technische universität
dortmund

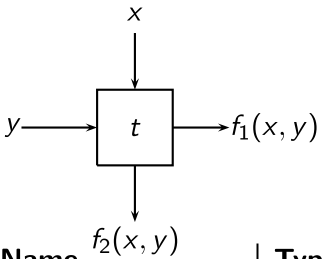
Rechnerstrukturen WS 2011/12

- Programmierbare Bausteine (Wiederholung)
- Sequenzielle Schaltungen
  - Einleitung
  - Modellierung mit Automaten
- Synchrone Schaltwerke
  - Einleitung
  - Flip-Flops

Hinweis: *Folien teilweise a. d. Basis von Materialien von Thomas Jansen*

29. August 2011

PLA Grundbausteine

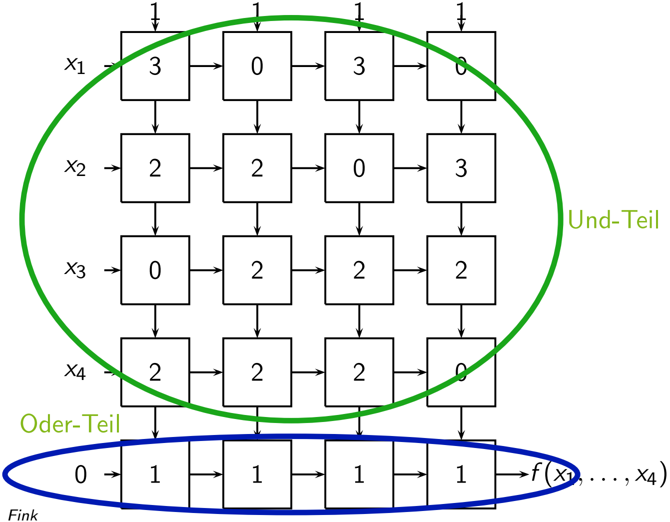


Name	Typ	$\mathbf{f_1(x, y)}$	$\mathbf{f_2(x, y)}$
Identer	0	y	x
Addierer	1	$x \vee y$	x
Multiplizierer	2	y	$x\,y$
Negat-Multiplizierer	3	y	$x\,\overline{y}$

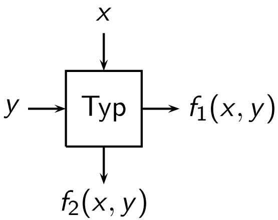
klar funktional vollständig

PLA: Ein konkretes Beispiel

Beispiel $f(x_1, x_2, x_3, x_4) = \overline{x_1}\, x_2\, x_4 \vee x_2\, x_3\, x_4 \vee \overline{x_1}\, x_3\, x_4 \vee \overline{x_2}\, x_3$



„Software"-PLAs



Typ	s	t	$f_1(x, y)$	$f_2(x, y)$
0	0	0	y	x
1	0	1	$x \vee y$	x
2	1	0	y	$x\,y$
3	1	1	y	$x\,\overline{y}$

Idee erweitere PLA-Baustein um zwei zusätzliche Eingaben, die den Baustein-Typ codieren

jetzt f_1 und f_2 als Funktionen von (s, t, x, y) darstellen

- $f_1(s, t, x, y) = y \vee \overline{s}\,t\,x$
- $f_2(s, t, x, y) = \overline{s}\,x \vee s\,x(t \oplus y)$

Fazit PLAs

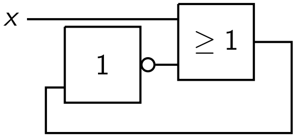
PLAs sind preiswerte, universelle Bausteine, die

- beliebige boolesche Funktionen leicht realisierbar machen,
  Jede Funktion $f\colon \{0,1\}^n \to \{0,1\}^m$, deren Polynom mit k Implikanten darstellbar ist, kann mit PLA mit $n+m$ Zeilen und k Spalten realisiert werden.

