

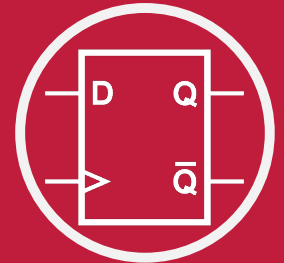


Technische
Universität
Braunschweig

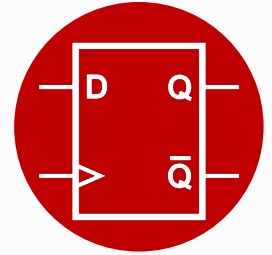


Informatik für Ingenieure – VL 7

3. ENTWURF UND ANALYSE SEQUENTIELLER SCHALTUNGEN



Flip-Flops



Wie kann die Empfindlichkeit von Latches gegen Störungen eliminiert werden?
Was muss geändert werden? → Taktflankensteuerung

Die Zustandsänderungen am Eingang werden nur in einem **kurzen Zeitfenster** wirksam, nämlich wenn das Taktsignal C wechselt ($C=0 \rightarrow 1$ oder $C=1 \rightarrow 0$) ⇒ Taktflankensteuerung

Es gibt generell zwei Möglichkeiten der Realisierung dieser Bedingung:

1. Statisch durch 2-Taktpegel-Steuerung (Master-Slave)
2. Dynamisch durch Taktflanken-Steuerung

Wie kann die Empfindlichkeit von Latches gegen Störungen eliminiert werden?
Was muss geändert werden? → Taktflankensteuerung

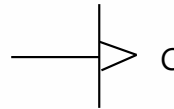
Die Zustandsänderungen am Eingang werden nur in einem **kurzen Zeitfenster** wirksam, nämlich wenn das Taktsignal C wechselt ($C=0 \rightarrow 1$ oder $C=1 \rightarrow 0$) ⇒ Taktflankensteuerung

Es gibt generell zwei Möglichkeiten der Realisierung dieser Bedingung:

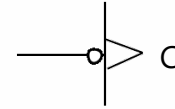
1. Statisch durch 2-Taktpegel-Steuerung (Master-Slave)
2. Dynamisch durch Taktflanken-Steuerung

Taktflankengesteuerte Schaltungen heissen **Flipflops**

Schaltzeichen für die
Taktflankensteuerung



Eingangsvariable werden beim 0 - 1
Übergang von C wirksam



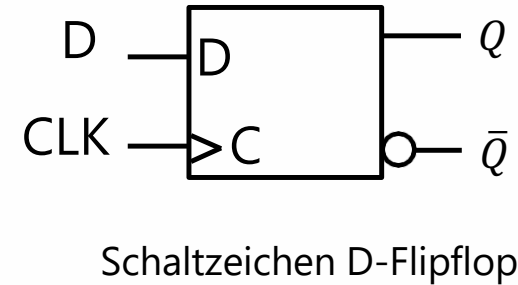
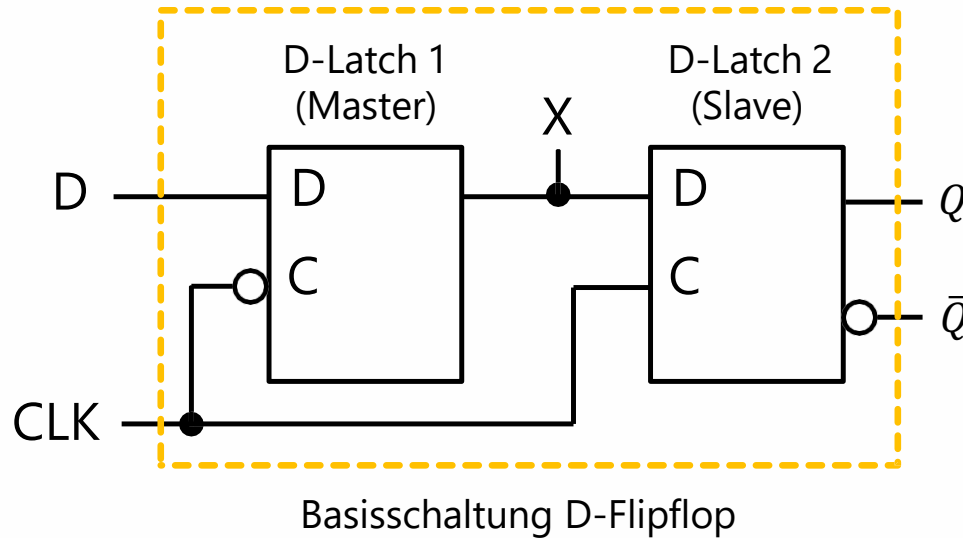
Eingangsvariable werden beim 1 - 0
Übergang von C wirksam

