	1	1
0	1	1	1	1
1	0	1	1	1
0	0	0	0	0

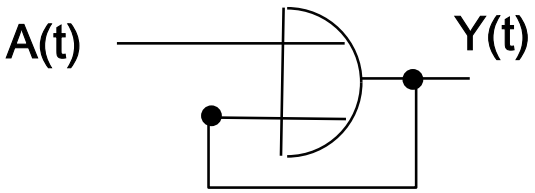
A(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	Y(t)	Y(t)	Y(t)
0	$\overline{Y}$(t)	$\overline{Y}$(t)	$\overline{Y}$(t)

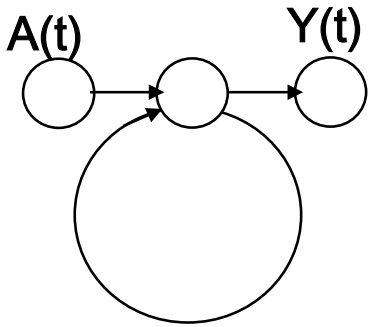
Schaltwerke

- Was geschieht in einer digitalen Schaltung bei der Rückkopplung eines Gatterausganges ?

- Beispiel 2:

rückgekoppeltes ODER-Gatter





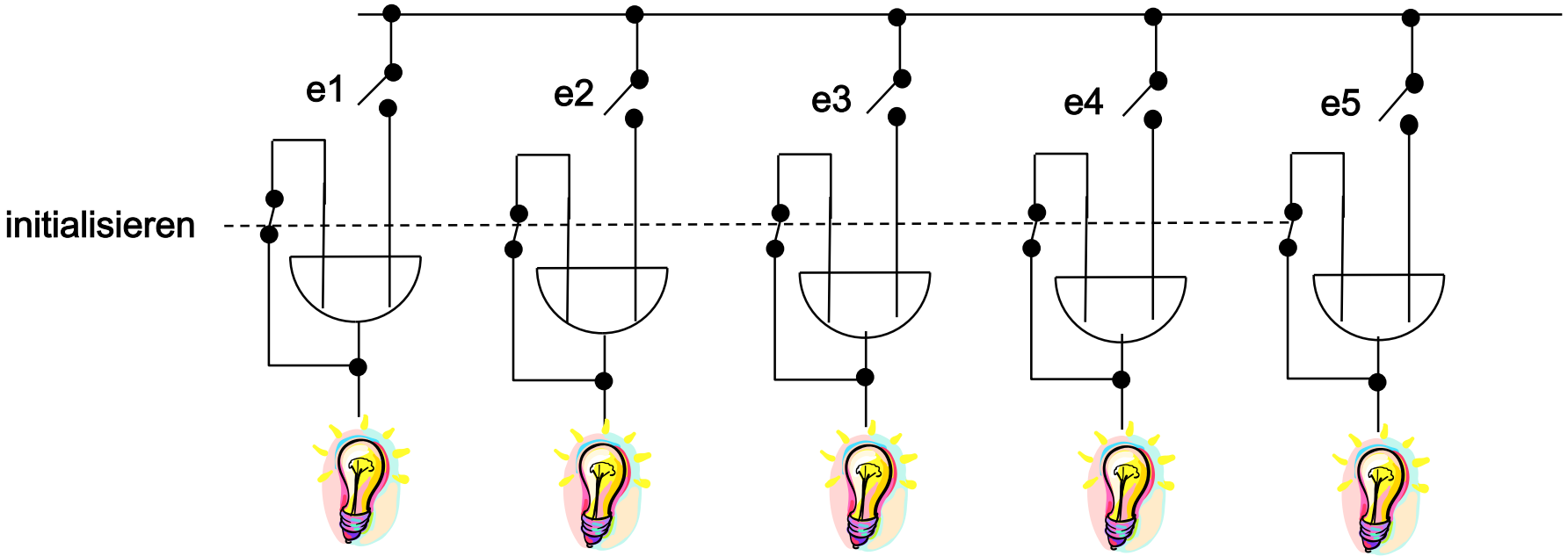
?

A(t)	Y(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	1	1	1	1
0	1	1	1	1
1	0	1	1	1
0	0	0	0	0

A(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	1	1	1
0	$\overline{Y}(t)$	$\overline{Y}(t)$	$\overline{Y}(t)$

Schaltwerke

Ereignisspeicher

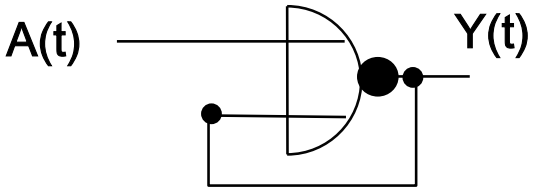


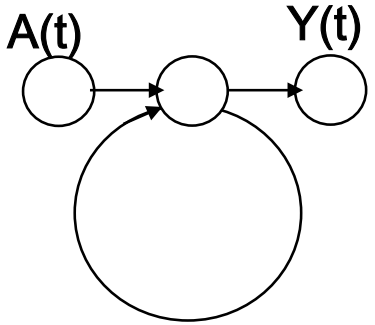
Schaltwerke

- Was geschieht in einer digitalen Schaltung bei der Rückkopplung eines Gatterausganges ?

- Beispiel 3:

rückgekoppeltes NOR-Gatter





?

die Schaltung „schwingt“ wenn A= 0 !

A(t)	Y(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	1	0	1	0
0	1	0	1	0
1	0	0	0	0
0	0	1	0	1

A(t)	Y(t+Δt)	Y(t+2Δt)	Y(t+3Δt)
1	0	0	0
0	$\overline{Y}$(t)	Y(t)	$\overline{Y}$(t)

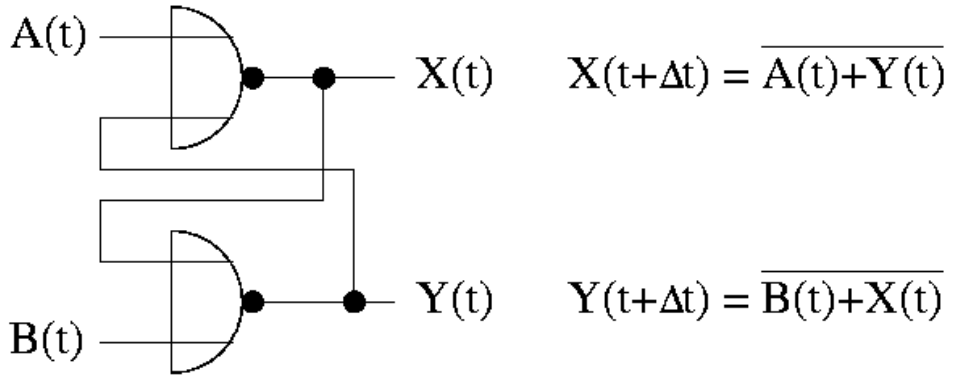
Berücksichtigung der Gatterverzögerung notwendig !

Schaltwerke: Flip-Flops

zwei rückgekoppelte NOR-Gatter

Flip-Flop
(auch:
bistabile Kippstufe oder
bistabiler Multivibrator)

Δt: Gatterverzögerung



A(t)	B(t)	X(t+Δt)	Y(t+Δt)	X(t+2Δt)	Y(t+2Δt)
0	0	$\overline{Y(t)}$	$\overline{X(t)}$	?	?
0	1	$\overline{Y(t)}$	0	1	0
1	0	0	$\overline{X(t)}$	0	1
1	1	0	0	?	?

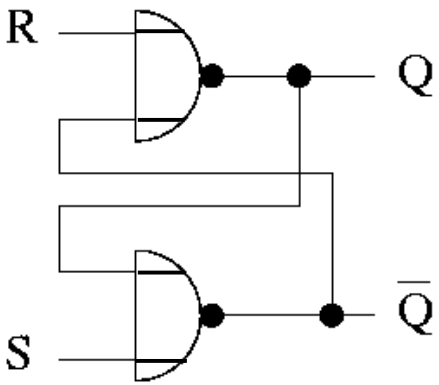
⇒ bei A = 0 oder B = 0 ergibt sich nach 2Δt ein **stabiler** Zustand;
bei A = 1 und B = 1 ergibt sich für X = Y ein **instabiles** Verhalten.

RS Flip-Flop

- bei Vermeidung von A(t) = B(t) = 1 liegen stabile Zustände mit Y = X vor; bistabile Kippstufe kann einen binären Wert speichern!
- Setzt man
  - R = A („*Reset*“, Löschen) und S = B („*Set*“, Setzen)
  - Q = X und Q = Y

  so ergibt sich ein RS Flip-Flop:

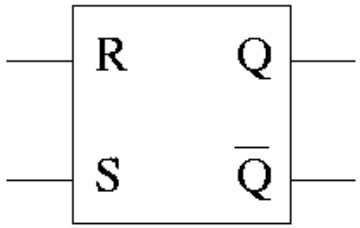
Realisierung mit NOR:



Wahrheitstabelle:

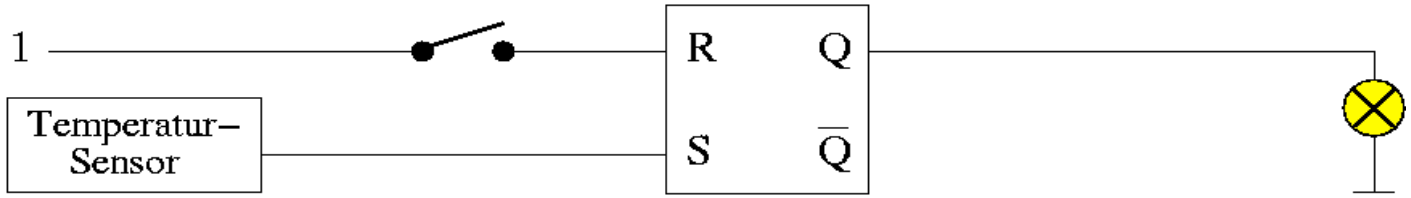
R(t)	S(t)	Q'(t) = Q(t+2Δt)
0	0	Q(t)
0	1	1
1	0	0
1	1	nicht erlaubt

Symbol:

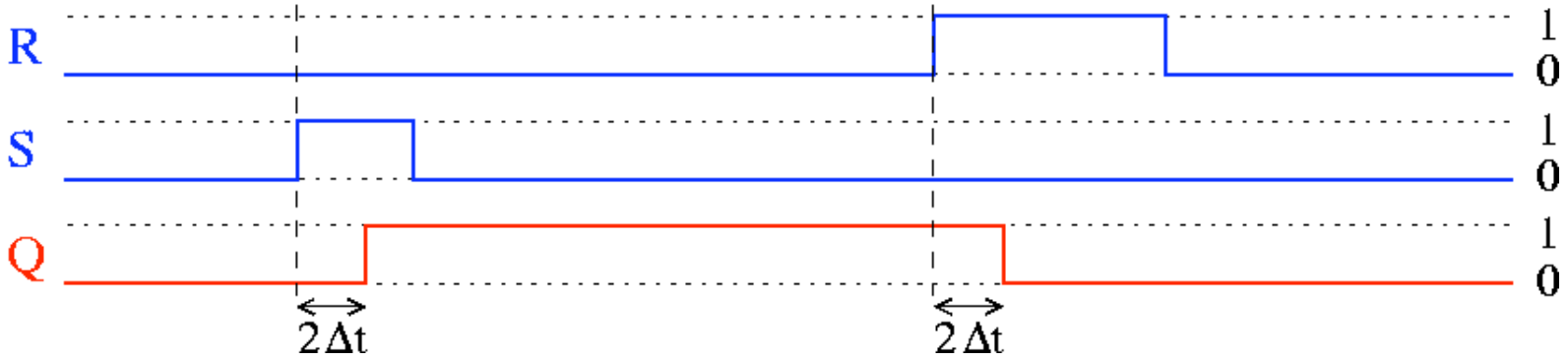


RS Flip-Flop

- Einsatz eines RS-Flipflops: Speichern eines kurzzeitigen Wertes
- Beispiel: Setzen einer Warnlampe bei kurzzeitiger Temperatur-überschreitung, manuelles Rücksetzen

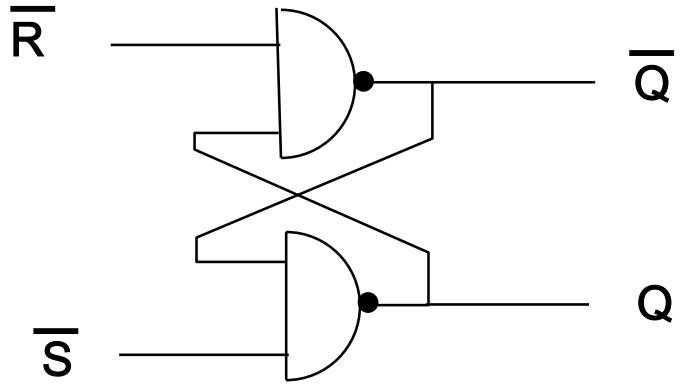


- Zeitverhalten eines RS Flip-Flops:



RS Flip-Flop

RS Flip-Flop mit NAND

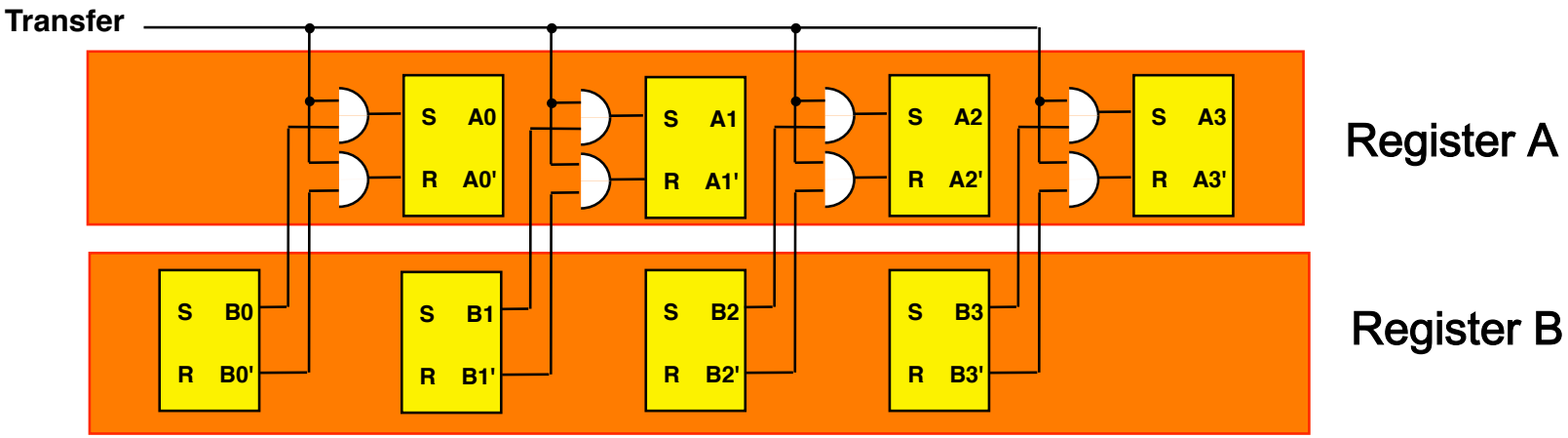


Wahrheitstabelle:

R(t)	S(t)	Q'(t) = Q(t+2Δt)
0	0	nicht erlaubt
0	1	1
1	0	0
1	1	Q(t)

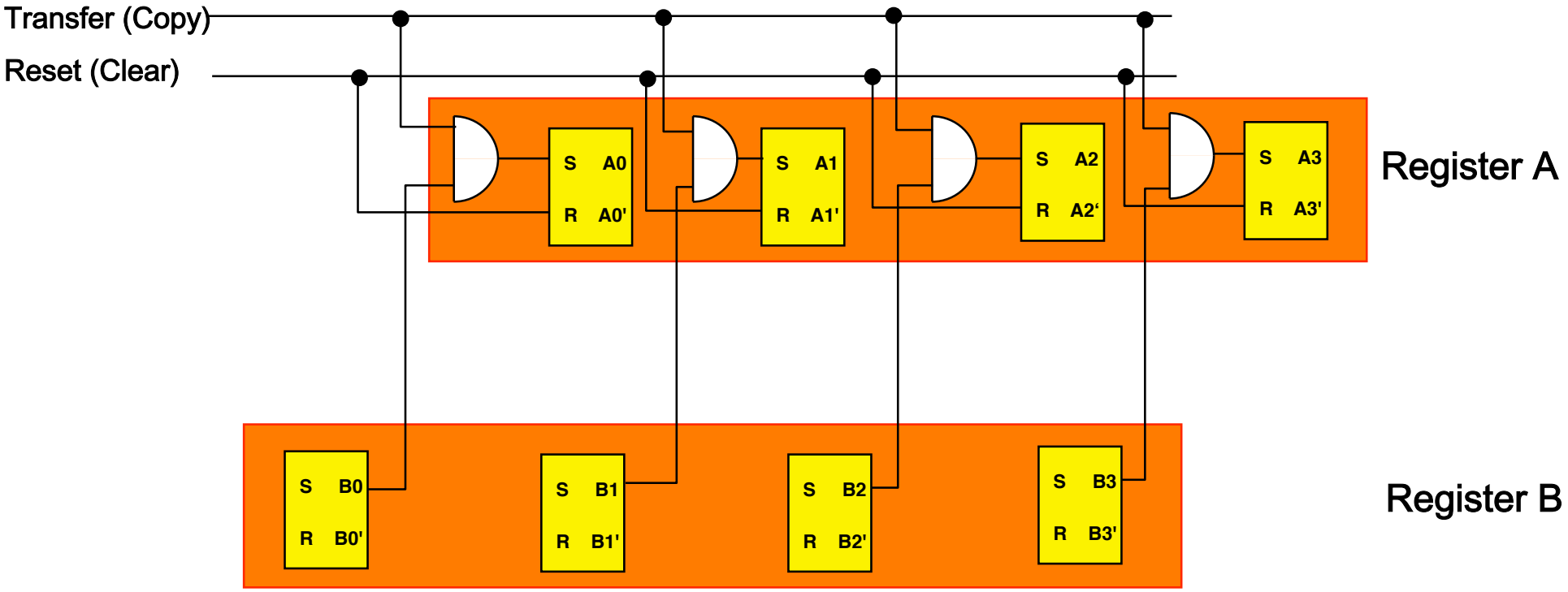
Einsatz von RS Flip-Flops in Registern

Alternative 1: Paralleler Transfer von B → A



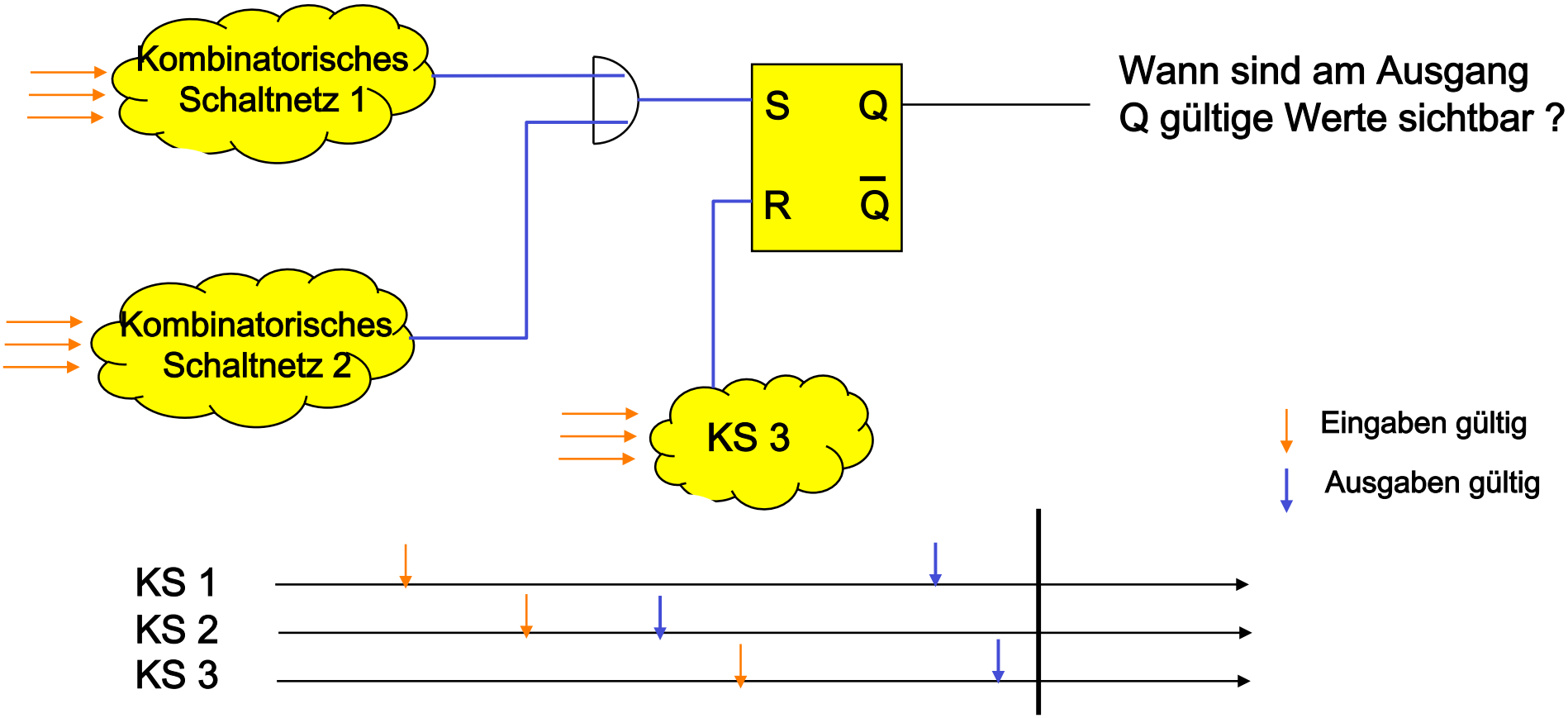
Einsatz von RS Flip-Flops in Registern

Alternative 2: Clear & Copy – Transfer von B → A

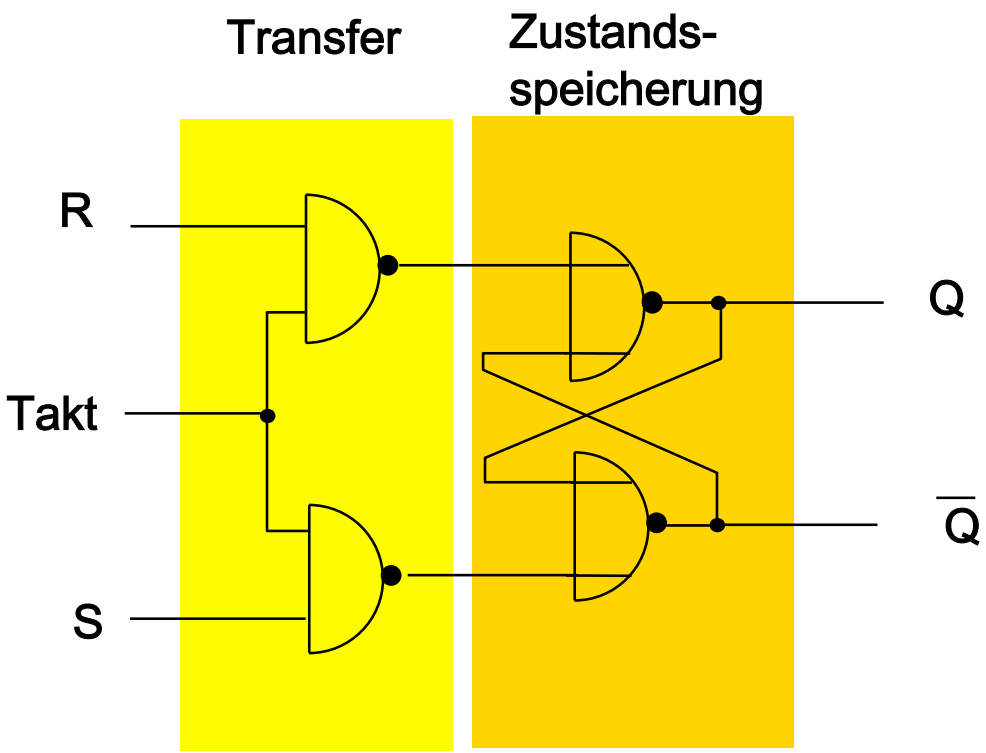


asynchrone und synchrone Schaltwerke

Das Dilemma asynchroner Schaltwerke oder die Frage: Wozu Takt?