  Speicherung von 2^n Wörtern der Länge m in $(n+m) \times 2^n$-PLA möglich.
- Minimalpolynomdarstellungen motivieren,
- mit Erweiterung auf "Software-PLAs" auch mehrfach programmierbar sind.

Also: PLAs erlauben einfache und günstige Realisierung von booleschen Funktionen in kleinen Stückzahlen!

Sequenzielle Schaltungen

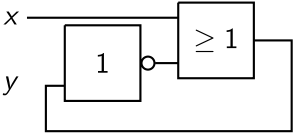


klar Das ist kein Schaltnetz.

allerdings Es ist eine „baubare“ Schaltung.

Was passiert in dieser Schaltung?

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

klar und immer so weiter. . .

natürlich in der Realität viel schneller
darum heißt die Schaltung Flimmerschaltung

Bewertung des Effekts

klar Unkontrolliertes Flimmern ist sehr unschön.

Also Kreise konsequent verbieten?

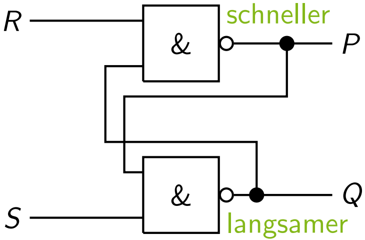
Wozu können Kreise gut sein?

Beobachtung Ausgänge werden zu Eingaben...

etwas anders Man kann schon Berechnetes noch einmal „sehen“.

Einsicht Das realisiert so etwas wie Speicher.

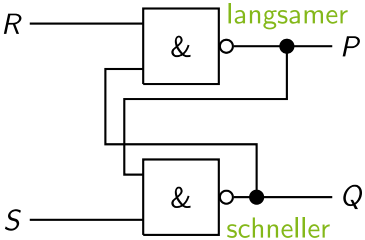
Ein zweites Beispiel



x	y	$\overline{x\,y}$
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

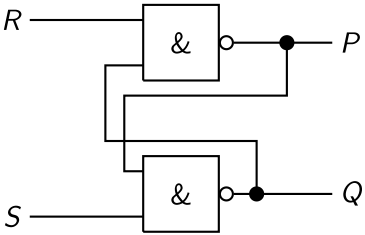
Ein zweites Beispiel



x	y	$\overline{x\,y}$
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	$\overline{P_t}$

Bi-stabile NAND-Kippstufe



Anmerkung

heißt auch Latch

positiv kippt, flimmert nicht

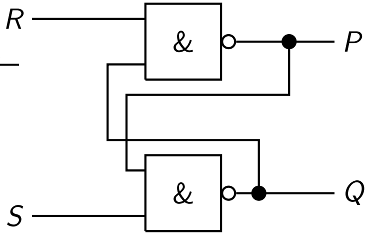
negativ Verhalten hängt von Schaltzeiten der beiden Gatter ab

genauer beobachtet Verhalten hängt manchmal von Schaltzeiten der beiden Gatter ab

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

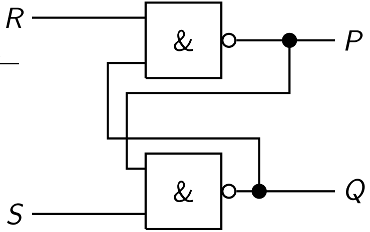


$$\begin{aligned} P_{t+2\Delta} &= \overline{R\,Q_{t+\Delta}} = \overline{R} \vee \overline{Q_{t+\Delta}} = \overline{R} \vee \overline{\overline{S\,P_{t+\Delta}}} \\ &= \overline{R} \vee S\,P_{t+\Delta} = \overline{R} \vee S\,\overline{R\,Q_t} = \overline{R} \vee S\left(\overline{R} \vee \overline{Q_t}\right) \\ &= \overline{R} \vee S\,\overline{R} \vee S\,\overline{Q_t} = \overline{R} \vee S\,\overline{Q_t} \end{aligned}$$