Aufbau des D-Flipflops

7.6

Taktzustandsgesteuerte Schaltungen können in taktflanken-gesteuerte Komponenten umgewandelt werden

Dafür müssen zwei mit **gegenphasigem Takt** gesteuerte Latches in Serie geschaltet werden:
Master-Slave Aufbau



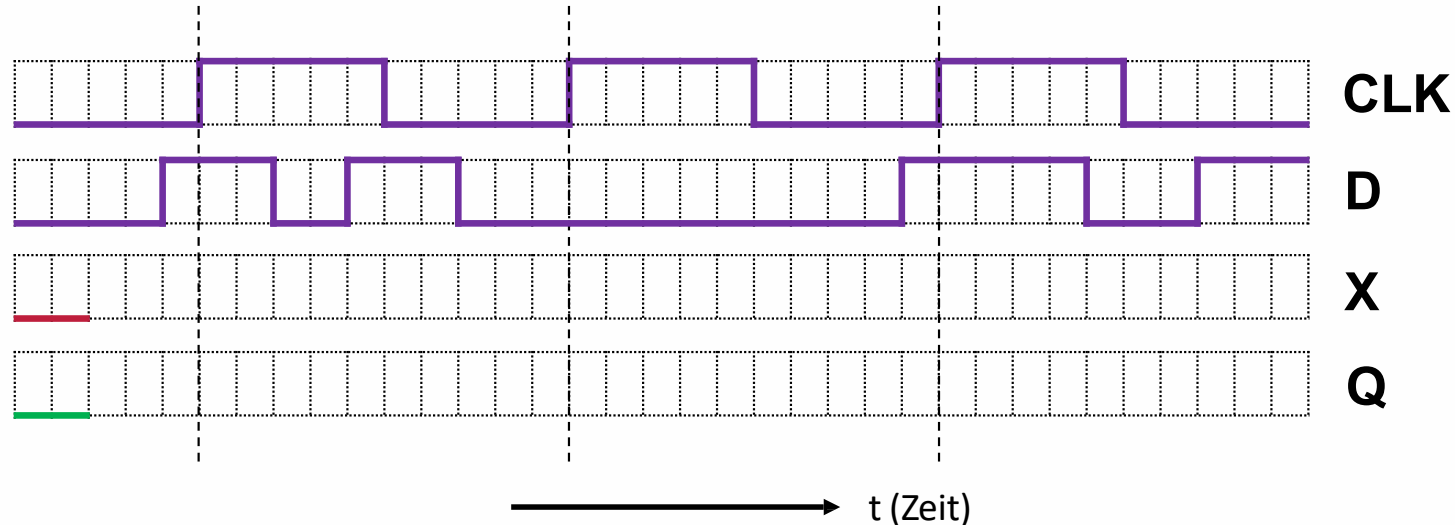
Gleichung: $Q_{n+1} = D$, wenn CLK 0→1, sonst keine Änderung

Funktionsprinzip des D-Flipflops

7.7

Die Information am **D-Latch 1 (Master)**-Eingang wird während CLK=0 eingespeichert, das D-Latch 2 ist verriegelt

Mit der Taktflanke CLK von 0 auf 1 wird **D-Latch 2 (Slave)** transparent und die Variable X erscheint am Ausgang **Q**. Während dieser Zeit ist das D-Latch 1 gesperrt