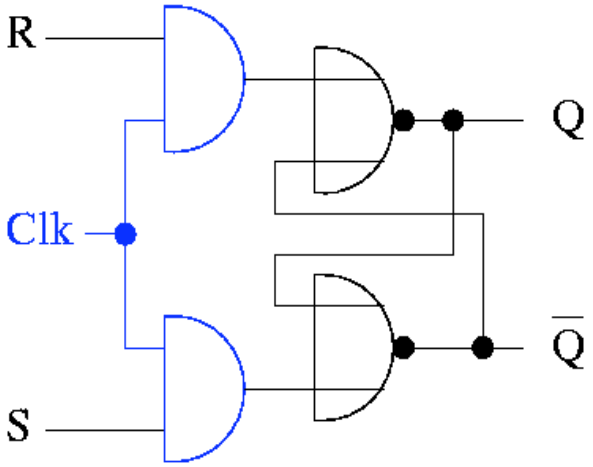
Getaktetes RS Flip-Flop



Getaktetes RS Flip-Flop

- **getaktetes RS Flip-Flop (RS-Latch):**
  - Synchrone Schaltung: Signale an R und S werden nur übernommen, wenn Taktsignal Clk aktiv ist
  - bei Clk = 0 sind R und S irrelevant (d = „*don´t care*“)
  - bei Clk = 1 stellt sich der neue Folgezustand Q´ ein

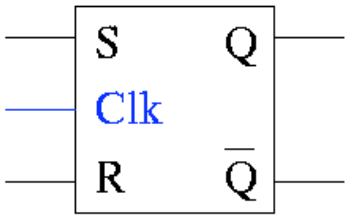
Realisierung:



Wahrheitstabelle für Q':

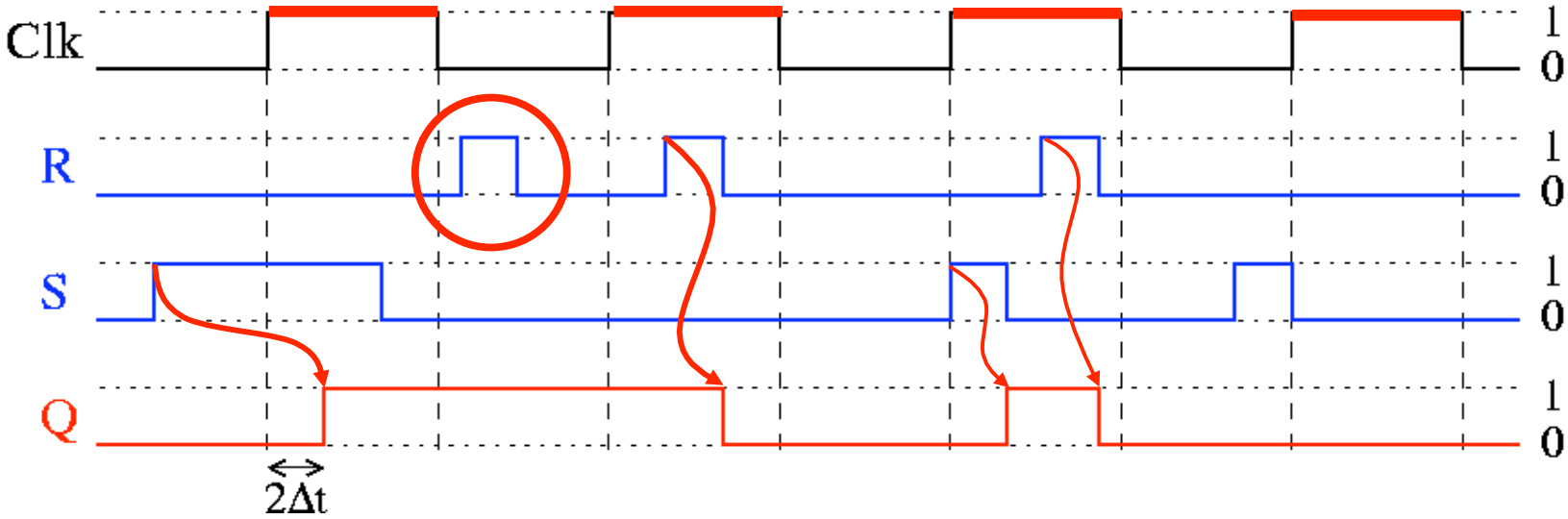
R	S	Clk	Q'
d	d	0	Q
0	0	1	Q
0	1	1	1
1	0	1	0
1	1	1	nicht erlaubt

Symbol:



Getaktetes RS Flip-Flop

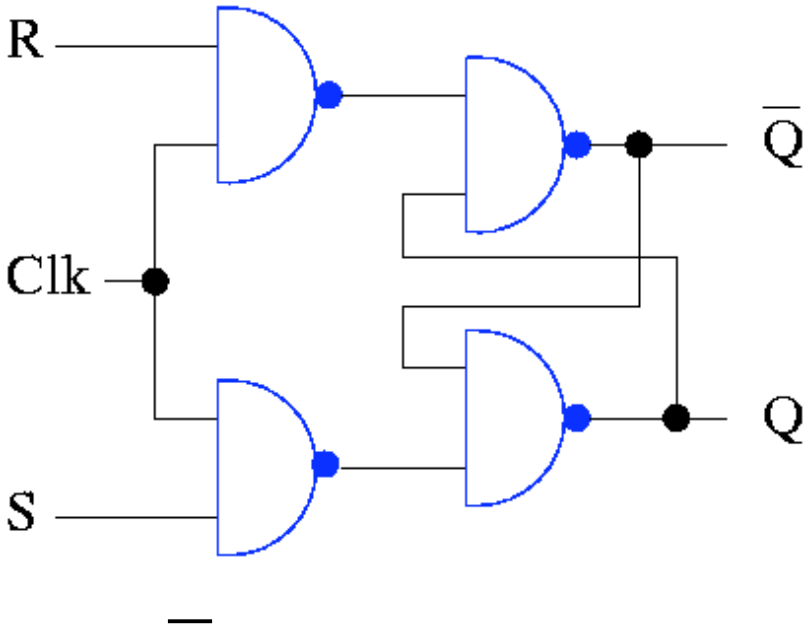
- Zeitverhalten eines getakteten RS Flip-Flops:



- Impulse auf den Eingangsleitungen R und S während der nichtaktiven Phase (Clk = 0) bleiben unberücksichtigt
- während aktiver Taktphase (Clk = 1) sind mehrere Zustands-änderungen möglich !

Getaktetes RS Flip-Flop

- ein getaktetes RS Flip-Flop läßt sich auch ausschließlich mit NAND-Gattern realisieren:



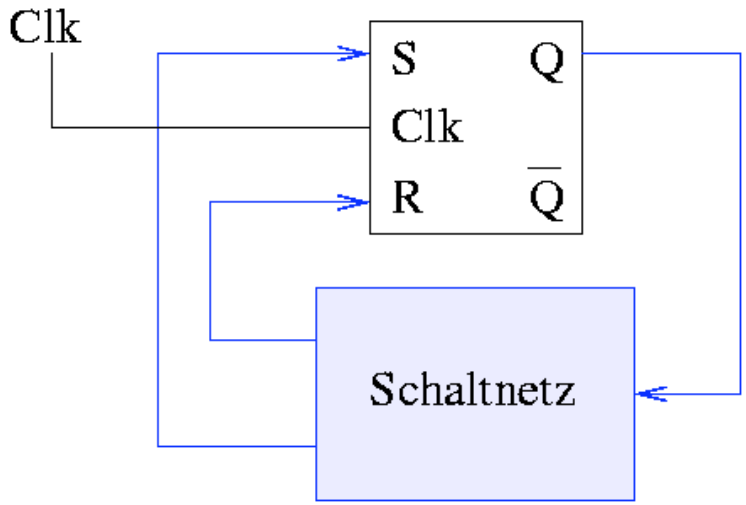
- die Ausgänge Q und $\overline{Q}$ sind hierbei jedoch vertauscht ! (vgl. de Morgansches Gesetz)

Getaktetes RS Flip-Flop

- durch die Möglichkeit mehrerer Zustandsänderungen in einer Taktphase ist das getaktete RS Flip-Flop für viele Anwendungen ungeeignet
- Beispiel:

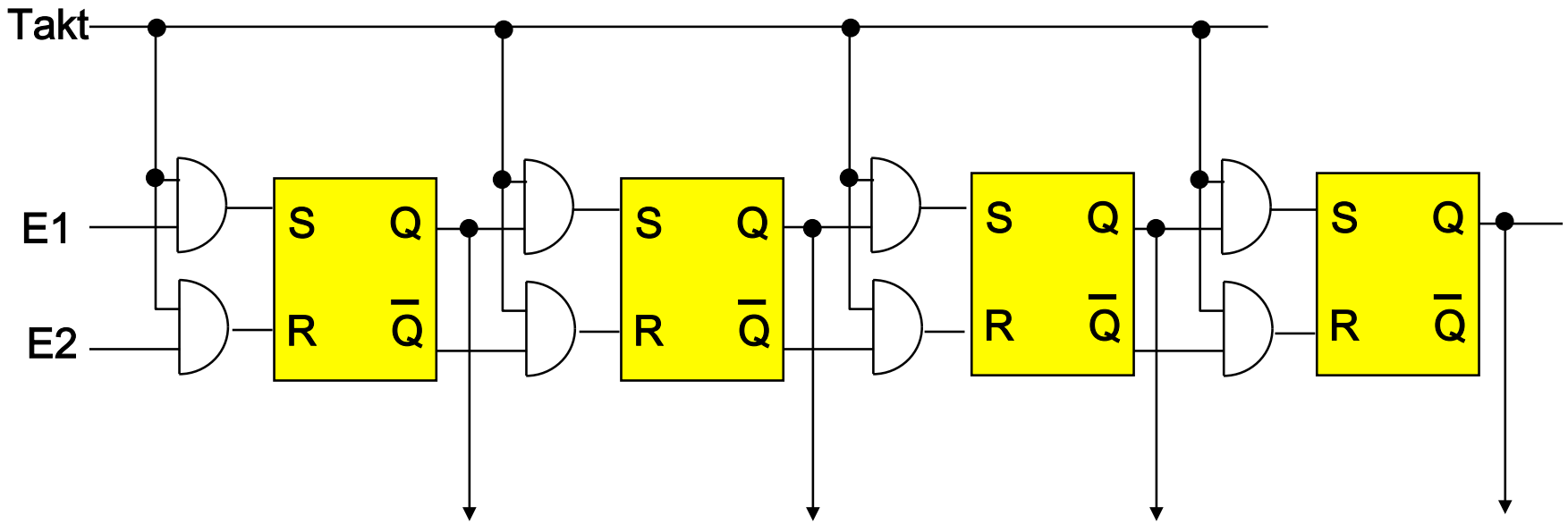
  Rückkopplung vom Ausgang des Flip-Flops zu den Eingängen über ein Schaltnetz

  ⇒ selbst bei kurzen Taktphasen sind mehrere ungewollte Rück-kopplungen je Takt möglich



Getaktetes RS Flip-Flop

Versuch: Realisierung eines Schieberegisters

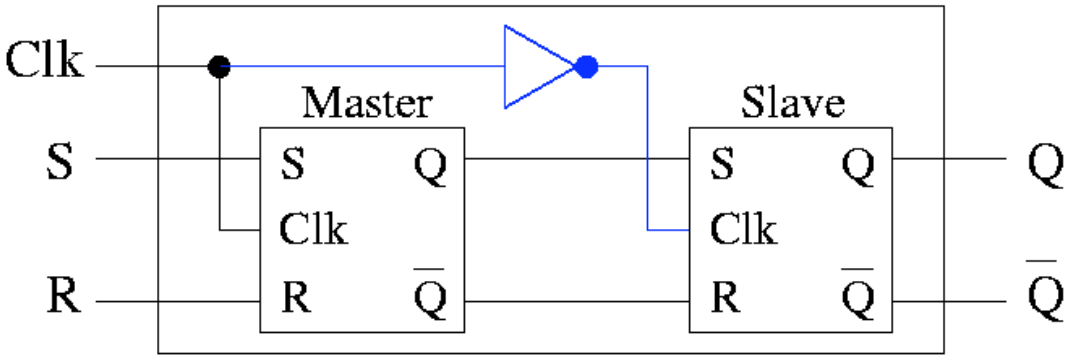


Pegel-getriggertes (Level-triggered) Flip-Flop.
Problem: Mehrere Änderungen sind während eines Zeitintervalls möglich.

Gewünscht: Flip-Flop Variante, die Änderungen nur zu einem definierten Zeitpunkt zuläßt

Master-Slave Flip-Flop

- ein **Master-Slave RS Flip-Flop** besteht aus 2 hintereinander-geschalteten getakteten RS Flip-Flops (als „*Master*“ und als „*Slave*“ bezeichnet)
  - zusätzlicher **Inverter** negiert Taktsignal für „*Slave*“



  - „*Master*“ übernimmt Eingangswerte bei **Clk = 1** („*Slave*“ ändert sich nicht)
  - „*Slave*“ übernimmt Werte vom „*Master*“ bei **Clk = 0** („*Master*“ ändert sich nicht)

Master-Slave Flip-Flop

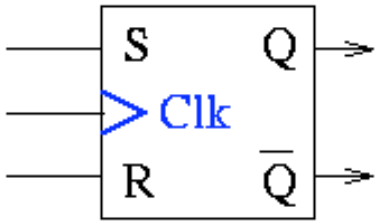
- durch spezielle Schaltungstechnik kann erreicht werden, daß auch die Eingangsleitungen nur
  - bei steigender Flanke oder
  - bei fallender Flanke

  berücksichtigt werden!

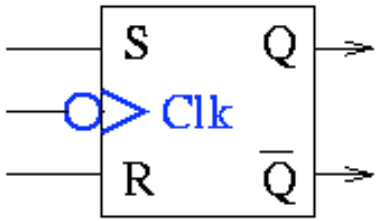
  ⇒ positiv oder negativ flanken-getriggertes RS Flip-Flop

  (positive Flanke = steigende Flanke, negative Flanke = fallende Flanke)

Symbol für positiv flankengetriggertes RS Flip–Flop:



Symbol für negativ flankengetriggertes RS Flip–Flop:

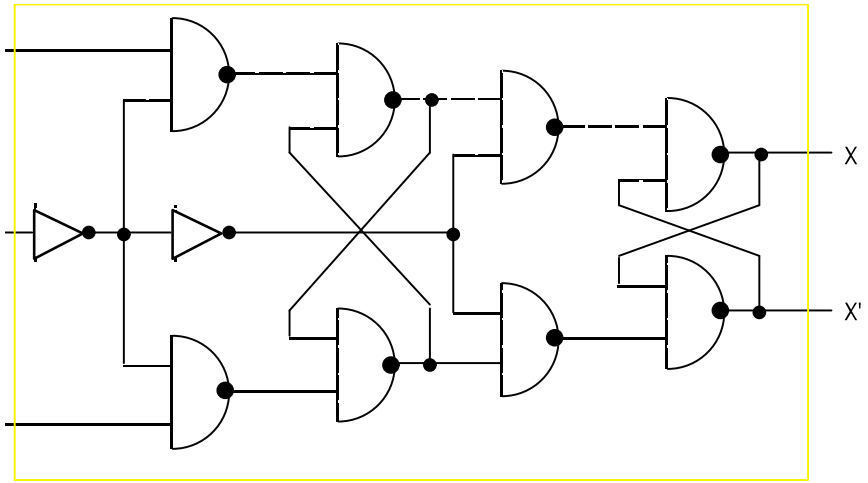


Master-Slave Flip-Flop

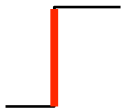
S

Takt

R



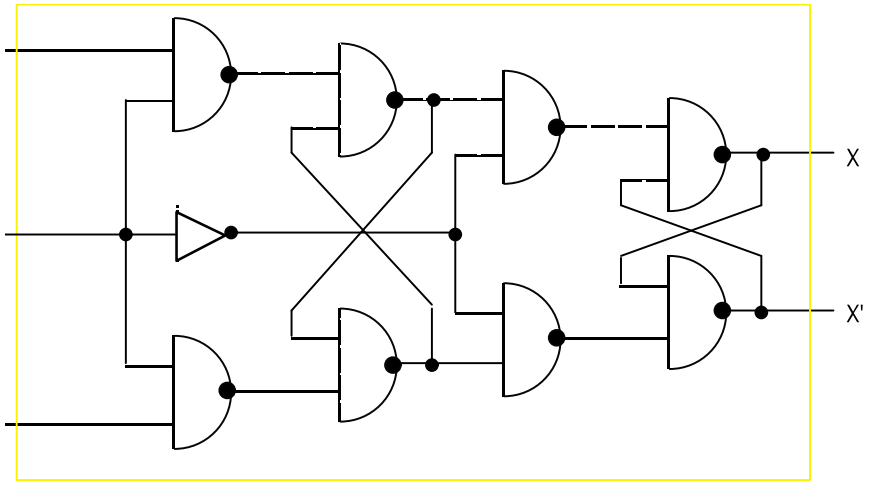
schaltet (trigger) bei
positiver Signalflanke



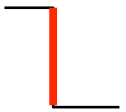
S

Takt

R

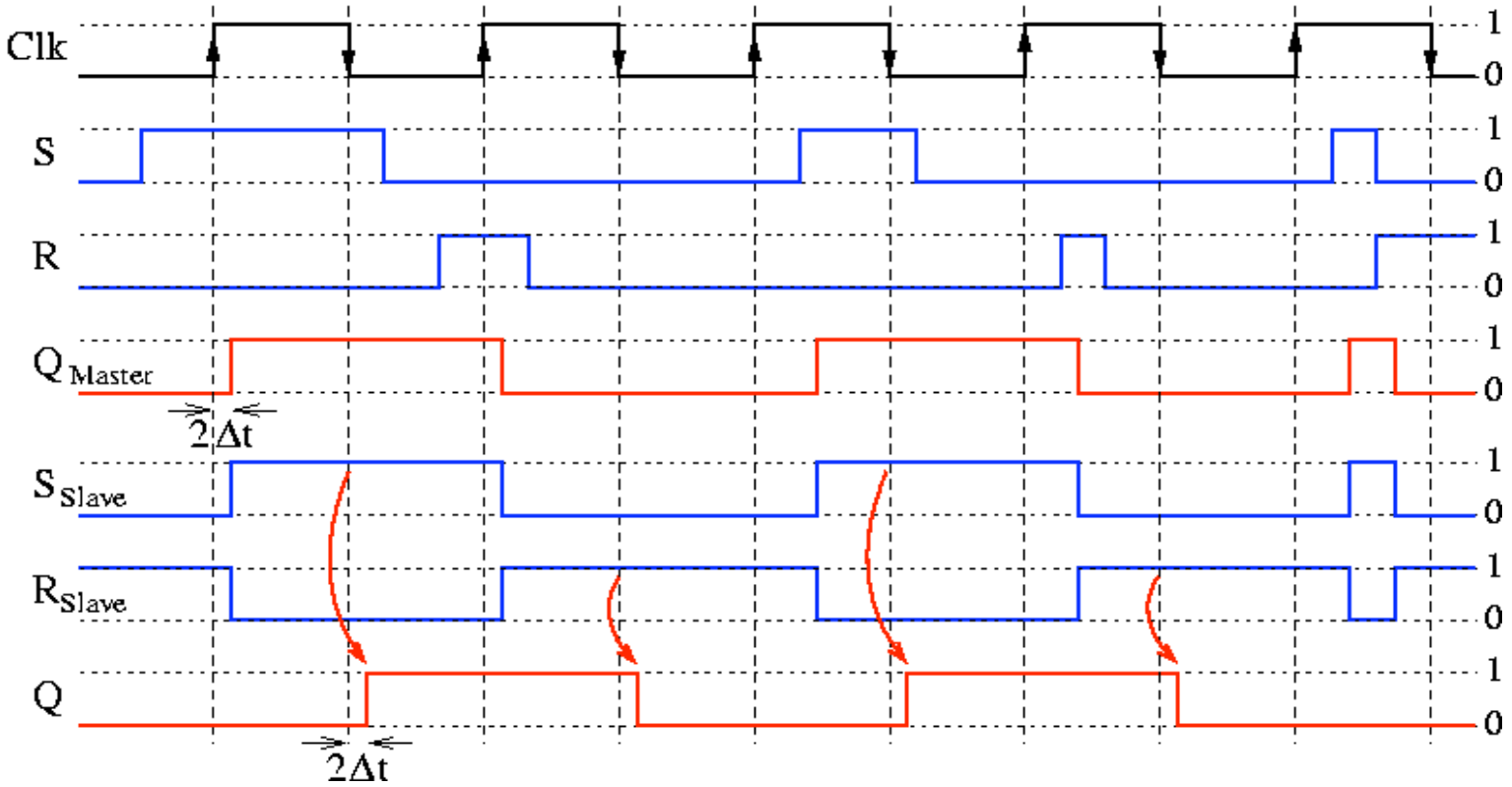


schaltet bei
negativer Signalflanke



Master-Slave Flip-Flop

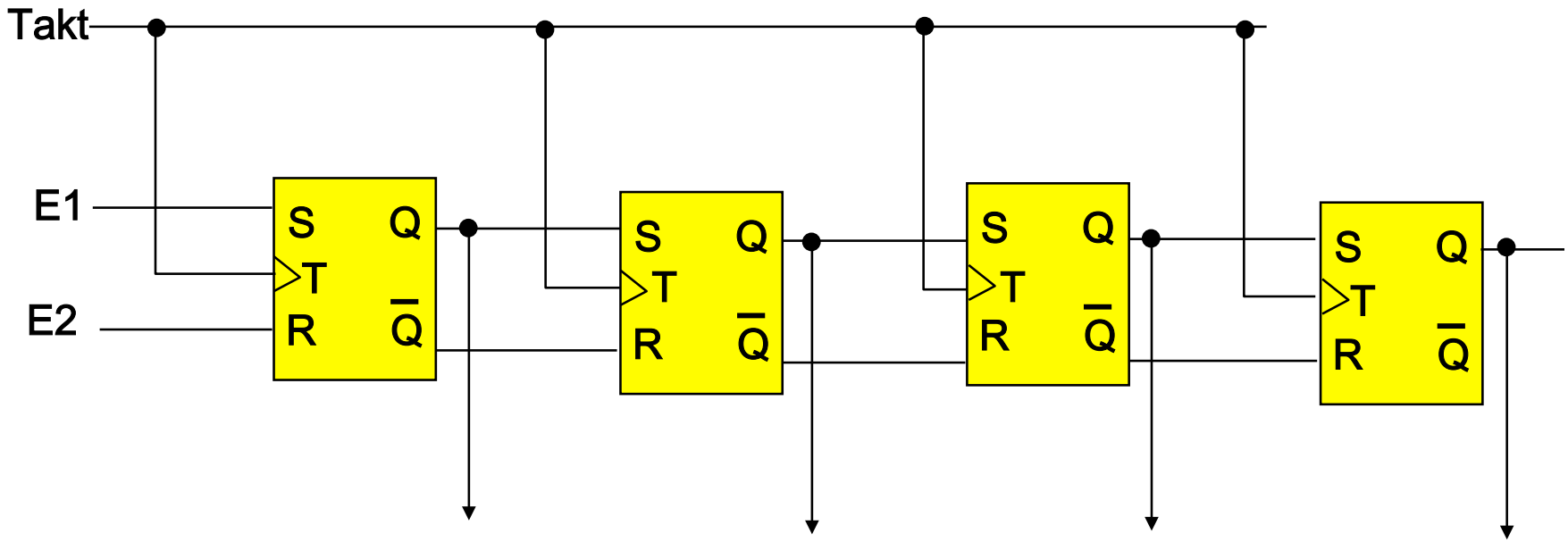
- Zeitverhalten des Master-Slave RS Flip-Flops:



- Master akzeptiert Änderungen an R und S während Clk = 1, Slave übernimmt Q_{Master} bei folgender fallender Taktflanke!