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	$\overline{P_t}$



$$
\begin{aligned}
P_{t+2\Delta} &= \overline{R\,Q_{t+2\Delta}} = \overline{R} \vee \overline{Q_{t+2\Delta}} = \overline{R} \vee \overline{\overline{S\,P_{t+\Delta}}} \\
&= \overline{R} \vee S\,P_{t+\Delta} = \overline{R} \vee S\,\overline{R\,Q_{t+\Delta}} = \overline{R} \vee S\,\left(\overline{R} \vee \overline{Q_{t+\Delta}}\right) \\
&= \overline{R} \vee S\,\overline{R} \vee S\,\overline{Q_{t+\Delta}} = \overline{R} \vee S\,\overline{Q_{t+\Delta}} = \overline{R} \vee S\,\overline{S\,P_t} \\
&= \overline{R} \vee S\,S\,P_t = \overline{R} \vee S\,P_t
\end{aligned}
$$

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

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

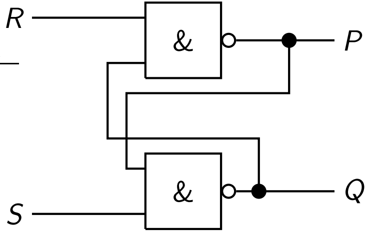
Beobachtung Wenn $P_t = \overline{Q_t}$, ist das Verhalten an P_t stabil, also von den Schaltzeiten der Gatter unabhängig.

Was ist mit dem anderen Ausgang?

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	0
1	1	$\overline{Q_t}$	Q_t	$\overline{Q_t}$	Q_t

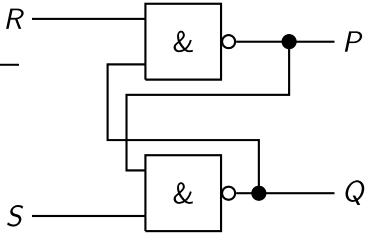


$$\begin{aligned}
Q_{t+2\Delta} &= \overline{S\,P_{t+2\Delta}} = \overline{S} \vee \overline{P_{t+2\Delta}} = \overline{S} \vee \overline{\overline{R\,Q_{t+\Delta}}} \\
&= \overline{S} \vee R\,Q_{t+\Delta} = \overline{S} \vee R\,\overline{S\,P_{t+\Delta}} = \overline{S} \vee R\left(\overline{S} \vee \overline{P_{t+\Delta}}\right) \\
&= \overline{S} \vee R\,\overline{S} \vee R\,\overline{P_{t+\Delta}} = \overline{S} \vee R\,\overline{P_{t+\Delta}} = \overline{S} \vee R\,\overline{\overline{R\,Q_t}} \\
&= \overline{S} \vee R\,R\,Q_t = \overline{S} \vee R\,Q_t
\end{aligned}$$

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	$\overline{P_t}$



$$
\begin{aligned}
Q_{t+2\Delta} &= \overline{S\,P_{t+\Delta}} = \overline{S} \vee \overline{P_{t+\Delta}} = \overline{S} \vee \overline{\overline{R\,Q_{t+\Delta}}} \\
&= \overline{S} \vee R\,Q_{t+\Delta} = \overline{S} \vee R\,\overline{S\,P_t} = \overline{S} \vee R\left(\overline{S} \vee \overline{P_t}\right) \\
&= \overline{S} \vee R\,\overline{S} \vee R\,\overline{P_t} = \overline{S} \vee R\,\overline{P_t}
\end{aligned}
$$

Fazit der Analyse der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller
$P_{t+2\Delta} = \overline{R} \vee S\,\overline{Q_t}$
$Q_{t+2\Delta} = S \vee R\,Q_t$

2. Fall unteres NAND-Gatter schneller
$P_{t+2\Delta} = \overline{R} \vee S\,P_t$
$Q_{t+2\Delta} = S \vee R\,\overline{P_t}$

also Wenn $Q_t = \overline{P_t}$, so ist das Verhalten
stabil, von den Schaltzeiten der Gatter unabhängig.

also Forderung $P_t \neq Q_t$

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!