Master-Slave Flip-Flop

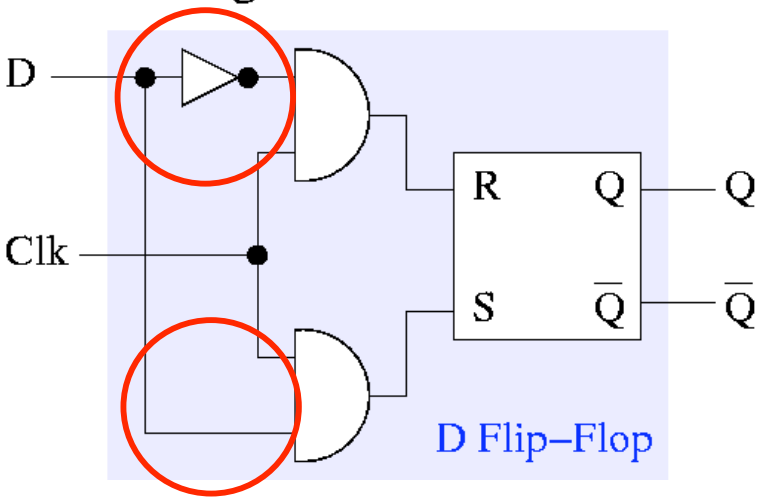
2. Versuch: Realisierung eines Schieberegisters mit Master-Slave Flip-Flops



D Flip-Flops

- D (von Delay) Flip-Flop:
  - bei Clk = 1 wird intern S = D und R = $\overline{D}$ gesetzt
  - hierdurch wird der unerlaubte Zustand R = S = 1 stets vermieden!
  - bei Clk = 0 bleibt Zustand unverändert
  - bei Clk = 1 ergibt sich der neue Folgezustand Q´ = D

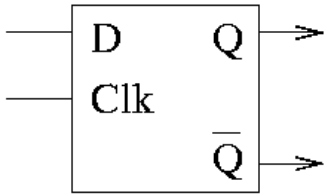
Realisierung:



Wahrheitstabelle:

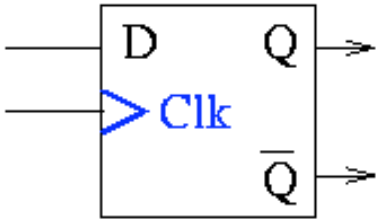
D	Clk	Q'
0	0	Q
0	1	0
1	0	Q
1	1	1

Symbol:

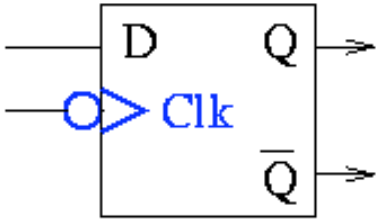


D Flip-Flops

- **Flankengetriggertes D Flip-Flop:**
  - D Flip-Flops werden meistens nur in der flankengetriggerten Version benutzt, d.h. lediglich bei Auftreten der entsprechenden Taktflanke wird das Signal vom Eingang D übernommen
  - D Flip-Flop mit positiver Flankentriggerung:

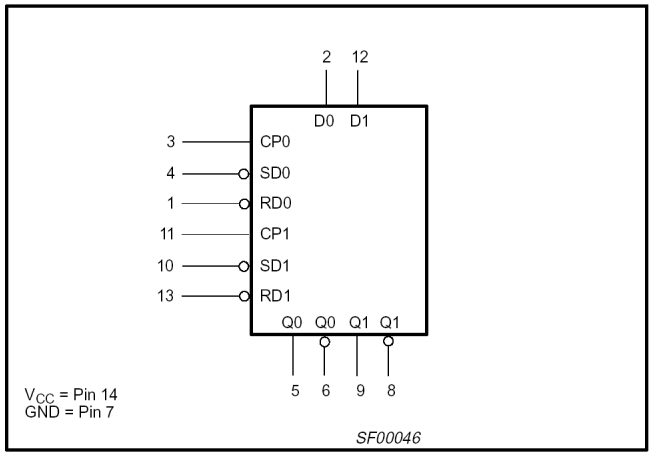


  - D Flip-Flop mit negativer Flankentriggerung:

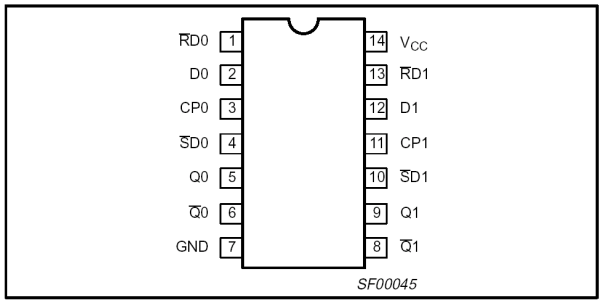


7474 Dual D Flip-Flop

LOGIC SYMBOL



PIN CONFIGURATION



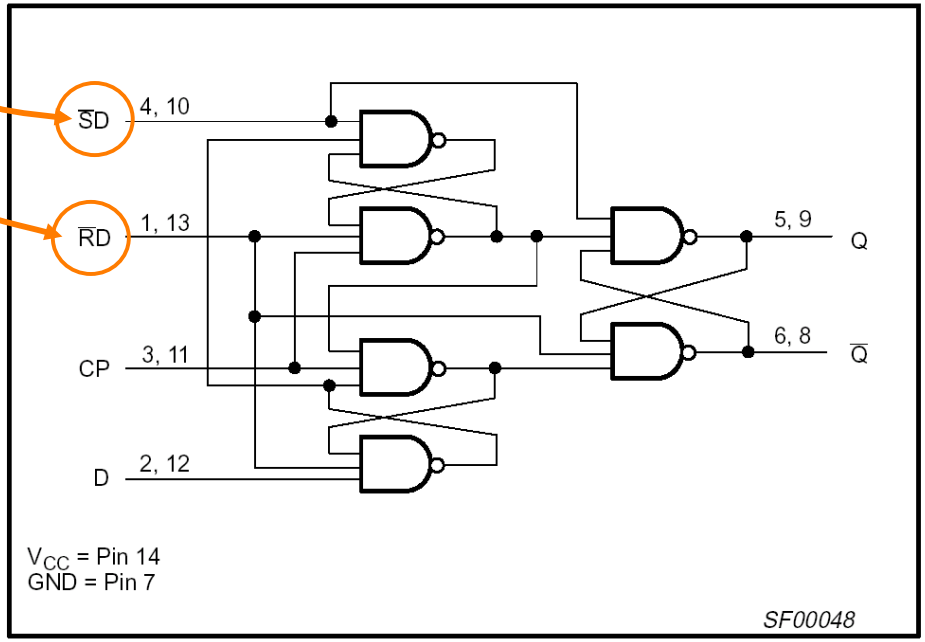
FUNCTION TABLE

INPUTS				OUTPUTS		OPERATING MODE
$\overline{SD}$	$\overline{RD}$	CP	D	Q	$\overline{Q}$	
L	H	X	X	H	L	Asynchronous set
H	L	X	X	L	H	Asynchronous reset
L	L	X	X	H	H	Undetermined*
H	H	↑	h	H	L	Load "1"
H	H	↑	l	L	H	Load "0"
H	H	↓	X	NC	NC	Hold

NOTES:

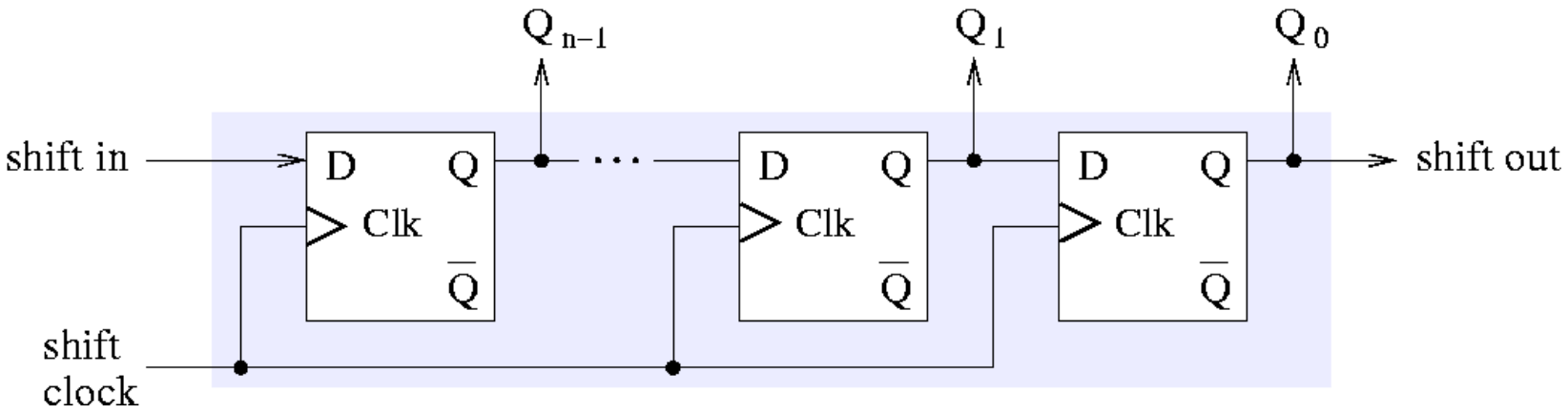
H = High voltage level
h = High voltage level one setup time prior to low-to-high clock transition
L = Low voltage level
l = Low voltage level one setup time prior to low-to-high clock transition
NC = No change from the previous setup
X = Don't care
↑ = Low-to-high clock transition
↓ = Not low-to-high clock transition
* = This setup is unstable and will change when either set or reset return to the high level.

LOGIC DIAGRAM



Schaltwerke mit D Flip-Flops

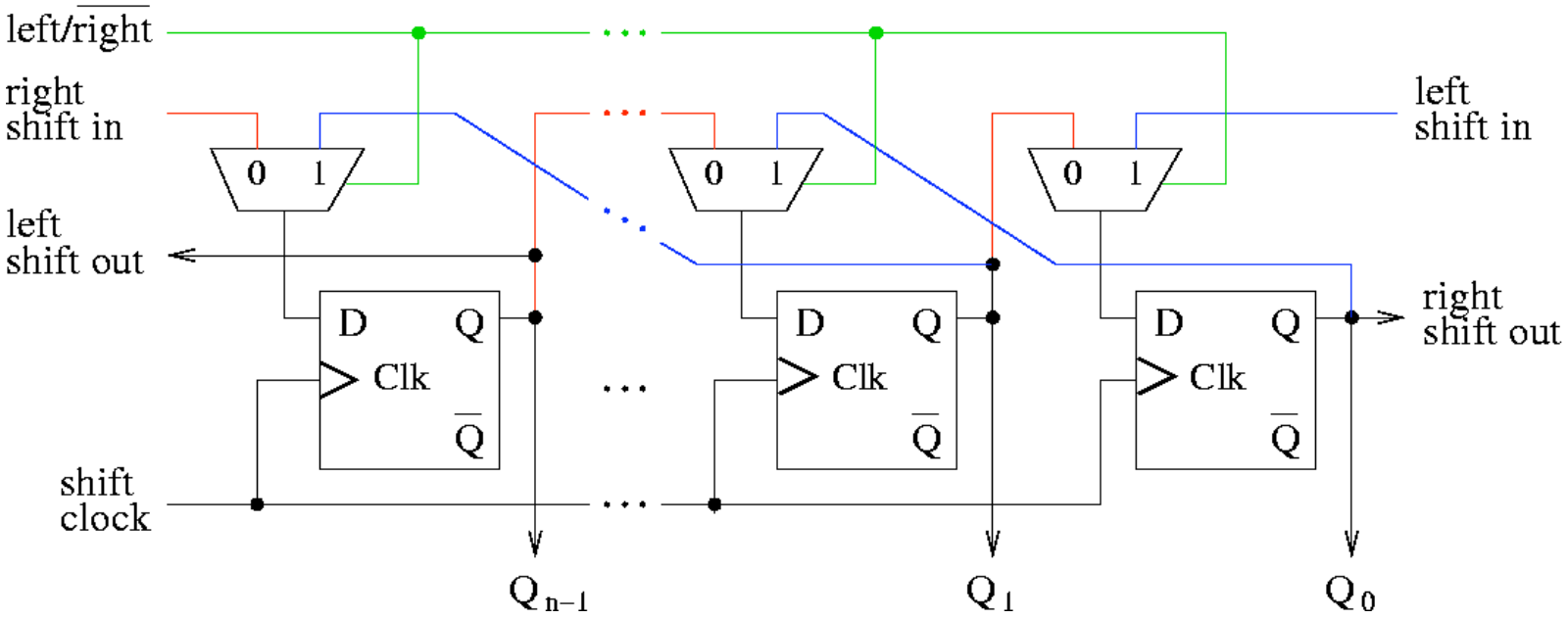
- *n*-Bit Schieberegister:



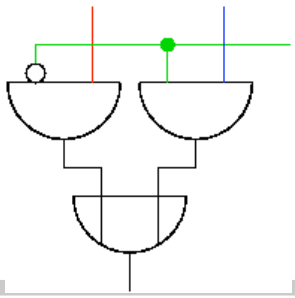
- in jedem Takt werden Binärwerte um eine Position nach rechts geschoben
- Anwendungen: Seriell-/Parallelwandlung, Teil arithmetischer Operationen

Schaltwerke mit D Flip-Flops

- ***n*-Bit Links/Rechts-Schieberegister:**

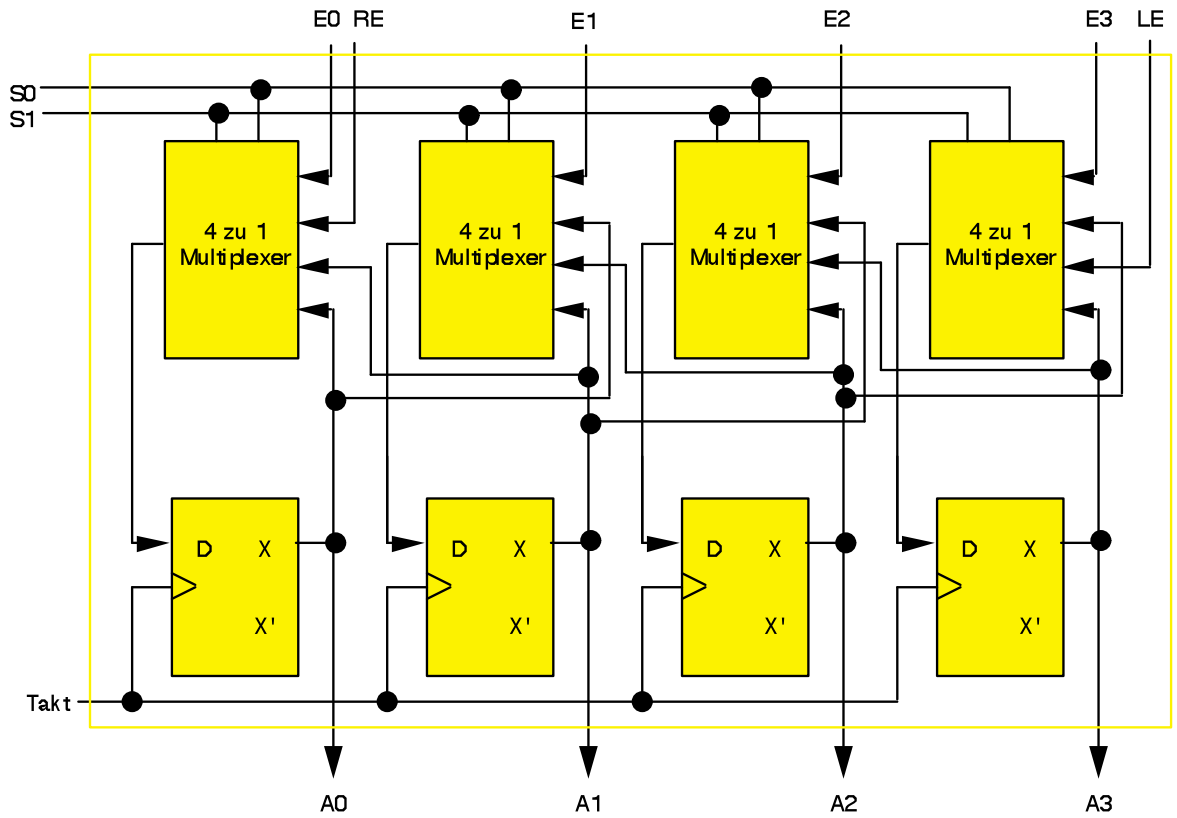


- Realisierung des 1-aus-2 Multiplexers:



Schaltwerke mit D Flip-Flops

Rechts- Links Schieberegister mit paralleler Ladeoption



Steuersignale:

S1	S	Operation+
0	0	keine Veränderung
0	1	nach links schieben
1	0	nach rechts schieben
1	1	paralleles Laden

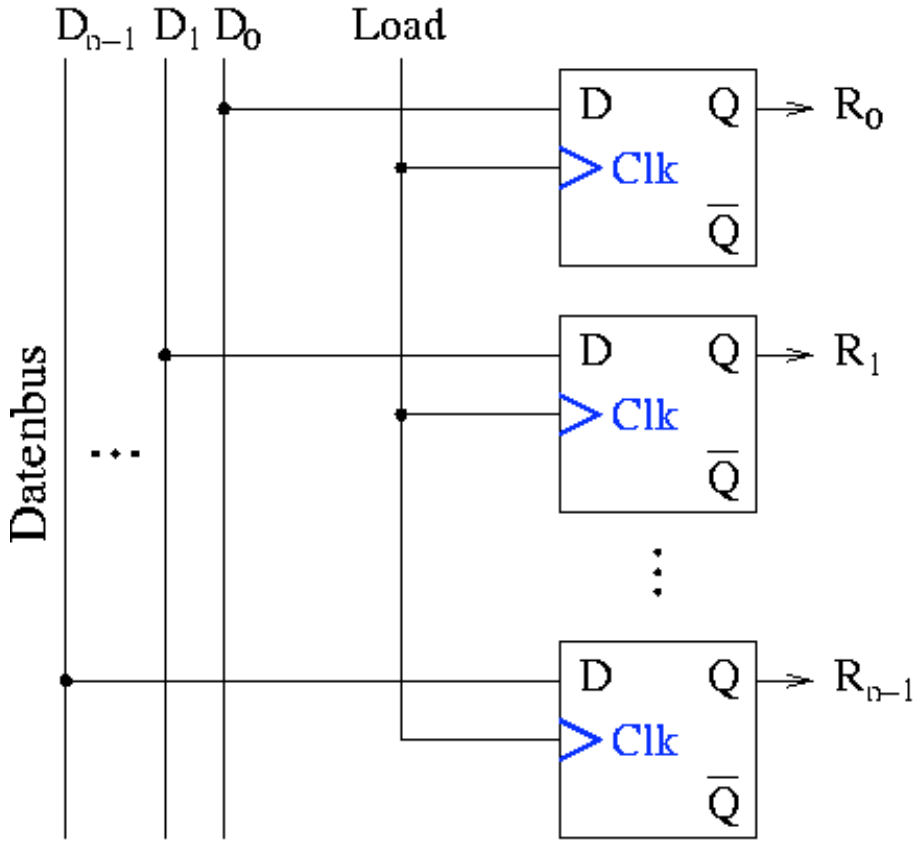
Dateneingänge:

E0..E3 : Parallele Ladeeingänge
RE: Eingang Rechtsshift
LE: Eingang Linksshift

Schaltwerke mit D Flip-Flops

- flankengetriggerte D Flip-Flops dienen als Grundbaustein für ein *n*-Bit Register:

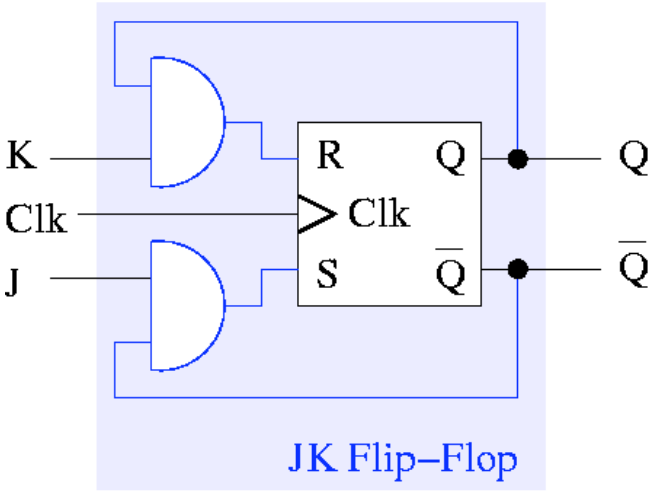
Daten vom Datenbus werden bei steigender Flanke des LOAD-Signals in das Register geladen



JK Flip-Flops

- JK Flip-Flop:
  - basierend auf flankengetriggertem RS Flip-Flop
  - jedoch Nutzung der nicht benötigten Eingangskombination 1, 1 für eine Invertierung von Q („Toggle“)

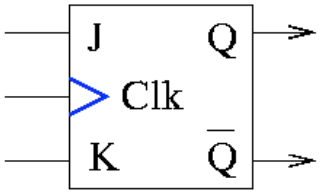
Realisierung mit positiv flanken–getriggertem RS–Flipflop:



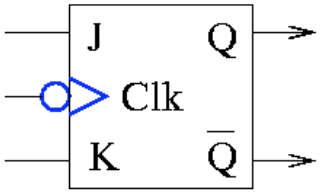
Wahrheitstabelle:

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

Symbol bei positiver Flankentriggerung



Symbol bei negativer Flankentriggerung

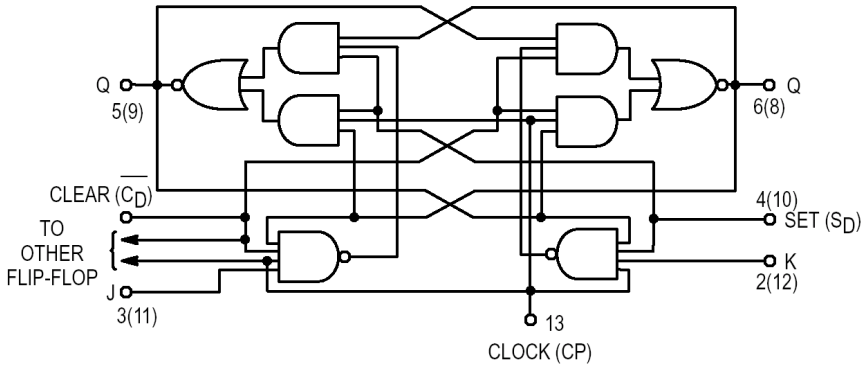


MOTOROLA

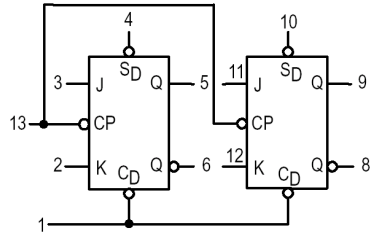
SN54/74LS114A

DUAL JK NEGATIVE EDGE-TRIGGERED FLIP-FLOP

LOGIC DIAGRAM (Each Flip-Flop)



LOGIC SYMBOL



V_{CC} = PIN 14
GND = PIN 7

MODE SELECT — TRUTH TABLE

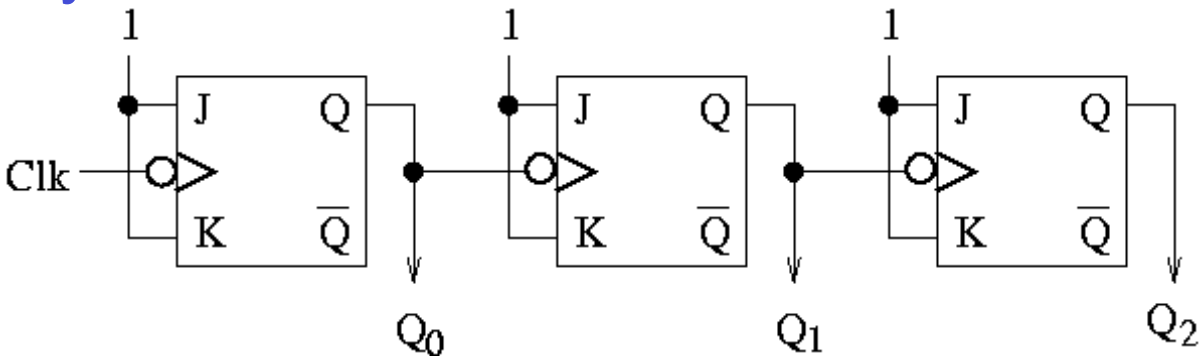
OPERATING MODE	INPUTS				OUTPUTS	
	$\overline{S_D}$	$\overline{C_D}$	J	K	Q	$\overline{Q}$
Set	L	H	X	X	H	L
Reset (Clear)	H	L	X	X	L	H
*Undetermined	L	L	X	X	H	H
Toggle	H	H	h	h	q	q
Load “0” (Reset)	H	H	l	h	L	H
Load “1” (Set)	H	H	h	l	H	L
Hold	H	H	l	l	q	q

* Both outputs will be HIGH while both S_D and C_D are LOW, but the output states are unpredictable if S_D and C_D go HIGH simultaneously.

H, h = HIGH Voltage Level
L, l = LOW Voltage Level
X = Don’t Care
l, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

Typische Schaltwerke

- **Asynchroner 3-Bit Binärzähler:**

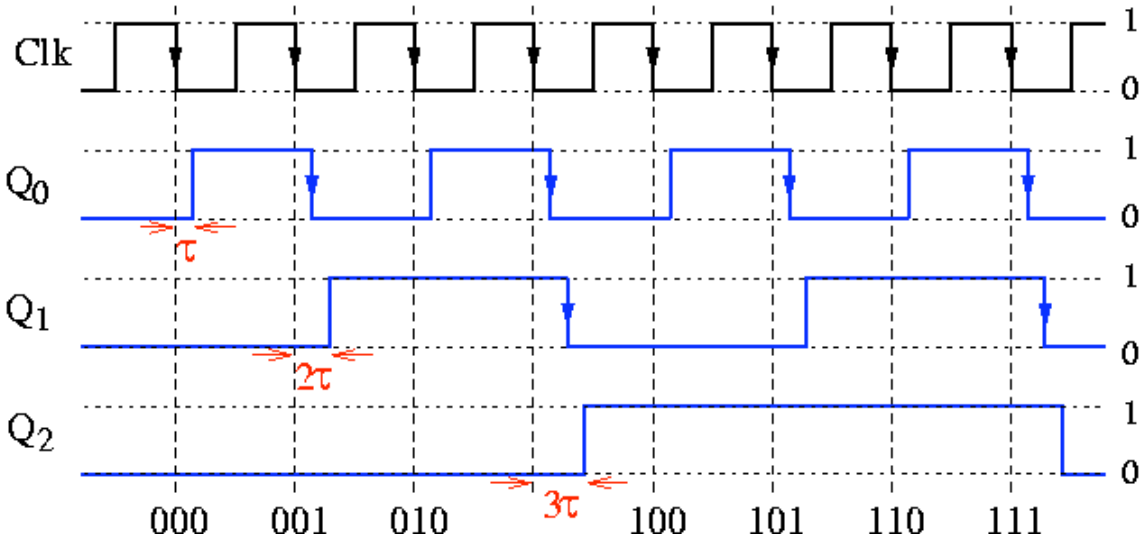


zählt fallende Taktflanken!

- Zeitverhalten:

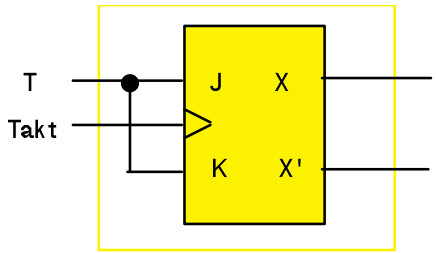
  τ = Verzögerung eines flanken-getriggerten JK-Flipflops

  $\Rightarrow \tau$ legt maximale Taktfrequenz fest



T Flip-Flops

kann als JK Flip-Flop aufgefasst werden, bei dem J und K fest miteinander verbunden sind.



Eingabe	nächster Zustand
T	
0	keine Änderung
1	invertiert

Beschreibung von Flip-Flops

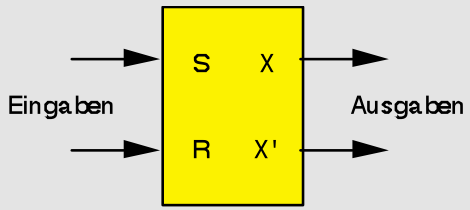
WAHRHEITSTABELLE (englisch: characteristic table):

→zeigt den Zustand nach einem Takt (wurde bisher verwendet)

INVERTIERTE WAHRHEITSTABELLE (englisch: excitation table) (auch Zustandsübergangstabelle):

→ zeigt die Eingaben, die notwendig sind, um eine bestimmte Zustandsänderung herbeizuführen.

RS Flip-Flop



Wahrheitstabelle

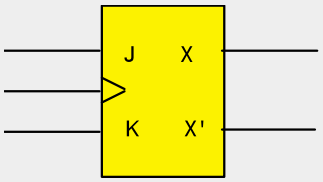
S	R	X(t + 1)
0	0	X(t)
0	1	0
1	0	1
1	1	nicht definiert

Invertierte Wahrheitstabelle

X(t)	X(t + 1)	S	R
0	0	0	d
0	1	1	0
1	0	0	1
1	1	d	0

Beschreibung von Flip-Flops

JK Flip-Flop



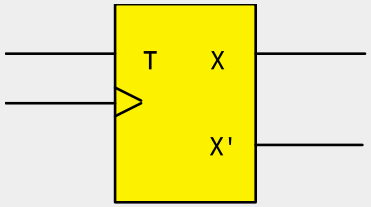
Wahrheitstabelle

J	K	X(t + 1)
0	0	X(t)
0	1	0
1	0	1
1	1	X'(t)

Invertierte Wahrheitstabelle

X(t)	X(t + 1)	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

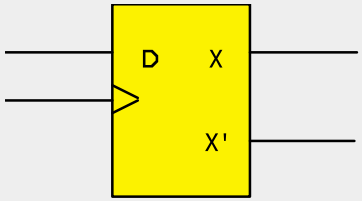
T Flip-Flop



T	X(t + 1)
0	X(t)
1	X'(t)

X(t)	X(t + 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

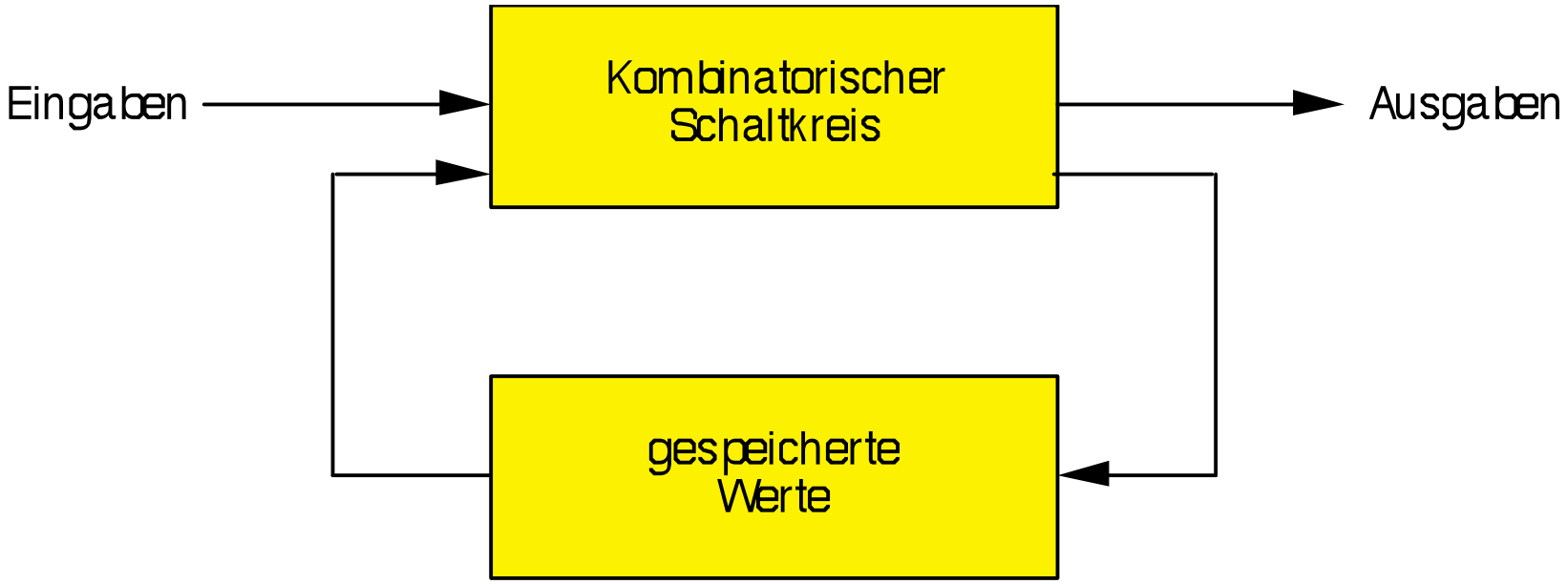
D Flip-Flop



D	X(t + 1)
0	0
1	1

X(t)	X(t + 1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Entwurf sequentieller Schaltwerke



- Folgezustand ergibt sich aus dem momentanen Zustand in Form gespeicherter Werte und den Eingaben.
- Formal kann das Schaltwerk als endlicher Automat beschrieben werden.
- Graphendarstellung als Zustandsdiagramm eignet sich als „High Level“ Spezifikation.

Entwurf eines Schaltwerks

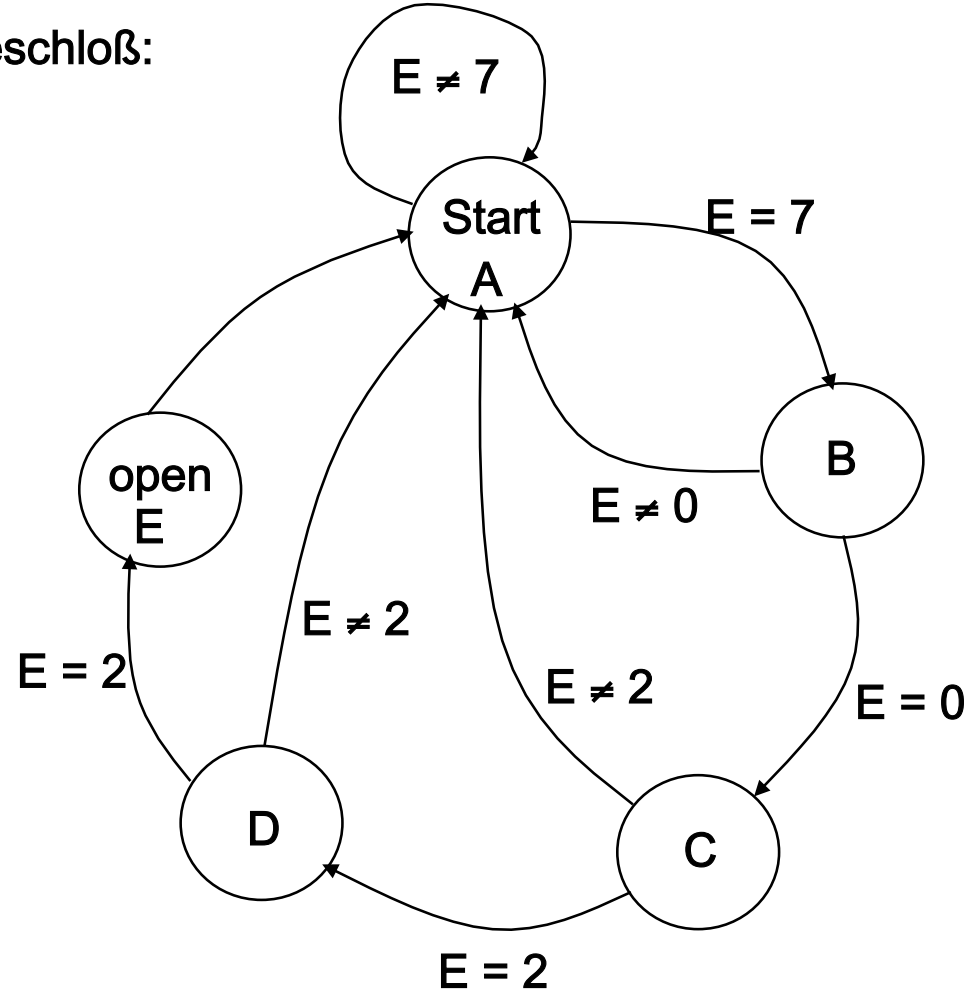
- Wie kann man systematisch ein synchrones Schaltwerk ausgehend von der Problembeschreibung entwerfen ?
- Verwendung eines endlichen Zustandsautomaten als zugrunde liegendes Modell

- Automat ist gekennzeichnet durch:
  - beliebige (jedoch endliche) Menge von Zuständen
  - Zustandsübergänge in jedem Takt abhängig von Eingangssignalen
  - Ausgangssignale werden durch ein Schaltnetz generiert

Entwurf sequentieller Schaltwerke

Beispiel Codeschloß:

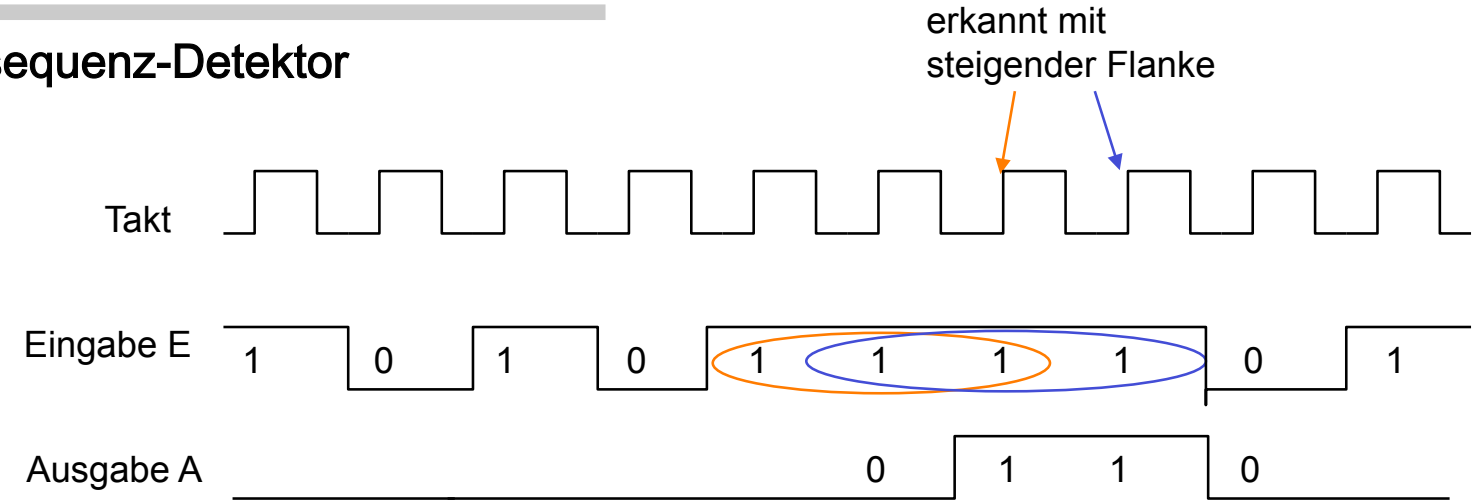
Code: 7022



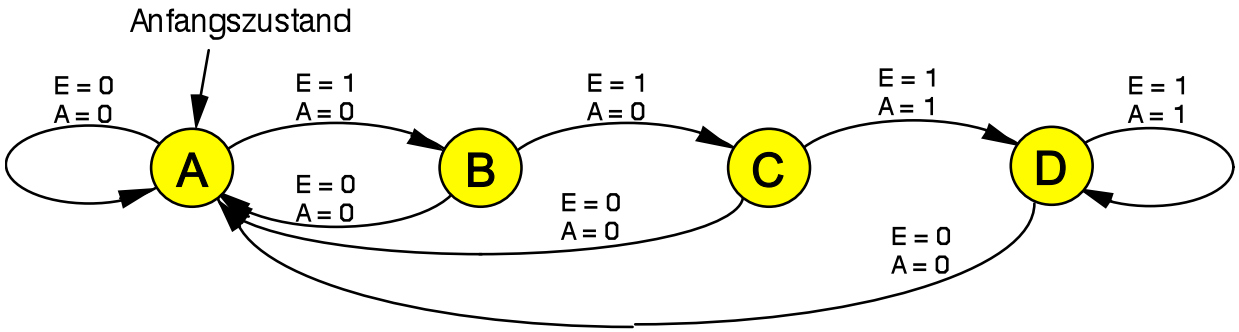
Entwurf eines Schaltwerks

Beispiel 1: Ein Binärsequenz-Detektor

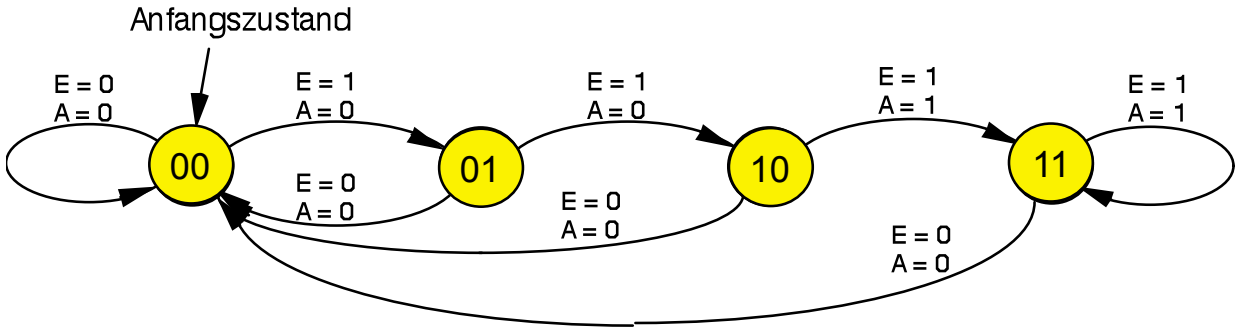
Verhalten:
3 aufeinanderfolgende „Einsen“ sollen erkannt werden.



Zustandsdiagramm:



Entwurf eines Schaltwerks



Zustandstabellen:

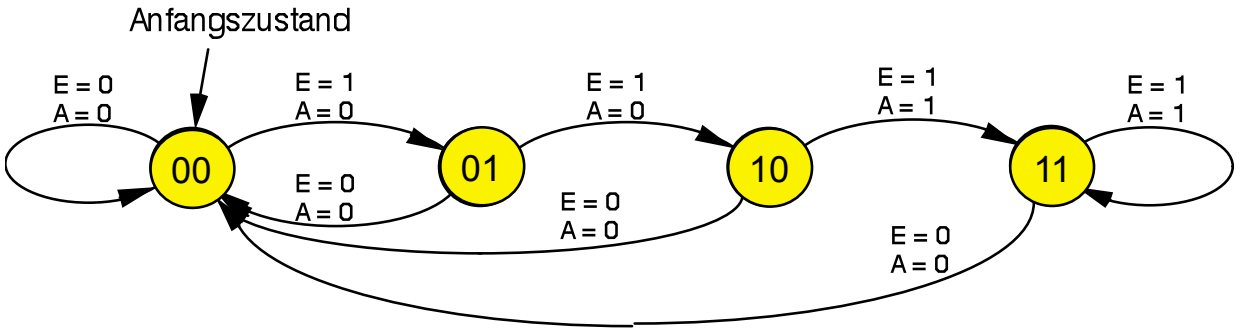
Zustände wie in Abb. bezeichnet

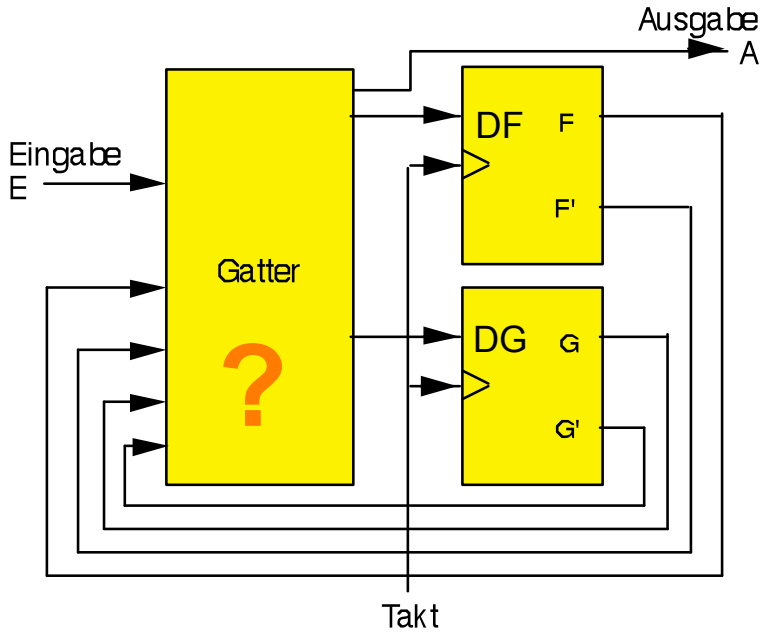
jetziger Zustand	nächster Zustand		Ausgaben	
	E = 0	E = 1	E = 0	E = 1
A	A	B	0	0
B	A	C	0	0
C	A	D	0	1
D	A	D	0	1

Zustände kodiert → 2 Flip-Flops

jetziger Zustand	nächster Zustand		Ausgaben	
	E = 0	E = 1	E = 0	E = 1
FG	FG	FG	A	A
00	00	01	0	0
01	00	10	0	0
10	00	11	0	1
11	00	11	0	1

Entwurf eines Schaltwerks





Realisierung mit D Flip-Flops

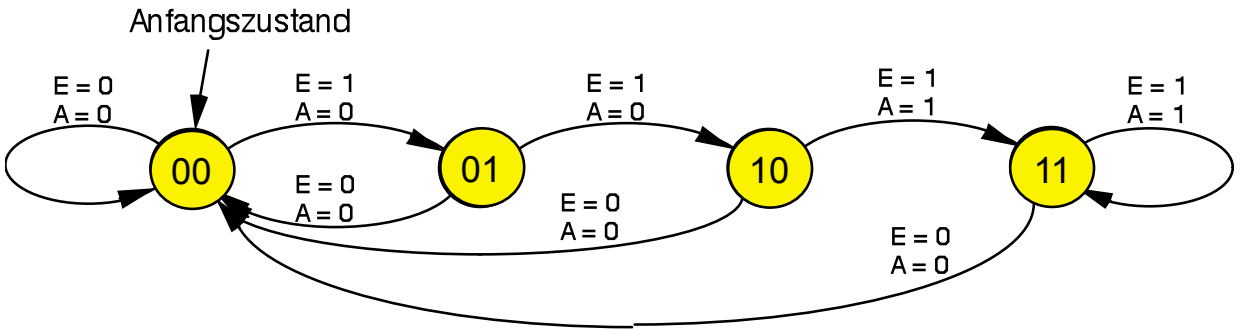
WT

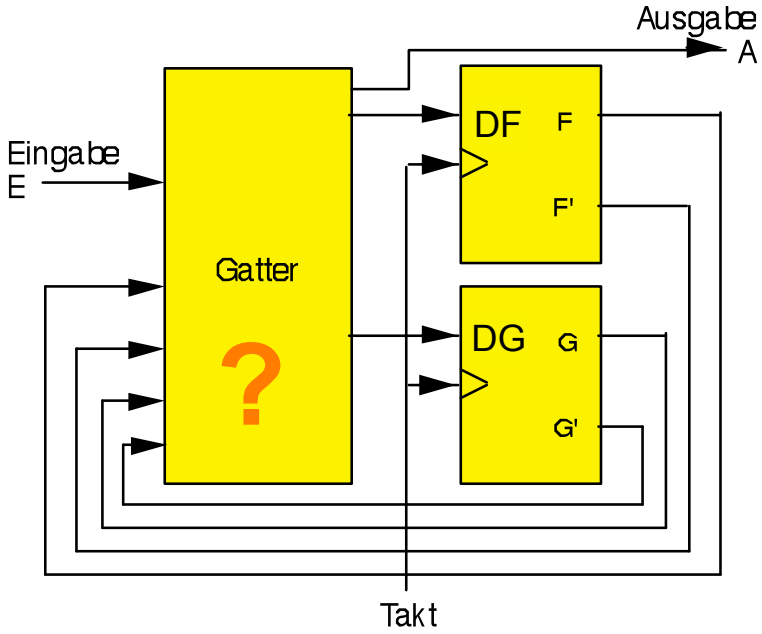
D	X(t + 1)
0	0
1	1

WT (invertiert)