Erstes Fazit zu sequenziellen Schaltungen

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

also

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

Wunsch strukturierter Entwurf

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

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

Definiton 19

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

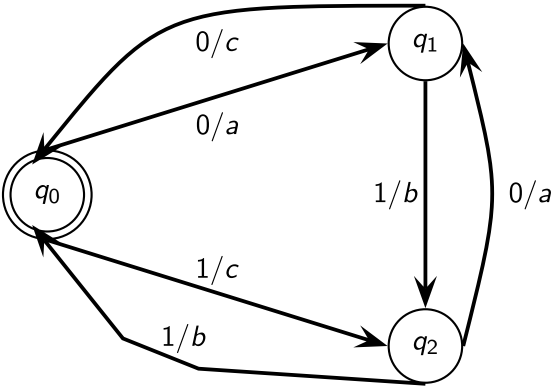
- endliche Zustandsmenge Q
- Startzustand $q_0 \in Q$
- endliches Eingabealphabet Σ
- endliches Ausgabealphabet Δ
- Zustandsüberführungsfunktion $\delta\colon Q \times \Sigma \to Q$
- Ausgabefunktion $\lambda\colon Q \times \Sigma \to \Delta \cup \{\varepsilon\}$

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

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



Eingabe 0 1 0 0 Ausgabe a b a c

Äquivalenz von Automaten

Definition

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

klar Äquivalente Automaten können unterschiedlich groß sein.

klar Man wünscht sich möglichst kleine Automaten.

Anmerkung Komplexer Problemkreis,
umfasst auch effiziente Minimierung von Automaten
$\Rightarrow$ Näher i.d. Theoretischen Informatik (GTI/TIfAI)

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

Definition Moore-Automat

Definiton 20

Ein Moore-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 $\delta\colon Q \times \Sigma \rightarrow Q$
- Ausgabefunktion $\lambda\colon Q \rightarrow \Delta \cup \{\varepsilon\}$

In einem Takt mit aktuellem Zustand q und Eingabesymbol w

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

Unterschied zum Mealy-Automaten: Ausgabe $\leftrightarrow$ Zustand

Mealy- und Moore-Automaten

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

denn zu Moore-Automat $A = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$
ist Mealy-Automat $A' = (Q, q_0, \Sigma, \Delta, \delta, \lambda')$
mit $\lambda'(q, w) := \lambda(\delta(q, w))$
offensichtlich äquivalent

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

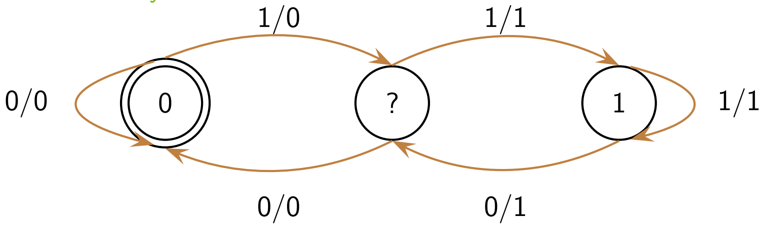
denn zu Mealy-Automaten $A = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$
ist Moore-Automat $A' = (Q', q_0', \Sigma, \Delta, \delta', \lambda')$
mit $Q' := Q \times (\Sigma \cup \{\varepsilon\})$, $q_0' := (q_0, \varepsilon)$,
$\delta'(q, w) := (\delta(q, w), w)$, $\lambda'(q') = \lambda'((q, w)) := \lambda(q, w)$
offensichtlich äquivalent

Einfacher Beispiel-Automat

Aufgabe einfache „Datenglättung“
Filtere isolierte Bits aus Datenstrom aus.