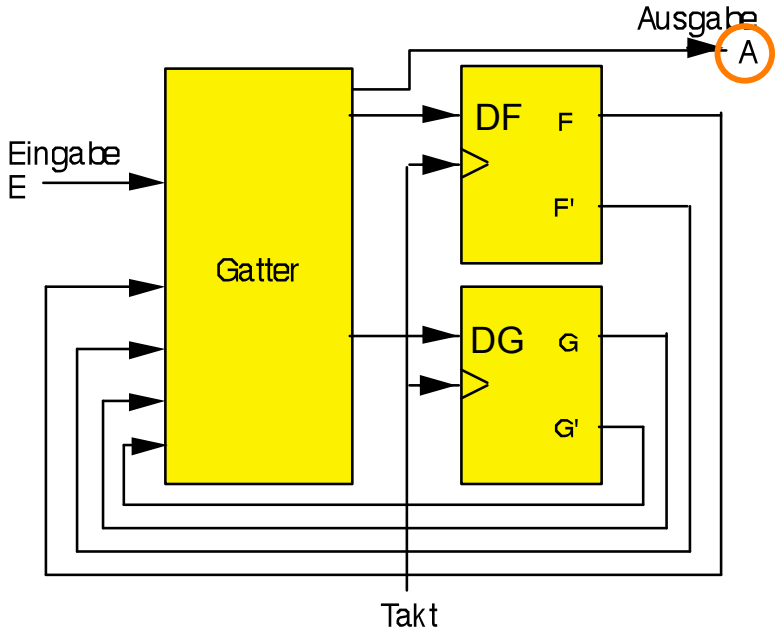
X(t)	X(t + 1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Entwurf eines Schaltwerks





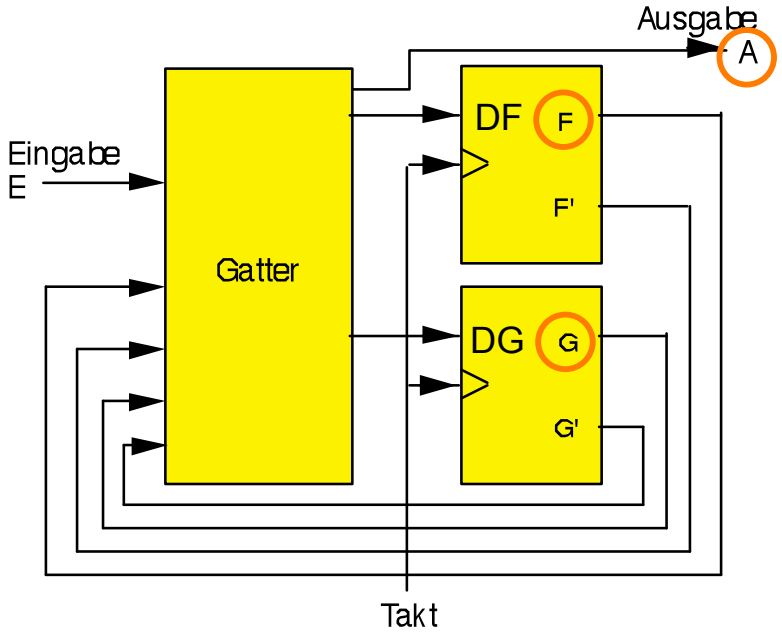
Entwurf eines Schaltwerks



jetziger Zustand	nächster Zustand		Ausgaben	
	E = 0	E = 1	E = 0	E = 1
FG	FG	FG	A	A
00	00	01	0	0
01	00	10	0	0
10	00	11	0	1
11	00	11	0	1

$A = EFG' + EFG$

Entwurf eines Schaltwerks



jetziger Zustand	nächster Zustand		Ausgaben	
	E = 0	E = 1	E = 0	E = 1
FG	FG	FG	A	A
00	00	01	0	0
01	00	10	0	0
10	00	11	0	1
11	00	11	0	1

$A = EFG' + EFG$

$DF = EF'G + EFG' + EFG$

$DG = EF'G' + EFG' + EFG$

E	F'G'	F'G	FG	FG'
E'	0	0	0	0
E	0	0	1	1

$A = EF$

E	F'G'	F'G	FG	FG'
E'	0	0	0	0
E	0	1	1	1

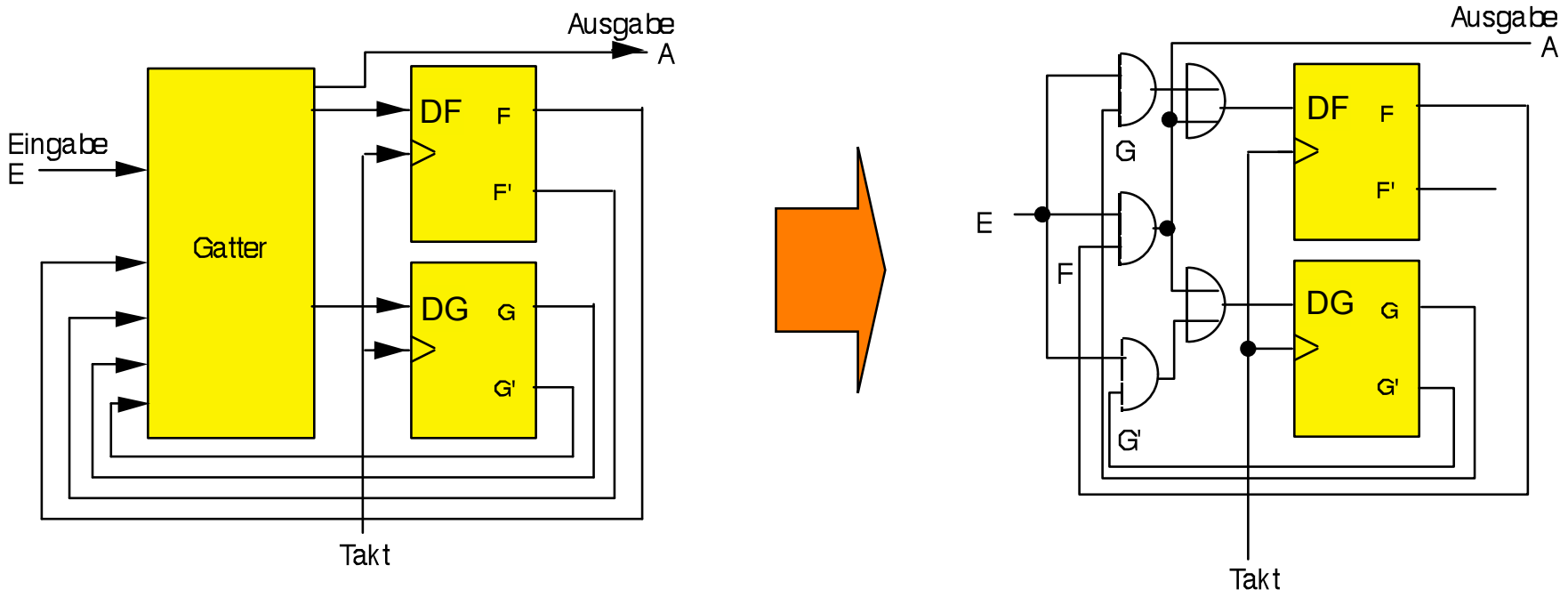
$DF = EF + EG$

E	F'G'	F'G	FG	FG'
E'	0	0	0	0
E	1	0	1	1

$DG = EF + EG'$

Entwurf eines Schaltwerks

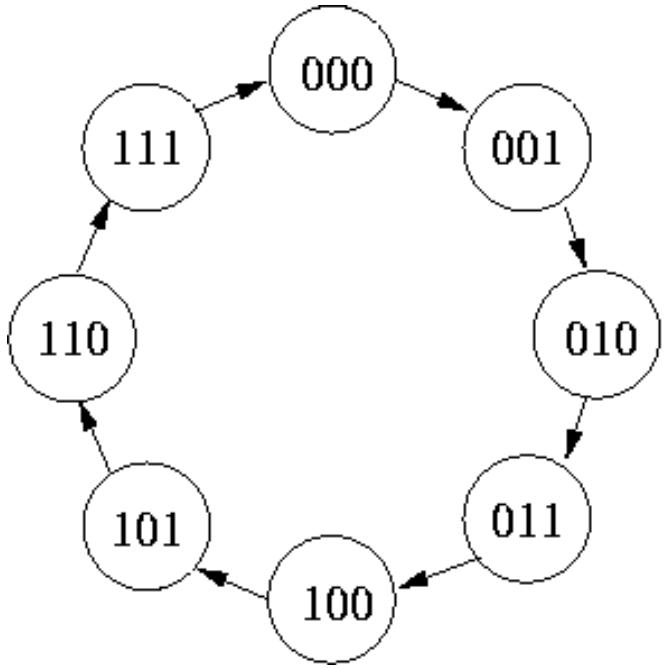
Der Binärsequenz-Detektor



1. Schritt: Spezifikation des Zustandsdiagramms
2. Schritt: Zuordnung von Zuständen zu Flip-Flop Belegungen (Assignment)
3. Schritt: Erstellung der Wahrheitstafel für Zustände und Ausgaben
4. Schritt: Ableitung einer KNF oder DNF
5. Schritt: Minimierung

Entwurf eines Schaltwerks

- **Synchroner 3-Bit Binärzähler**
  - Zustandswechsel in allen Flipflops soll gleichzeitig (z.B. bei einer fallenden Taktflanke) erfolgen
  - zunächst Erstellen eines Zustandsdiagramms mit:
    1) allen möglichen Zuständen des 3-Bit Binärzählers
    2) allen möglichen Zustands-übergängen



Entwurf eines Schaltwerks

- Synchroner 3-Bit Binärzähler (Forts.)
  - Es sollen (flankengetriggerte) JK Flip-flops eingesetzt werden
  - Erstellen einer Zustandsübergangstabelle für ein JK Flip-Flop:

bei Q = 0 ist Eingang K irrelevant!

bei Q = 1 ist Eingang J irrelevant!

Übergang Q → Q´		JK Flip-Flop Eingänge	
Q	Q´	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

  - für 3-Bit Binärzähler werden drei JK Flip-Flops mit zentralem Takt benötigt

Entwurf eines Schaltwerks

- Synchroner 3-Bit Binärzähler (Forts.)
  - Zustandsübergangstabelle für 3-Bit Binärzähler:

aktueller Zustand			Folgezustand			Eingänge der JK Flip-Flops						Ausgabe		
Q_2	Q_1	Q_0	Q'_2	Q'_1	Q'_0	J_2	K_2	J_1	K_1	J_0	K_0	Q_2	Q_1	Q_0
0	0	0	0	0	1	0	d	0	d	1	d	0	0	0
0	0	1	0	1	0	0	d	1	d	d	1	0	0	1
0	1	0	0	1	1	0	d	d	0	1	d	0	1	0
0	1	1	1	0	0	1	d	d	1	d	1	0	1	1
1	0	0	1	0	1	d	0	0	d	1	d	1	0	0
1	0	1	1	1	0	d	0	1	d	d	1	1	0	1
1	1	0	1	1	1	d	0	d	0	1	d	1	1	0
1	1	1	0	0	0	d	1	d	1	d	1	1	1	1

der aktuelle Zustand stellt gleichzeitig die Ausgabe des Zählers dar!

Entwurf eines Schaltwerks

- Synchroner 3-Bit Binärzähler
- Es muß nun ein **Schaltnetz** entwickelt werden, das die Ansteuer-signale der JK Flip-Flops aus dem aktuellen Zustand generiert:
  - Ausgangssignale: Q_2, Q_1, Q_0
  - Eingangssignale: J_2, K_2, J_1, K_1, J_0, K_0
  - Minimierung der Ansteuergleichungen für die JK Flip-Flops mit Karnaugh-Veitch-Diagrammen

	$\bar{Q}_0\bar{Q}_1$	$\bar{Q}_0 Q_1$	$Q_0 Q_1$	$Q_0\bar{Q}_1$
$\bar{Q}_2$	0	0	1	0
Q_2	d	d	d	d

Resultat: $J_2 = Q_1 \cdot Q_0$

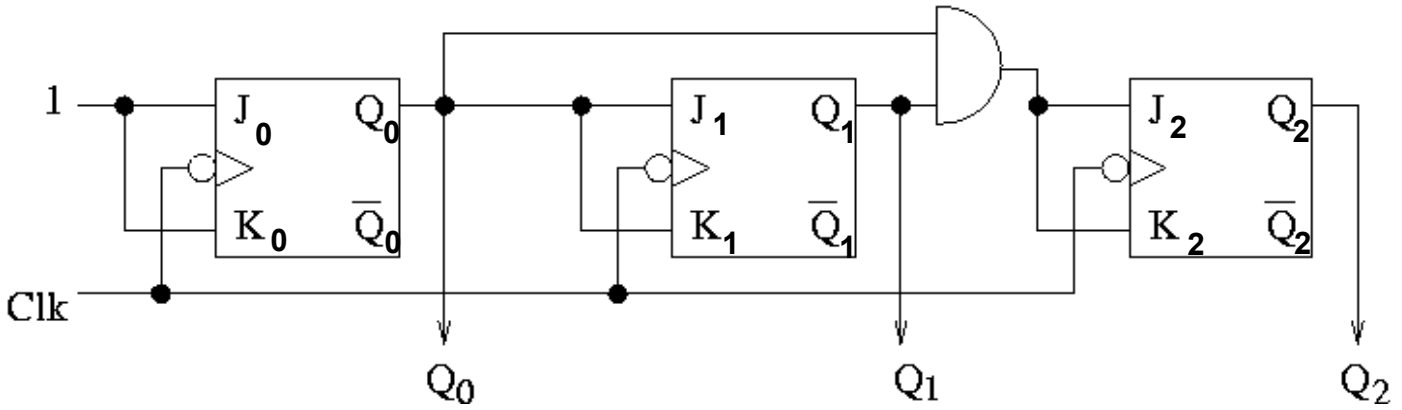
	$\bar{Q}_0\bar{Q}_1$	$\bar{Q}_0 Q_1$	$Q_0 Q_1$	$Q_0\bar{Q}_1$
$\bar{Q}_2$	d	d	d	d
Q_2	0	0	1	0

$K_2 = Q_1 \cdot Q_0$

Entwurf eines Schaltwerks

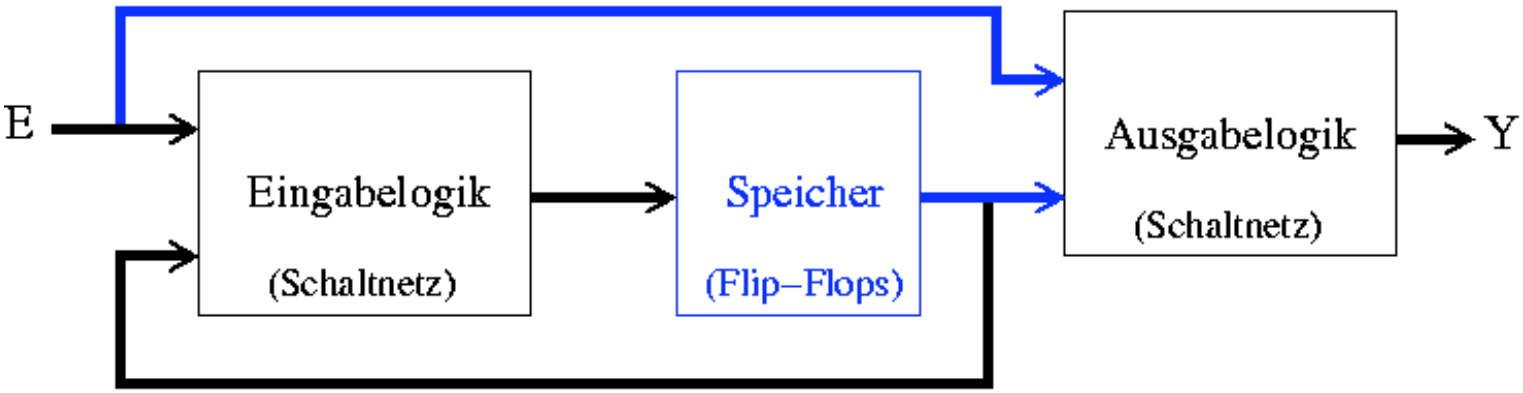
- Synchroner 3-Bit Binärzähler (Forts.)
  - für J_1, K_1, J_0 und K_0 können die Ansteuergleichungen auch direkt der Zustandsübergangstabelle des Zählers entnommen werden
  - insgesamt ergibt sich folgende Schaltfunktion:

    $J_2 = Q_1 \cdot Q_0$ $\quad K_2 = Q_1 \cdot Q_0$
    $J_1 = Q_0$ $\quad K_1 = Q_0$
    $J_0 = 1,$ $\quad K_0 = 1$
  - Realisierung des synchronen 3-Bit Binärzählers:



Entwurf eines Schaltwerks

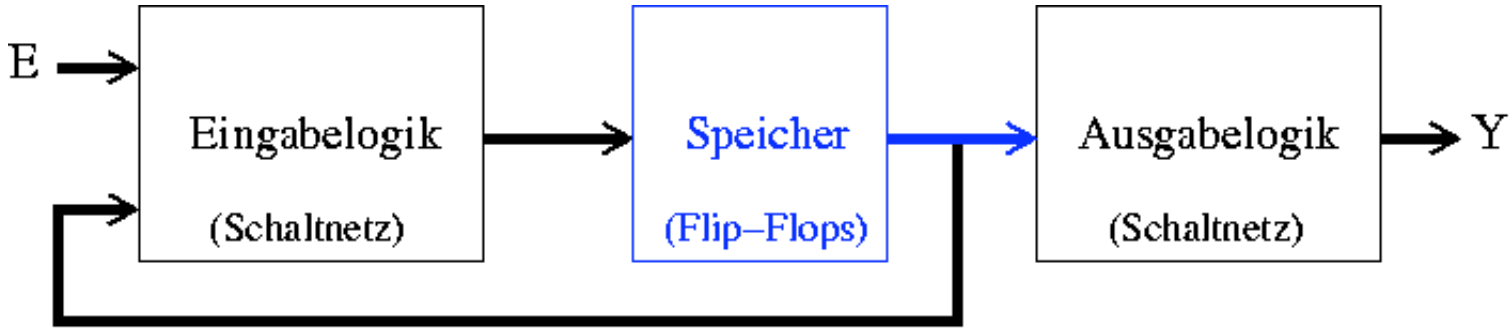
- **Mealy-Automat:**
  - nach G. Mealy (IBM)
  - allgemeiner Aufbau:



  - Eingabelogik wie beim Moore-Automaten
  - Ausgabe Y hängt jedoch durch die Ausgabelogik vom aktuellen Zustand und vom Eingabesignal E ab

Entwurf eines Schaltwerks

- **Moore-Automat:**
  - nach E. Moore (Bell Labs)
  - allgemeiner Aufbau:



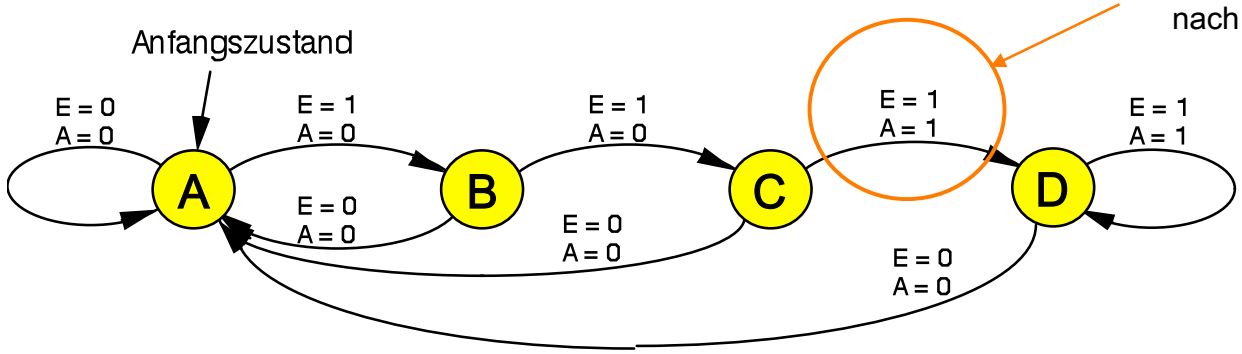
  - Eingabelogik bestimmt Zustandsübergänge, die von den Eingabesignalen E und vom aktuellen Zustand abhängen
  - Ausgabelogik bestimmt Ausgabe Y, die nur vom aktuellen Zustand abhängt

Entwurf eines Schaltwerks

- Ein **Zustandsdiagramm** ist Darstellung einer Aufgabenstellung als gerichteter, zyklischer Graph, wobei die Knoten den Zuständen und die Kanten den Zustandübergängen entsprechen.
- Zustandsdiagramm für einen **Mealy**-Automaten
  - **Knoten:** Markierung **S** gibt nur die Bezeichnung des Zustands an
  - **Kanten:** Markierung **E/A** mit der für den jeweiligen Zustands-übergang erforderlichen Eingabe **E** = E_1 E_2 ... E_m und der resultierenden Ausgabe **A** = A_1A_2 ... A_n.
- Zustandsdiagramm für einen **Moore**-Automaten
  - **Knoten:** Markierung **S/A** gibt Bezeichnung des Zustands **S** und die zugehörige Ausgabe **A** = A_1A_2 ... A_n an
  - **Kanten:** Markierung mit der für den jeweiligen Zustandsübergang erforderlichen Eingabe **E** = E_1 E_2 ... E_m.

Mealy und Moore Automaten

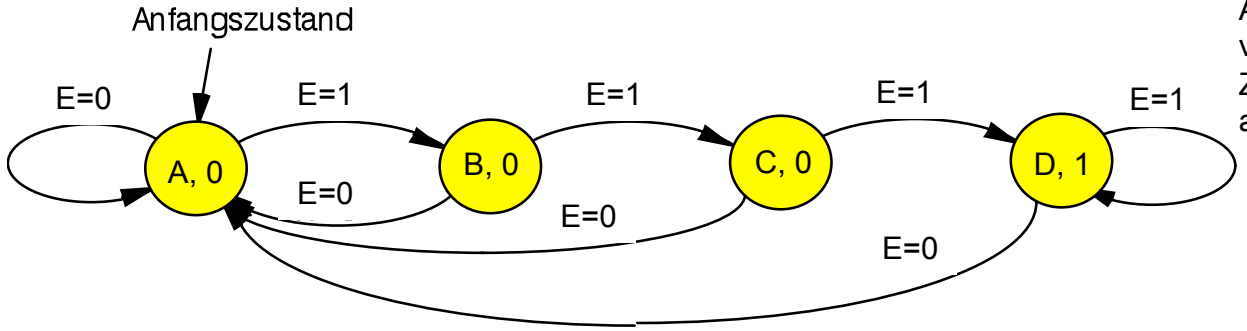
Mealy:



bereits in Zustand C
wird bei E=1 → A=1
erzeugt UND der Übergang
nach D bewirkt.

→ asynchron

Moore:



Ausgabe erfolgt nur abhängig vom Zustand. Lediglich die Zustandsübergänge sind Eingabe-abhängig.