klar $\Sigma := \{0,1\}$, $\Delta := \{0,1\}$

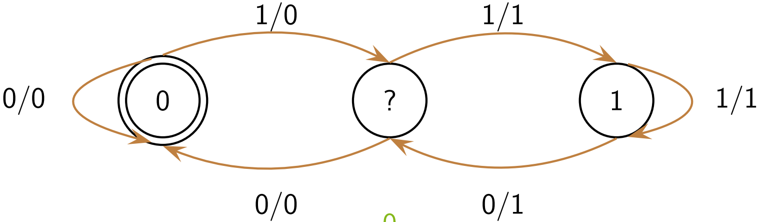
Mealy-Automat



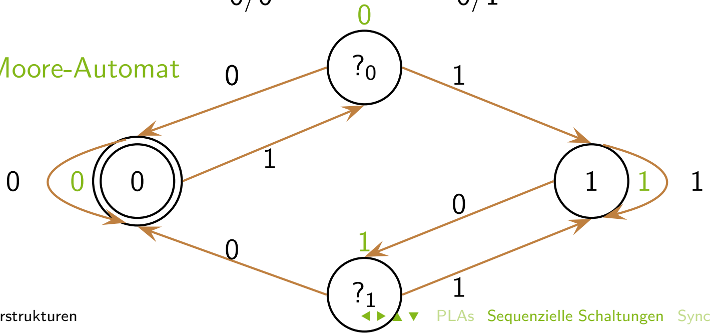
übrigens $Q := \{0,1,?\}$, $q_0 := 0$

Äquivalente Automaten „Bit-Filter“

Mealy-Automat

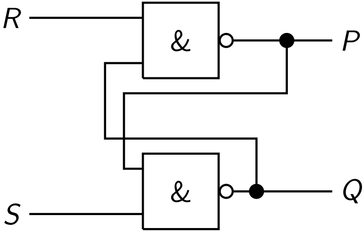


Moore-Automat

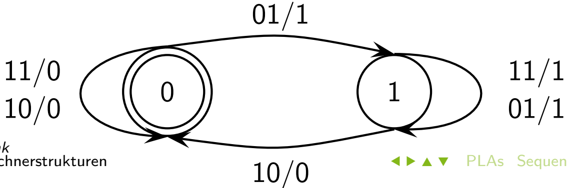


Automaten und Schaltungen: Synchrone Schaltwerke

Erinnerung bi-stabile NAND-Kippstufe



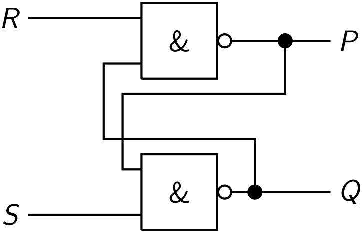
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	1	1	0
1	0	0	1
1	1	P_t	$\overline{P_t}$



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

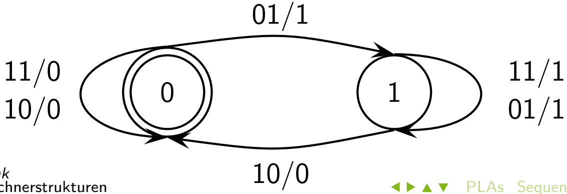
Vergleich Automat und NAND-Kippstufe

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



$Q = \{0, 1\},\ q_0 = 0$

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

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

Synchrone Schaltwerke

ab jetzt getaktete Schaltwerke

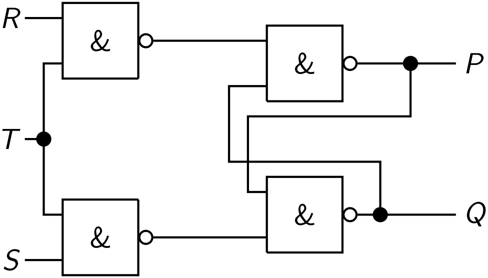
also Führe Taktsignal T ein

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

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