Mealy und Moore Automaten

Mealy

jetziger Zustand	nächster Zustand		Ausgaben	
	E = 0	E = 1	E = 0	E = 1
A	A	B	0	0
B	A	C	0	0
C	A	D	0	1
D	A	D	0	1

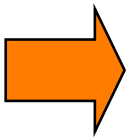
Moore

jetziger Zustand	nächster Zustand		Ausg.
	E = 0	E = 1	A
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

Entwurf eines Moore Automaten

Zustandstabelle

jetziger Zustand	nächster Zustand		Ausg.
	E = 0	E = 1	A
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1



Zuweisung der Flip-Flop Zustände

jetziger Zustand	nächster Zustand		Ausg.
	E = 0	E = 1	A
FG	FG	FG	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

Entwurf eines Moore Automaten

Mealy

jetziger Zustand	nächster Zustand		Ausgaben	
	E = 0	E = 1	E = 0	E = 1
FG	FG	FG	A	A
00	00	01	0	0
01	00	10	0	0
10	00	11	0	1
11	00	11	0	1

Moore

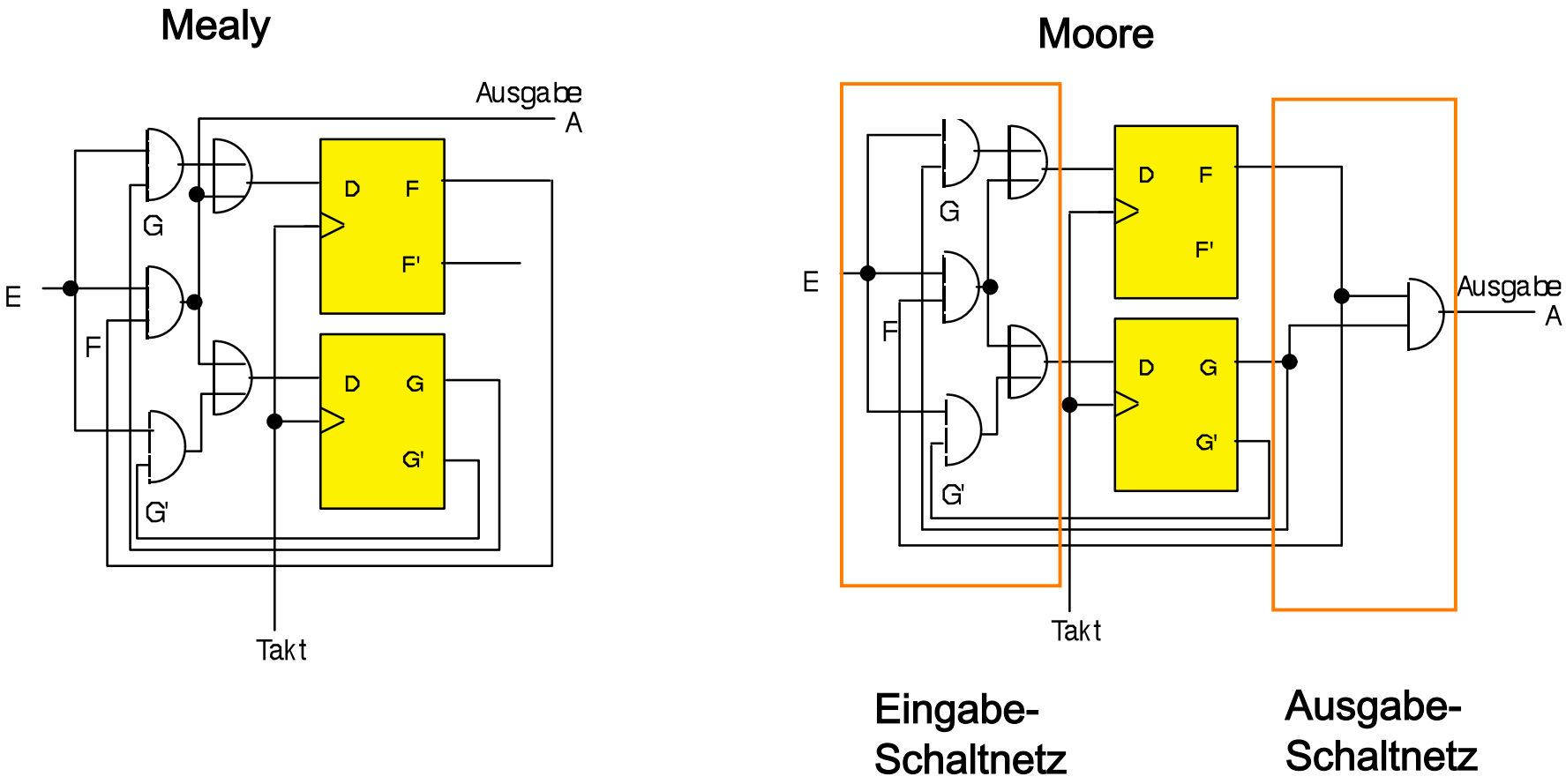
jetziger Zustand	nächster Zustand		Ausg.
	E = 0	E = 1	A
FG	FG	FG	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

A = EF abhängig vom Zustand UND E

identisch, d.h. keine Änderungen am Eingabeschaltnetz

A = FG nur vom Zustand abhängig

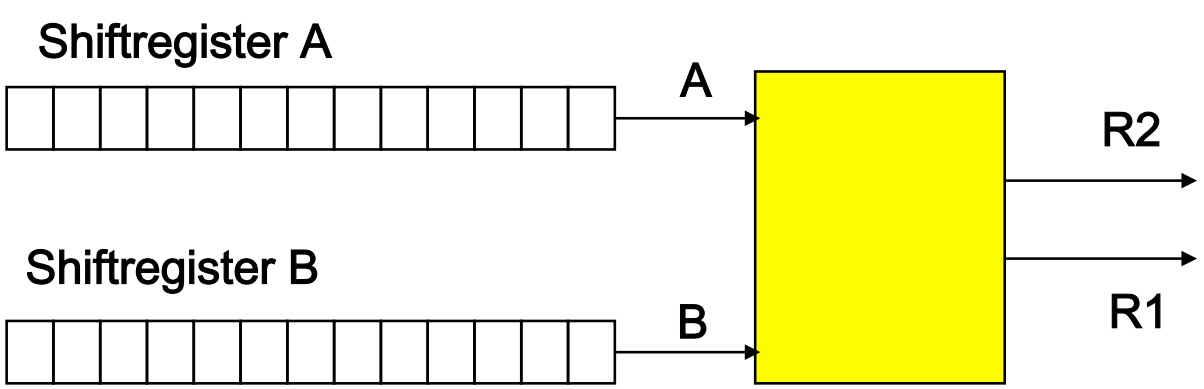
Entwurf eines Moore Automaten



Entwurf eines Schaltwerks

- **Vorgehensweise:**
  1. Erstellen eines Zustandsdiagramms
  2. Erstellen einer Zustandstabelle
  3. Auswahl einer binären Zustandskodierung und Generierung einer binären Zustandstabelle
  4. Auswahl eines Flip-Flop Typs und Ermittlung der für jeden Zustandsübergang benötigten Flip-Flop Ansteuerungen
  5. Ermittlung der Ausgabegleichungen
  6. Minimierung der Ansteuer- und Ausgabegleichungen
  7. Realisierung des Schaltwerks

Entwurf eines sequentiellen Binärzahlen-Vergleichers

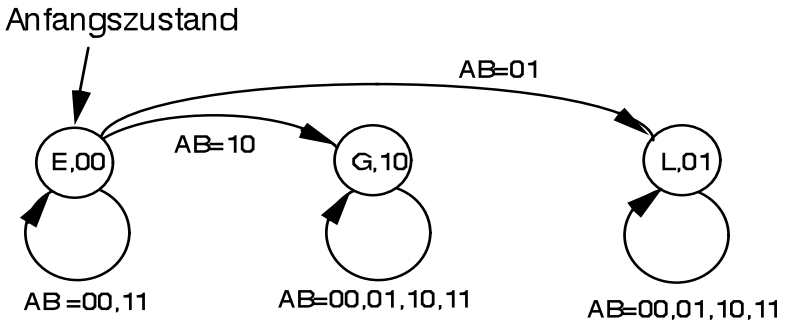


Ausgabe:

	R1	R2
E: A = B	0	0
G: A > B	1	0
L: A < B	0	1

1. Schritt: Aufgabenspezifikation, Erstellen eines Zustandsdiagramms

Zustands-Diagramm:



Entwurf eines sequentiellen Binärzahlen-Vergleichers

2. Schritt: Erstellen einer Zustandstabelle

jetziger Zustand	nächster Zustand				Ausgaben	
	AB=00	AB=01	AB=10	AB=11	R1	R2
E	E	L	G	E	0	0
G	G	G	G	G	1	0
L	L	L	L	L	0	1

3. Schritt: Auswahl einer binären Zustandskodierung und Generierung einer binären Zustandstabelle

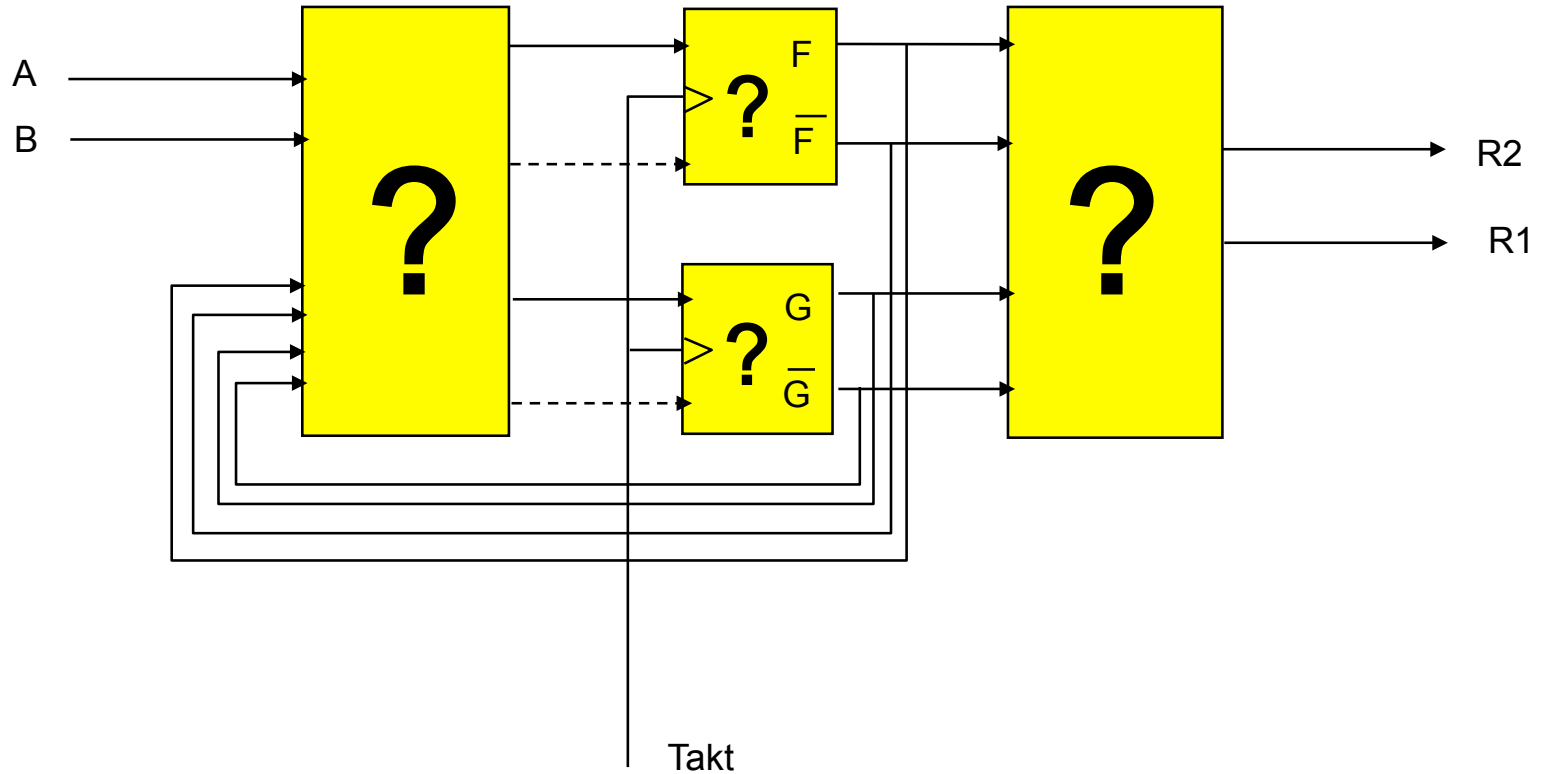
$E = \overline{F}\,\overline{G}$

$G = \overline{F}\,G$

$L = F\,\overline{G}$

	jetziger Zustand	nächster Zustand				Ausgaben	
		AB=00	AB=01	AB=10	AB=11	R1	R2
E	00	00	10	01	00	0	0
G	01	01	01	01	01	1	0
L	10	10	10	10	10	0	1

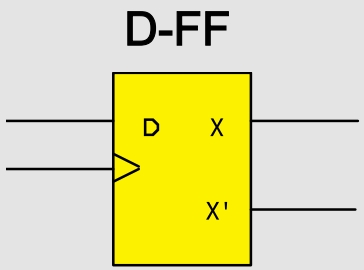
Entwurf eines sequentiellen Binärzahlen-Vergleichers



4. Schritt: Auswahl eines Flip-Flop Typs und Ermittlung der für jeden Zustandsübergang benötigten Flip-Flop Ansteuerungen

Entwurf eines sequentiellen Binärzahlen-Vergleichers

4. Schritt: Auswahl eines Flip-Flop Typs und Ermittlung der für jeden Zustandsübergang benötigten Flip-Flop Ansteuerungen

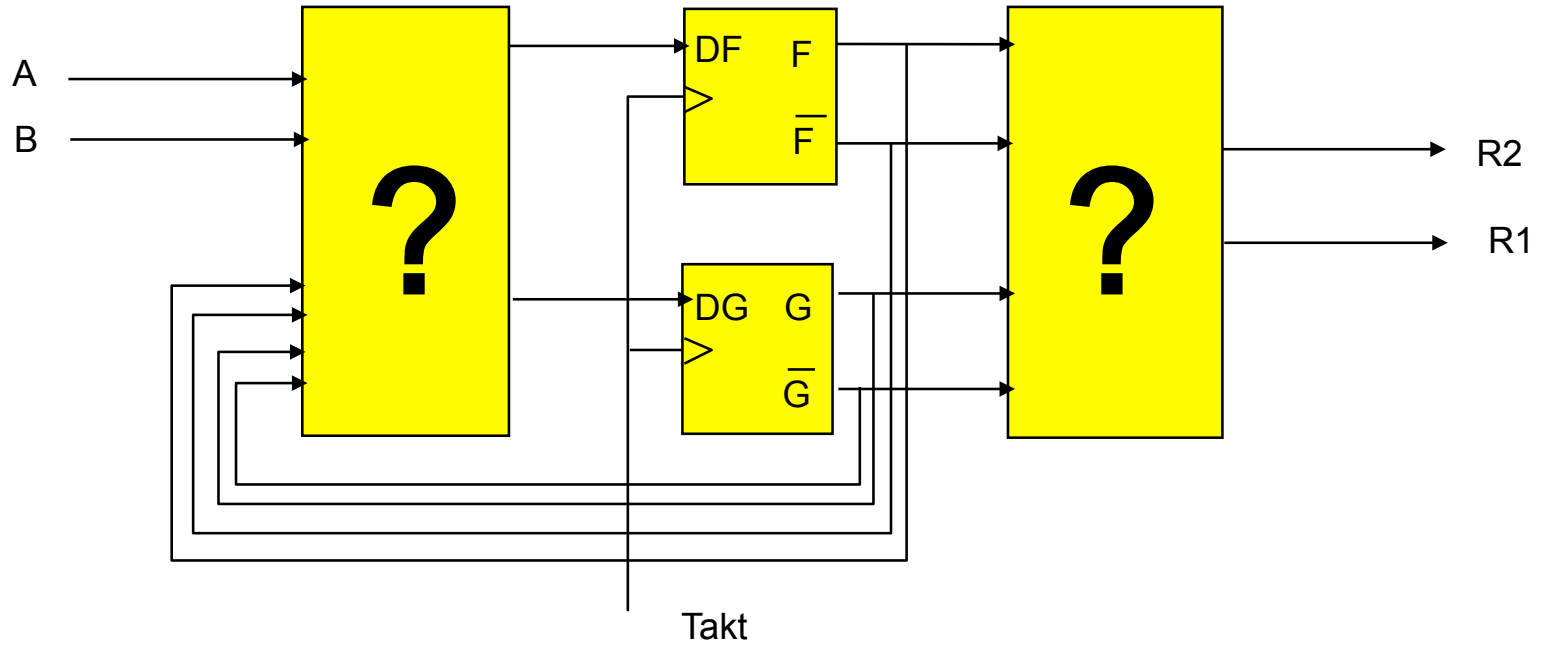


D	X(t + 1)
0	0
1	1

X(t)	X(t + 1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Entwurf eines sequentiellen Binärzahlen-Vergleichers

Zustands-Fortschaltung



Equal:
Ausgabe 00

$E = \overline{F}\,\overline{G}$

Eingaben d. komb. Schaltnetzes						Ausgaben d. komb. Schaltnetzes			
FF-Zust. (t)		Eingabe		FF-Zust. (t+1)		FF- Eingaben		Ausgaben	
F	G	A	B	F	G	DF	DG	R1	R2
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	0	0	0	0	0

Entwurf eines sequentiellen Binärzahlen-Vergleichers

Zustands-Fortschaltung

Greater:
Ausgabe 01

$G = \overline{F}\, G$

Eingaben d. komb. Schaltnetzes						Ausgaben d. komb. Schaltnetzes			
FF-Zust. (t)		Eingabe		FF-Zust. (t+1)		FF- Eingaben		Ausgaben	
F	G	A	B	F	G	DF	DG	R1	R2
0	1	0	0	0	1	0	1	1	0
0	1	0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1	1	0
0	1	1	1	0	1	0	1	1	0

Less:
Ausgabe 10

$L = F\, \overline{G}$

Eingaben d. komb. Schaltnetzes						Ausgaben d. komb. Schaltnetzes			
FF-Zust. (t)		Eingabe		FF-Zust. (t+1)		FF- Eingaben		Ausgaben	
F	G	A	B	F	G	DF	DG	R1	R2
1	0	0	0	1	0	1	0	0	1
1	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0	0	1

FG tritt nicht auf

Eingaben d. komb. Schaltnetzes						Ausgaben d. komb. Schaltnetzes			
FF-Zust. (t)		Eingabe		FF-Zust. (t+1)		FF- Eingaben		Ausgaben	
F	G	A	B	F	G	DF	DG	R1	R2
1	1	0	0	d	d	d	d	d	d
1	1	0	1	d	d	d	d	d	d
1	1	1	0	d	d	d	d	d	d
1	1	1	1	d	d	d	d	d	d

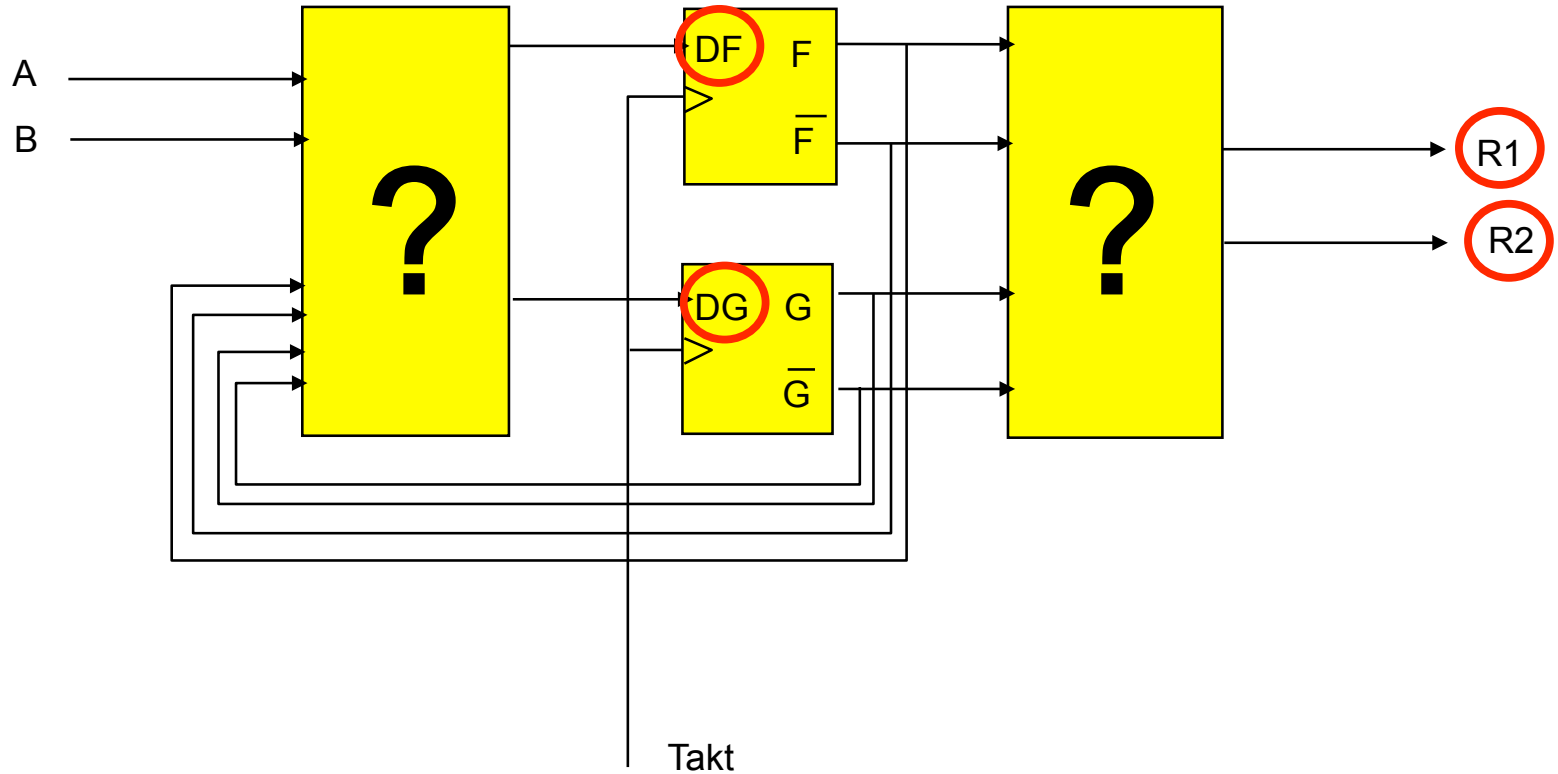
Entwurf eines sequentiellen Binärzahlen-Vergleichers

Eing. d. komb. Schaltkr.				Nächster		Ausg. d. komb. Schaltkr.			
Jetziger FF-Zustand		Eingaben		FF-Zustand		Flip-Flop-Eingaben		Ausgaben	
F	G	A	B	F	G	DF	DG	R1	R2
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	1	0	1	1	0
0	1	0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1	1	0
0	1	1	1	0	1	0	1	1	0
1	0	0	0	1	0	1	0	0	1
1	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0	0	1
1	1	0	0	d	d	d	d	d	d
1	1	0	1	d	d	d	d	d	d
1	1	1	0	d	d	d	d	d	d
1	1	1	1	d	d	d	d	d	d

$$DF = F + \overline{F}\,\overline{G}\,\overline{A}B = F + \overline{G}\,\overline{A}B$$

$$DG = G + \overline{F}\,\overline{G}A\overline{B} = G + \overline{F}A\overline{B}$$

Entwurf eines sequentiellen Binärzahlen-Vergleichers



5. Schritt: Ermittlung der Ausgabegleichungen

Entwurf eines sequentiellen Binärzahlen-Vergleichers

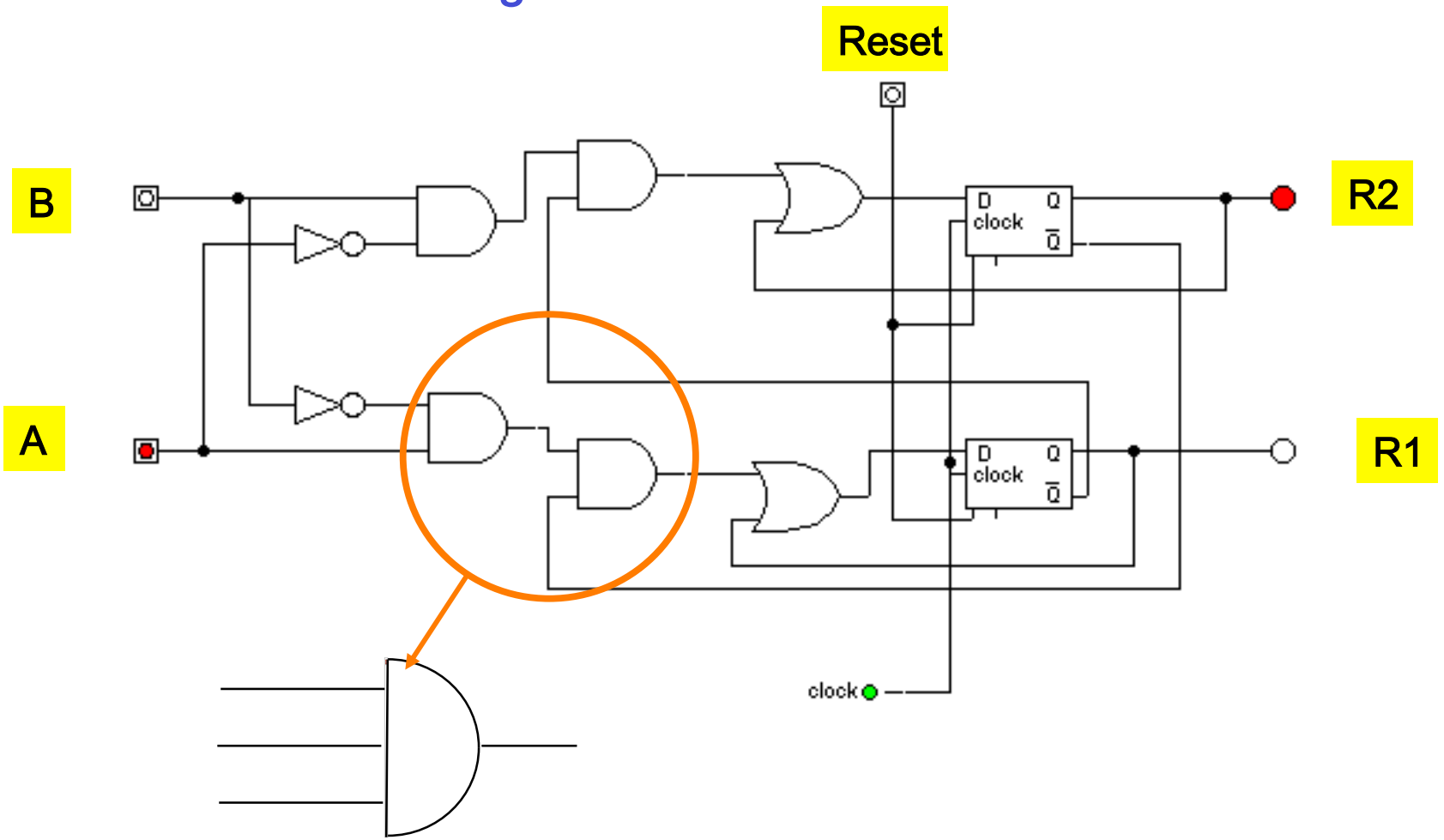
Eing. d. komb. Schaltkr.				Nächster		Ausg. d. komb. Schaltkr.			
Jetziger FF-Zustand		Eingaben		FF-Zustand		Flip-Flop-Eingaben		Ausgaben	
F	G	A	B	F	G	DF	DG	R1	R2
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	1	0	1	1	0
0	1	0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1	1	0
0	1	1	1	0	1	0	1	1	0
1	0	0	0	1	0	1	0	0	1
1	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0	0	1
1	1	0	0	d	d	d	d	d	d
1	1	0	1	d	d	d	d	d	d
1	1	1	0	d	d	d	d	d	d
1	1	1	1	d	d	d	d	d	d

R1 = G

R2 = F

Entwurf eines sequentiellen Binärzahlen-Vergleichers

6. Schritt: Realisierung des Schaltwerks



Vergleich Moore- und Mealy-Automat

- sowohl Moore-Automat als auch Mealy-Automat zum Entwurf beliebiger Schaltwerke geeignet
- **Vorteile Moore-Automat:**
  - taktsynchrone Ausgabe A, asynchron auftretende Störungen der Eingabesignale wirken sich nicht auf A aus
  - geringerer Schaltungsaufwand für Ausgabelogik, wenn Ausgabe A eigentlich nur vom Zustand abhängt
- **Vorteile Mealy-Automat:**
  - schnellere Reaktion auf Veränderung der Eingabesignale E
  - Realisierung ist mit einer kleineren Anzahl an Zuständen möglich, wenn mehrere Zustandsübergänge zu einem Zustand verschiedene Ausgaben erfordern

???

1. Hat der Typ des gewählten Flip-Flops Auswirkungen auf den Entwurf ?

2. Können Automaten minimiert, d.h. die Anzahl der Zustände verringert werden?

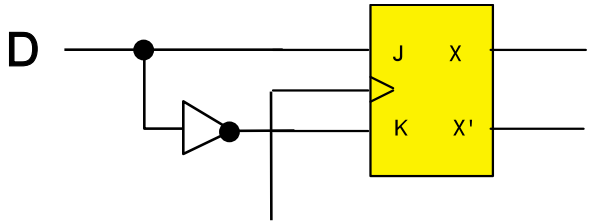
3. Ist das resultierende Schaltwerk dann einfacher?

4. Welche Auswirkungen hat die Zuordnung von Automatenzuständen zu den Flip-Flop Zuständen?

   → Assignment Problem

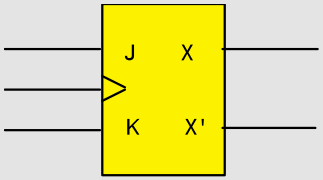
Beisp. Realisierung des Detektors mit JK-FF

Trivialansatz:



don‘t cares ermöglichen Flexibilität im Entwurf

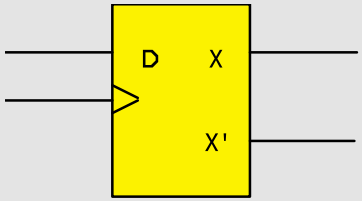
JK Flip-Flop



J	K	X(t + 1)
0	0	X(t)
0	1	0
1	0	1
1	1	X'(t)

X(t)	X(t + 1)	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

D Flip-Flop



D	X(t + 1)
0	0
1	1

X(t)	X(t + 1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Beisp. Realisierung des Detektors mit JK-FF

D-FF

jetziger Zustand		Eingaben	nächster Zustand		Flip-Flop Eingaben		Ausgaben
F	G	E	F	G	DF	DG	A
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1

JK-FF

Eing. d. kombin. Schaltkr.			nächster Zustand		Ausgänge des kombinatorischen Schaltkreises				
Jetziger Zustand		Eingabe			Flip-Flop-Eingaben				Ausgabe
F	G	E	F	G	JF	KF	JG	KG	A
0	0	0	0	0	0	d	0	d	0
0	0	1	0	1	0	d	1	d	0
0	1	0	0	0	0	d	d	1	0
0	1	1	1	0	1	d	d	1	0
1	0	0	0	0	d	1	0	d	0
1	0	1	1	1	d	0	1	d	1
1	1	0	0	0	d	1	d	1	0
1	1	1	1	1	d	0	d	0	1

Beisp. Realisierung des Detektors mit JK-FF

Eing. d. kombin. Schaltkr.			nächster Zustand		Ausgänge des kombinatorischen Schaltkreises				
Jetziger Zustand		Eingabe			Flip-Flop-Eingaben				Ausgabe
F	G	E	F	G	JF	KF	JG	KG	A
0	0	0	0	0	0	d	0	d	0
0	0	1	0	1	0	d	1	d	0
0	1	0	0	0	0	d	d	1	0
0	1	1	1	0	1	d	d	1	0
1	0	0	0	0	d	1	0	d	0
1	0	1	1	1	d	0	1	d	1
1	1	0	0	0	d	1	d	1	0
1	1	1	1	1	d	0	d	0	1

$JF = GE$

E'	F'G'	F'G	FG	FG'
E'	0	0	d	d
E	0	1	d	d

$JG = E$

E'	F'G'	F'G	FG	FG'
E'	0	d	d	0
E	1	d	d	1

keine Änderung für den Ausgang:

$KF = \overline{E}$

E'	F'G'	F'G	FG	FG'
E'	d	d	1	1
E	d	d	0	0

$KG = \overline{E} + \overline{F}$

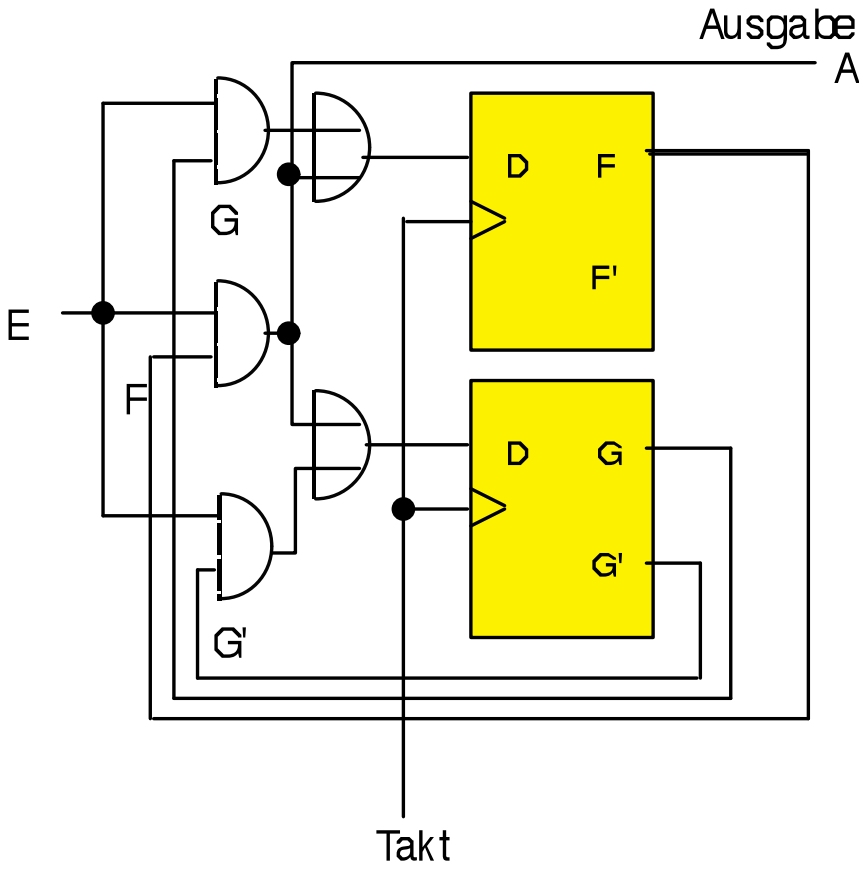
E'	F'G'	F'G	FG	FG'
E'	d	1	1	d
E	d	1	0	d

$A = EF$

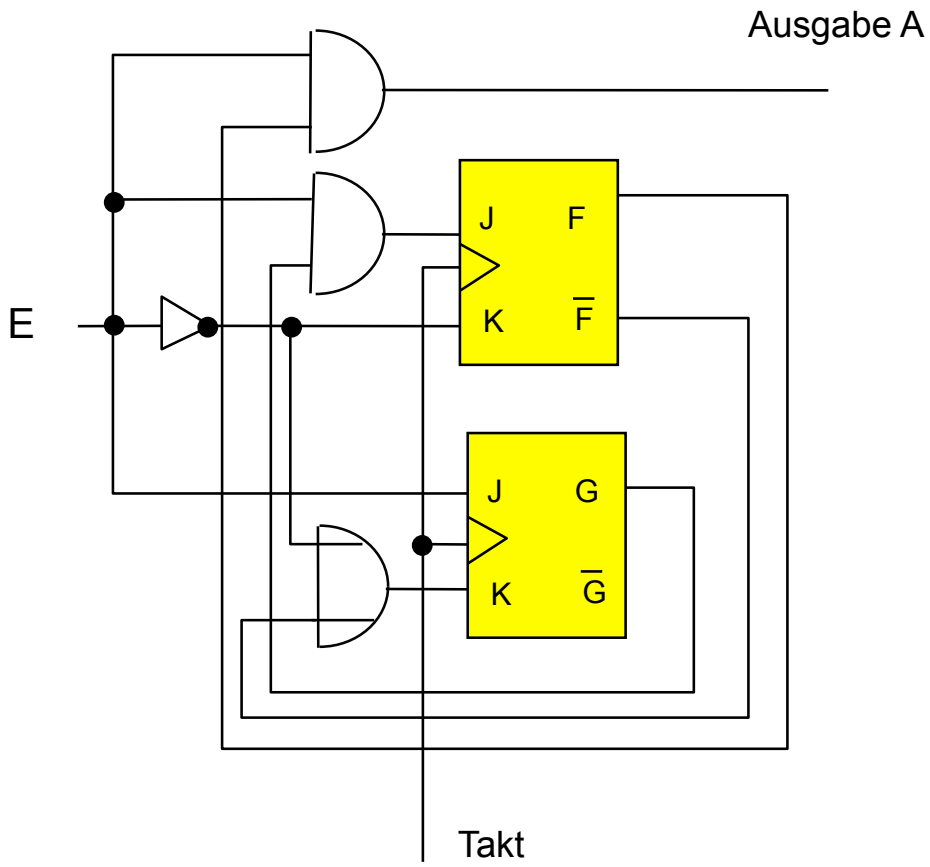
E'	F'G'	F'G	FG	FG'
E'	0	0	0	0
E	0	0	1	1

Beisp. Realisierung des Detektors mit JK-FF

mit D-Flip-Flops



mit JK-Flip-Flops



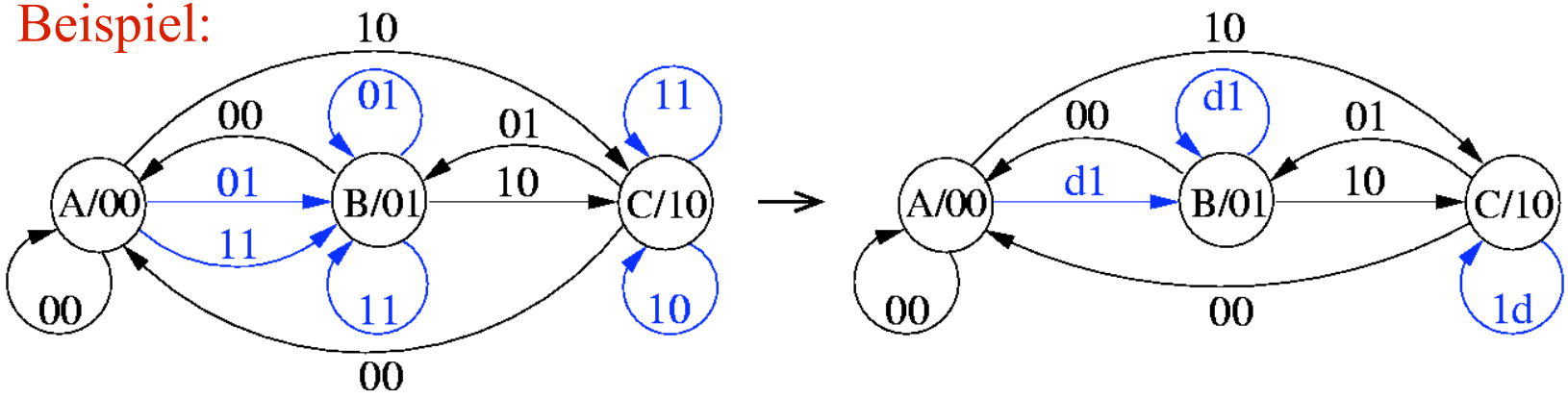
Anmerkungen

- jeder beliebige getaktete Flip-Flop Typ darf verwendet werden
  - zunächst ist Flip-Flop Zustands-Übergangstabelle aufzustellen
  - in Ansteuertabelle müssen die entsprechenden Ansteuersignale für die benötigten Zustands-Übergänge eingetragen werden
  - Schaltungsaufwand für Eingabelogik abhängig vom Flip-Flop Typ

Anmerkungen

- zur Vereinfachung des Zustandsdiagramms dürfen Zustands-übergänge, die von einer Eingangsvariablen unabhängig sind, auch mit d („don‘t care“) beschriftet werden:

Beispiel:



???

1. Hat der Typ des gewählten Flip-Flops Auswirkungen auf den Entwurf ?

2. Können Automaten minimiert, d.h. die Anzahl der Zustände verringert werden?

   Ist das resultierende Schaltwerk dann einfacher?

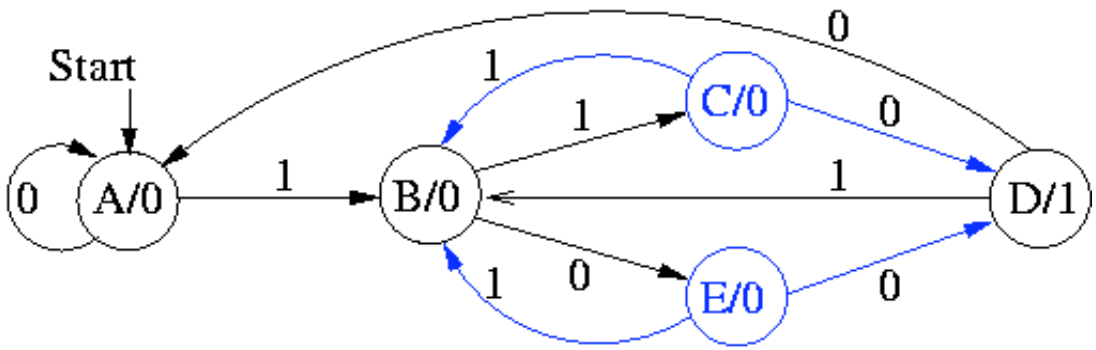
1. Welche Auswirkungen hat die Zuordnung von Automatenzuständen zu den Flip-Flop Zuständen?

   → Assignment Problem

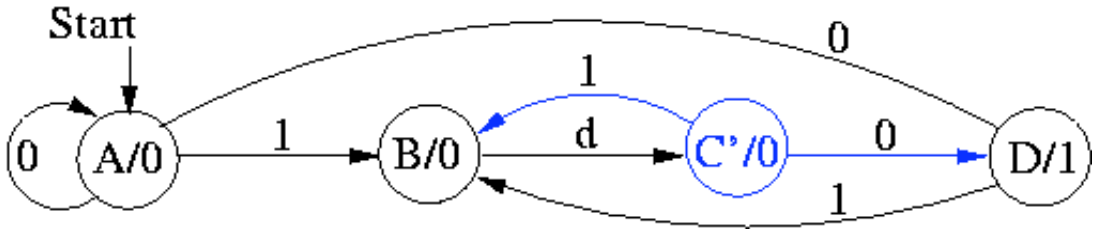
Reduktion von Zuständen

- im Moore-Automat können Zustände mit gleichen Ausgaben und gleichen Folgezuständen zusammengefasst werden

Beispiel:
Automat zur Erkennung von 110 oder 100



⇒ Einsparung eines Flip-Flops !



- im Mealy-Automat sind Zustände mit gleichen Folgezuständen und gleichen Ausgaben bei den Übergängen zusammenfassbar

???

1. Hat der Typ des gewählten Flip-Flops Auswirkungen auf den Entwurf ?

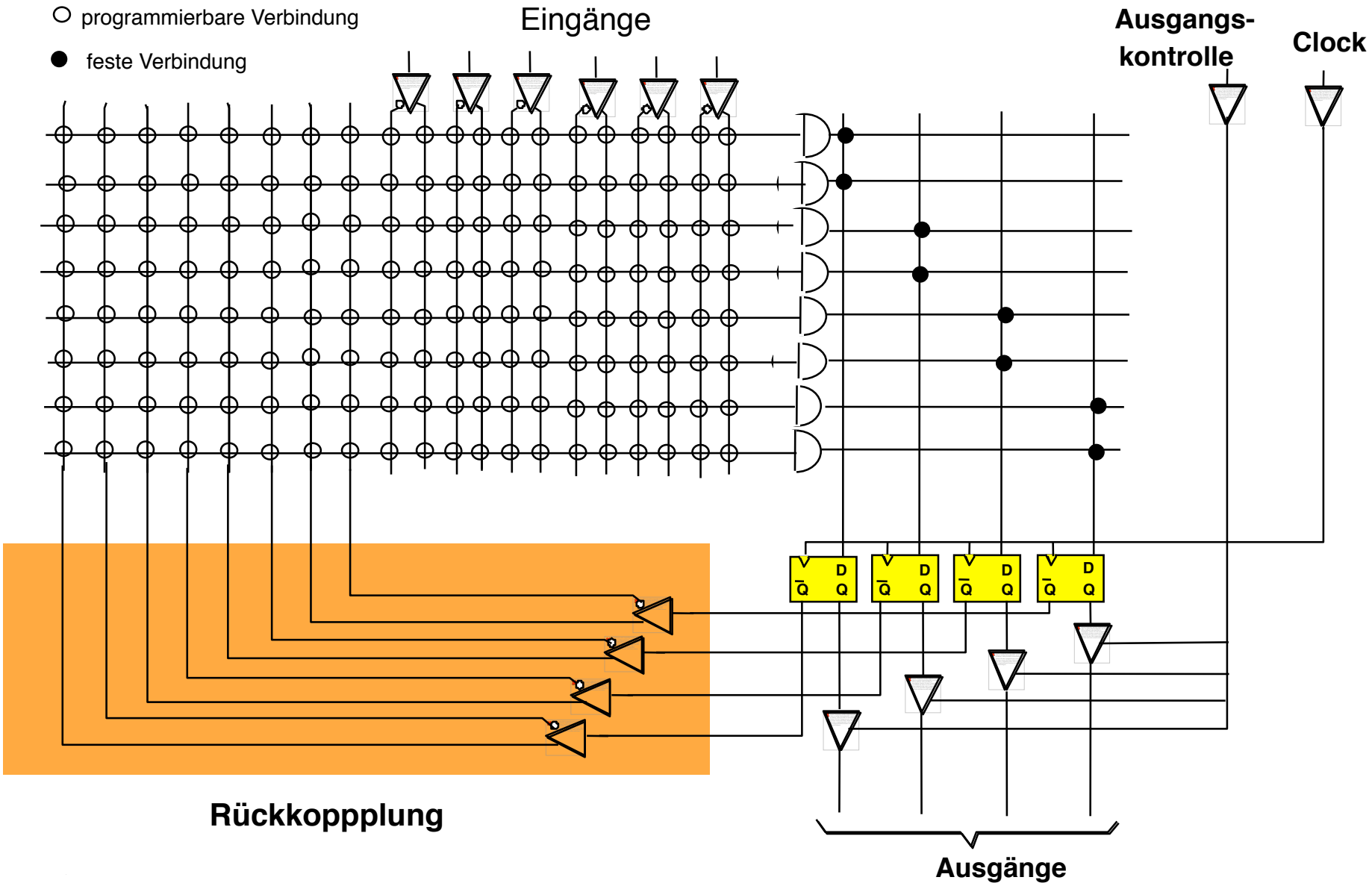
2. Können Automaten minimiert, d.h. die Anzahl der Zustände verringert werden?

   Ist das resultierende Schaltwerk dann einfacher?

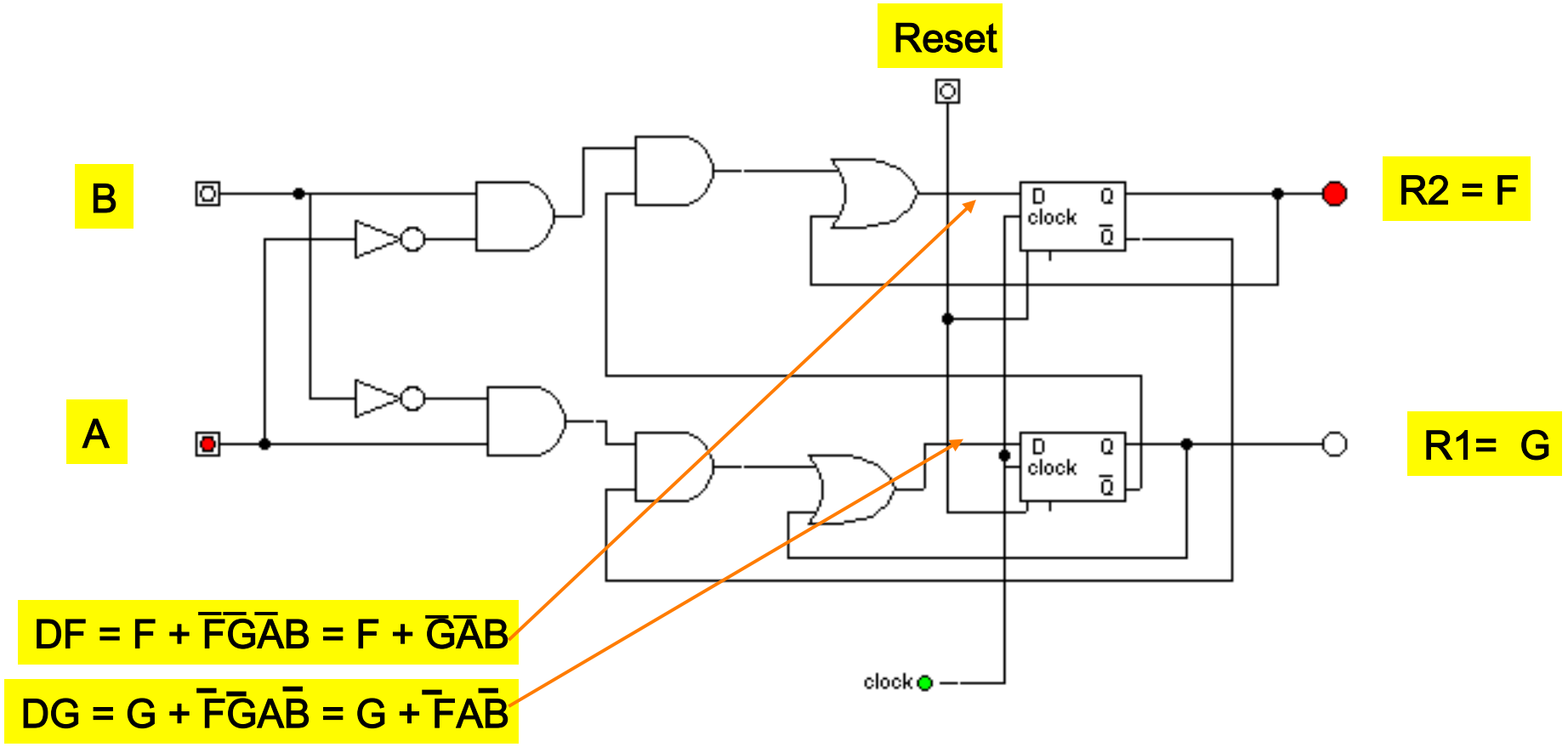
1. Welche Auswirkungen hat die Zuordnung von Automatenzuständen zu den Flip-Flop Zuständen?

   → Assignment Problem

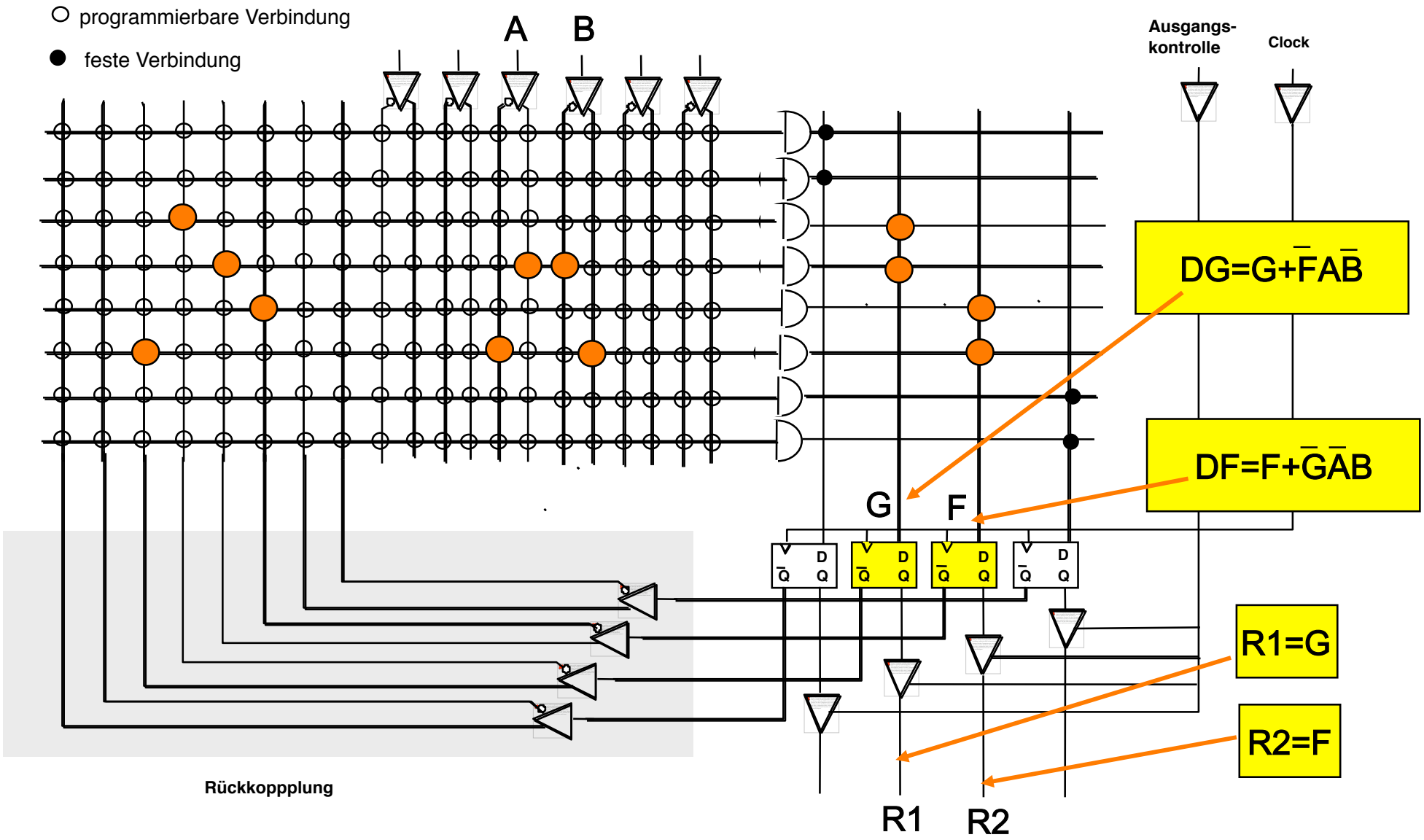
Realisierung von Schaltwerken



Realisierung des Binärzahl-Vergleichers

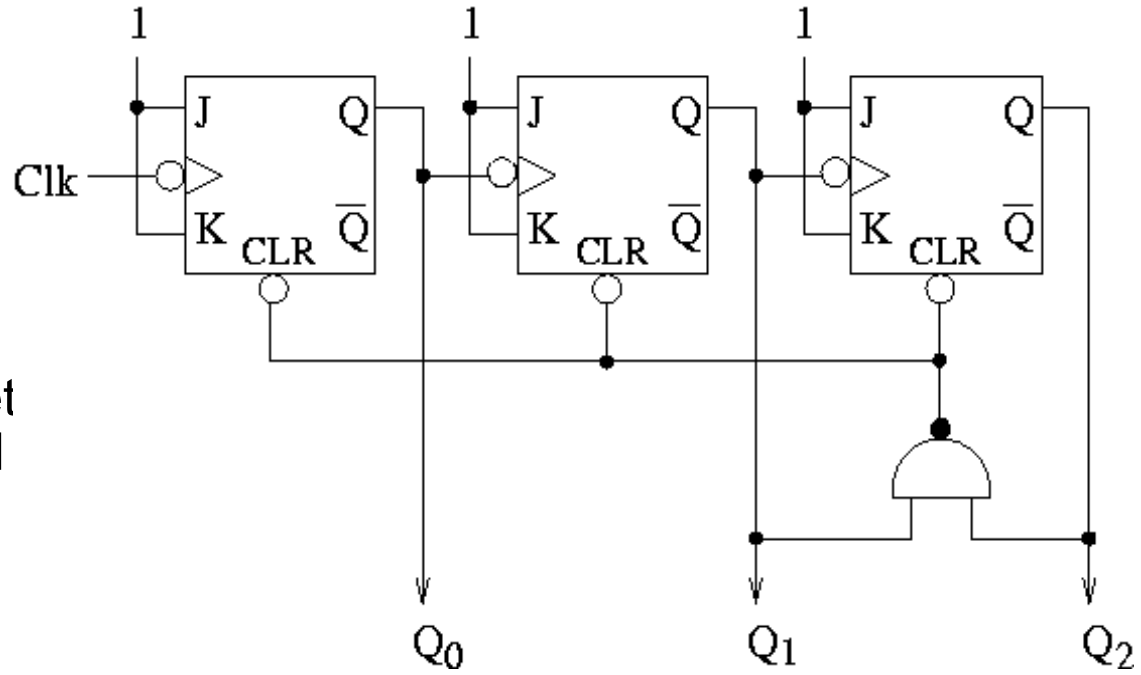


Realisierung des Binärzahl-Vergleichers



Asynchrone Schaltwerke

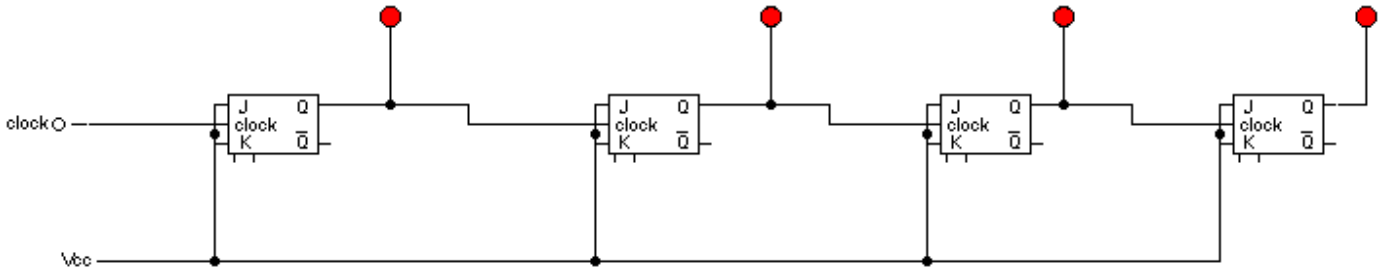
- im Gegensatz zum Entwurf synchroner Schaltwerke gibt es für asynchrone Schaltwerke keinen systematischen Ansatz
- oft werden durch Intuition trickreiche Schaltungen entwickelt, wobei z.B. die asynchronen CLR-Eingänge benutzt werden
- Beispiel:
  asynchroner
  Modulo-6 Zähler



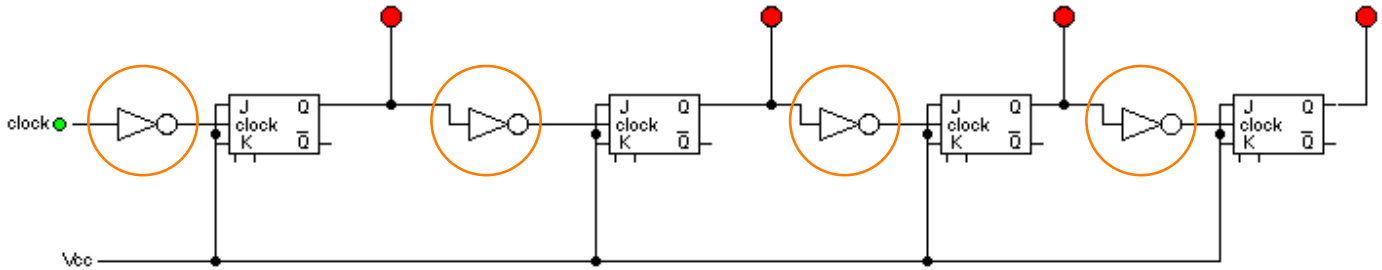
⇒ asynchroner Reset
bei $Q_1 = 1$ und $Q_2 = 1$

Asynchrone Zähler

asynchroner (vorwärts-) Zähler



asynchroner (rückwärts-) Zähler



Bei asynchronen Zählern ist es wichtig, auf welche Flanke getriggert wird !

Asynchrone Zähler

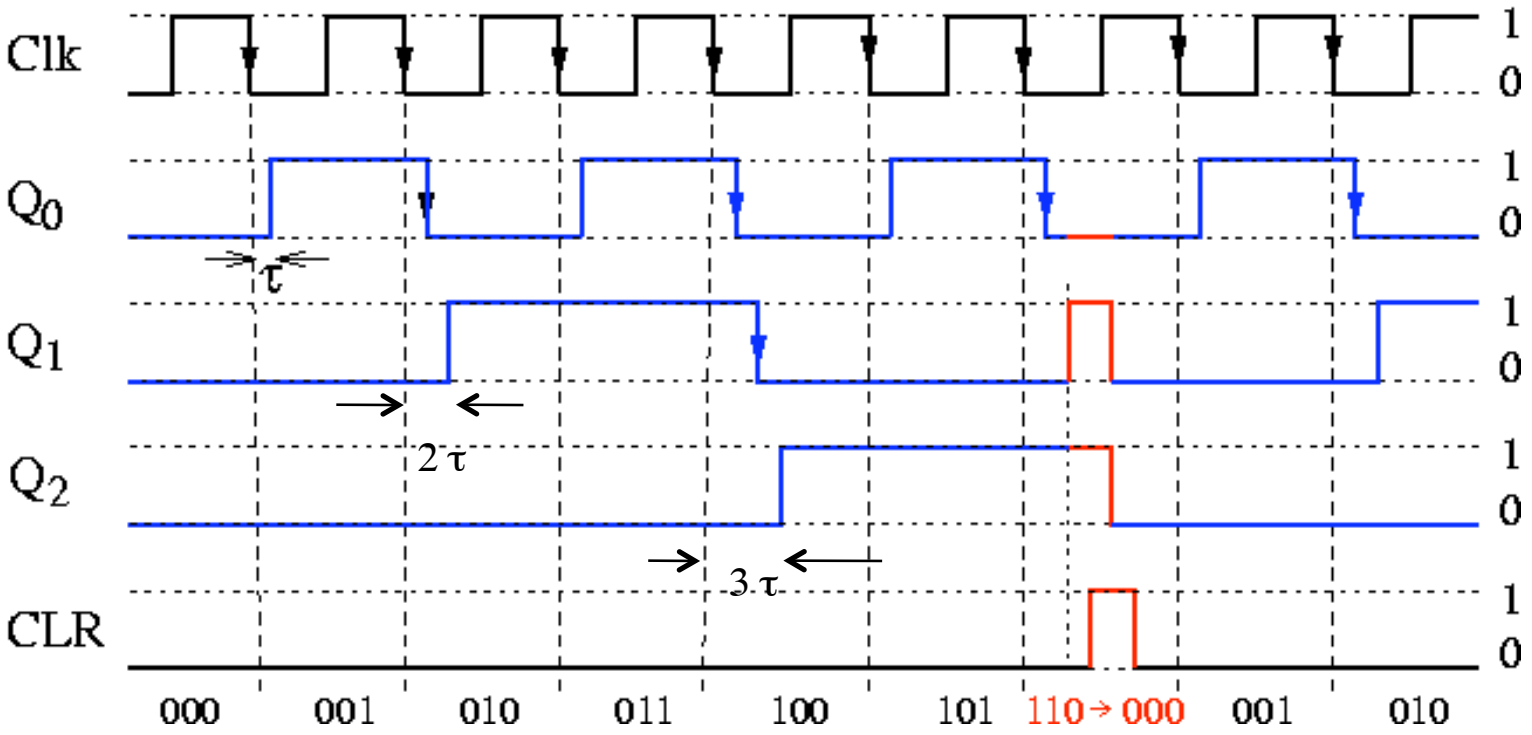
↓ Trigger durch negative Flanke

← Propagieren zum nächsten Takteingang

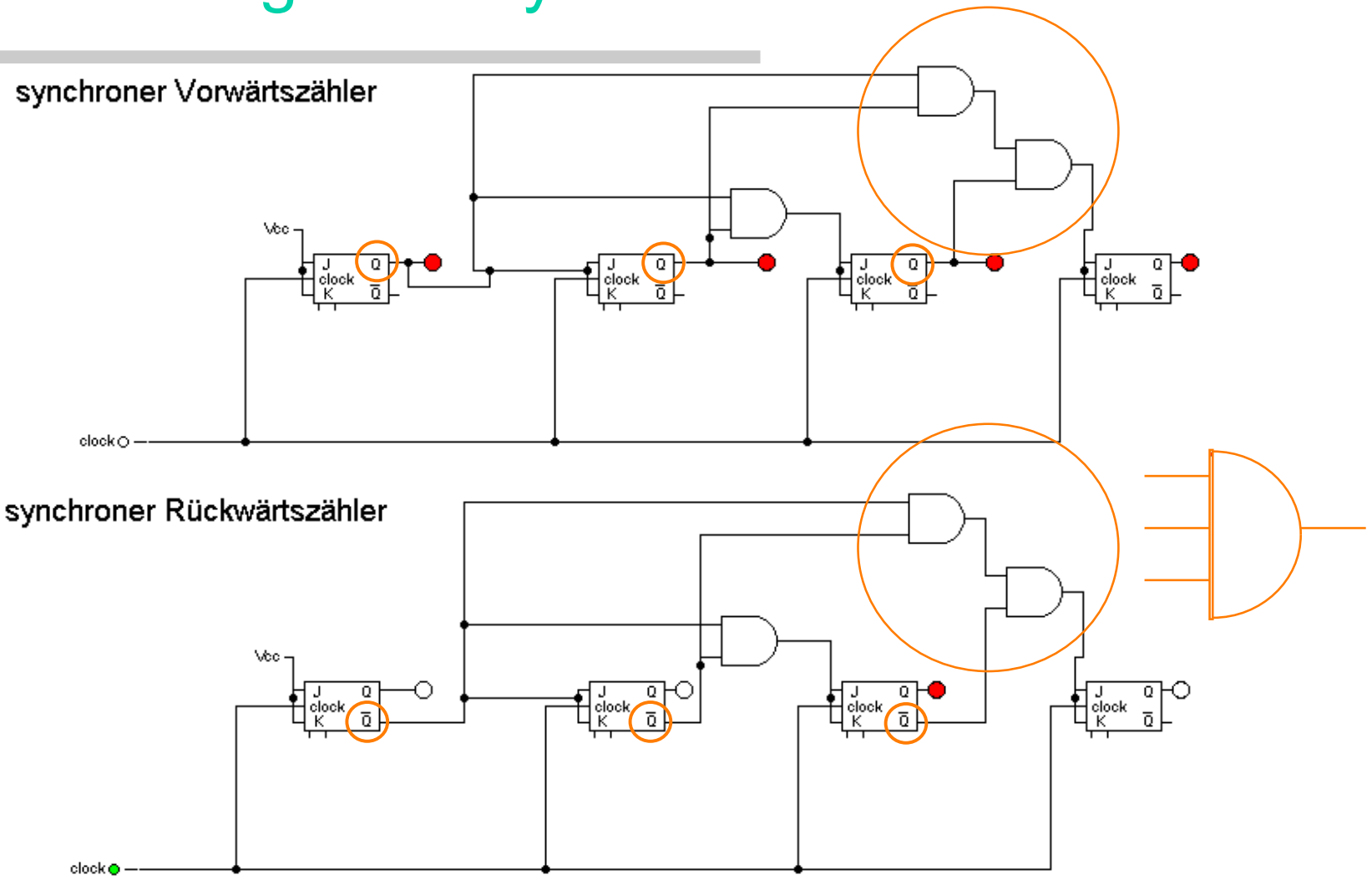
X2	X1	X0	
0	0	0	Initialer Zustand
0	0	1	X0 komplementieren (von 0 auf 1): triggert nicht
	←	↓	X0 komplementieren (von 1 auf 0):
0	1	0	triggert X1 (von 0 auf 1)
0	1	1	X0 komplementieren (von 0 auf 1): triggert nicht
←	←	↓	X0 komplementieren (von 1 auf 0):
1	0	0	triggert X1 (von 1 auf 0), triggert X2
1	0	1	X0 komplementieren (von 0 auf 1): triggert nicht
	←	↓	X0 komplementieren (von 1 auf 0):
1	1	0	triggert X1 (von 0 auf 1)
1	1	1	X0 komplementieren (von 0 auf 1): triggert nicht
←	←	↓	X0 komplementieren (von 1 auf 0):
0	0	0	triggert X1 (von 1 auf 0), triggert X2

Asynchroner Zähler

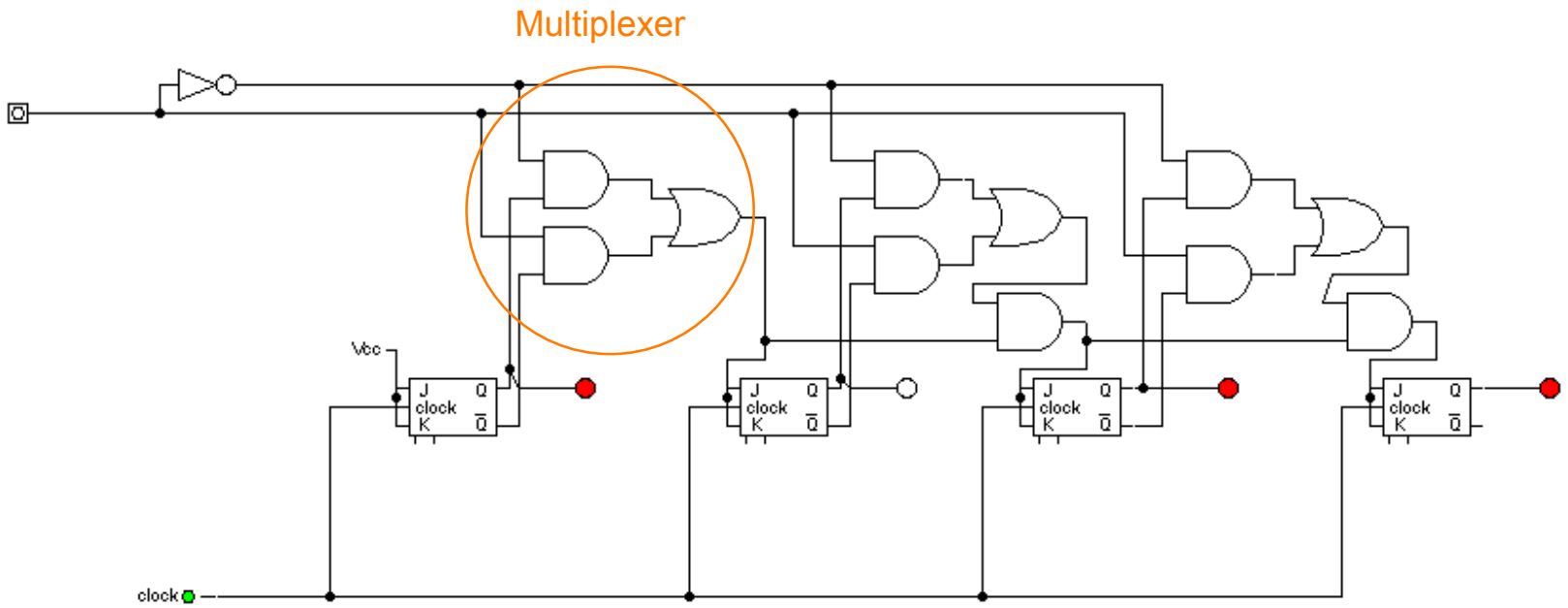
- Zeitverhalten des asynchronen Zählers:



Anmerkungen zu synchronen Zählern



Synchroner up-down-Counter



Lernziele

- Begriffe: asynchrones/synchrones Schaltwerk, sequentielle Logik, Zustandsdiagramm, Mealy/Moore-Automat, ...
- RS Flip-Flop, Master/Slave RS Flip-Flop, JK Flip-Flop, D Flip-Flop:
  - Wahrheitstabellen, Funktionsweise, Zeitverhalten
  - ungetaktete, getaktete und flankengetriggerte Flip-Flops
- einfache Schaltwerke
  - *n*-Bit Register, *n*-Bit Schieberegister
  - asynchrone/synchrone *n*-Bit Zähler
- systematischer Entwurf eines synchronen Schaltwerks aus einer Problembeschreibung
  - als Moore-Automat
  - als Mealy-Automat