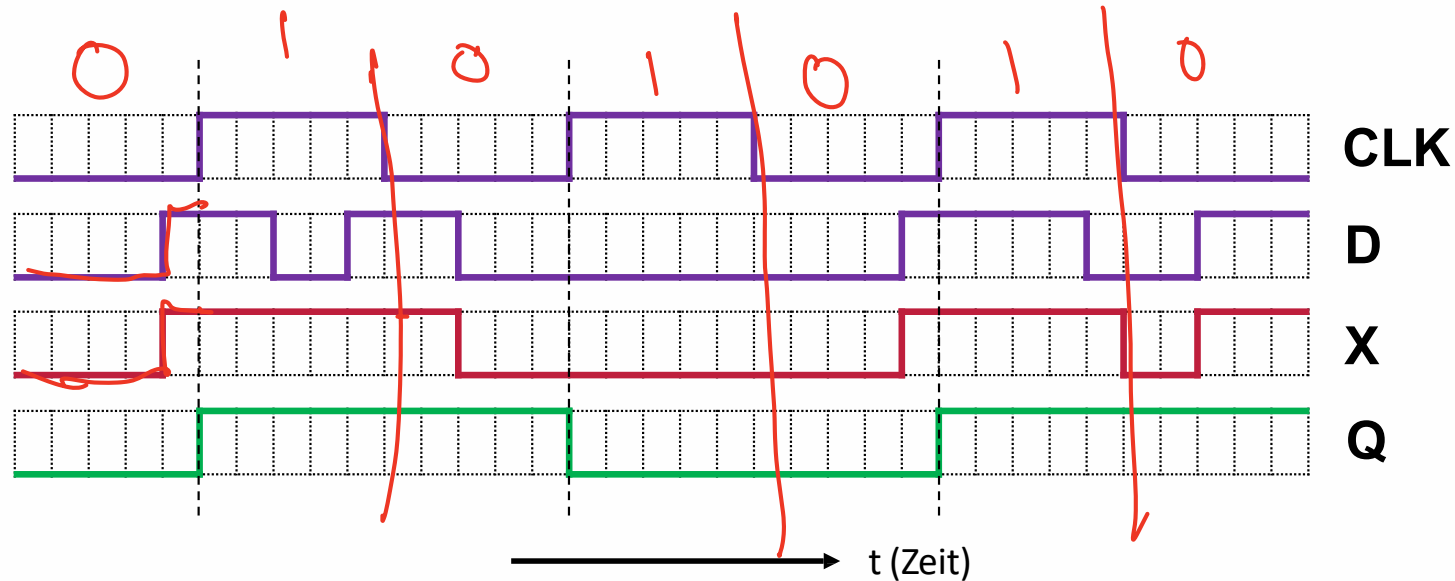


Funktionsprinzip des D-Flipflops

7.8

Die Information am **D-Latch 1 (Master)**-Eingang wird während $CLK=0$ eingespeichert, das D-Latch 2 ist verriegelt

Mit der Taktflanke CLK von 0 auf 1 wird **D-Latch 2 (Slave)** transparent und die Variable X erscheint am Ausgang **Q**. Während dieser Zeit ist das D-Latch 1 gesperrt

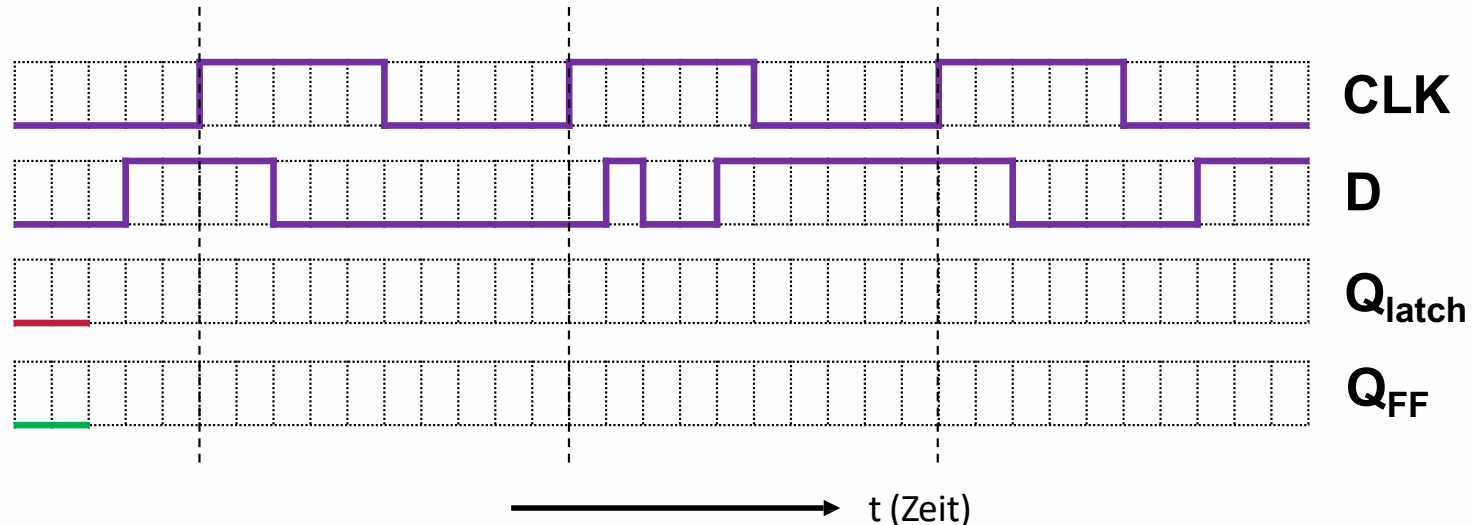


D-Latch vs. D-Flipflop

7.9

D-Latch: (*takt*)zustandsgesteuerte Schaltung. Änderungen am Eingang können während der ganzen aktiven Taktphase den Ausgang beeinflussen, da das Latch transparent ist

D-Flipflop: *taktflankengesteuerte* Kippschaltung. Der Eingangszustand zum Zeitpunkt des Taktwechsels (Flanke) wird wirksam, nicht während der Taktphase

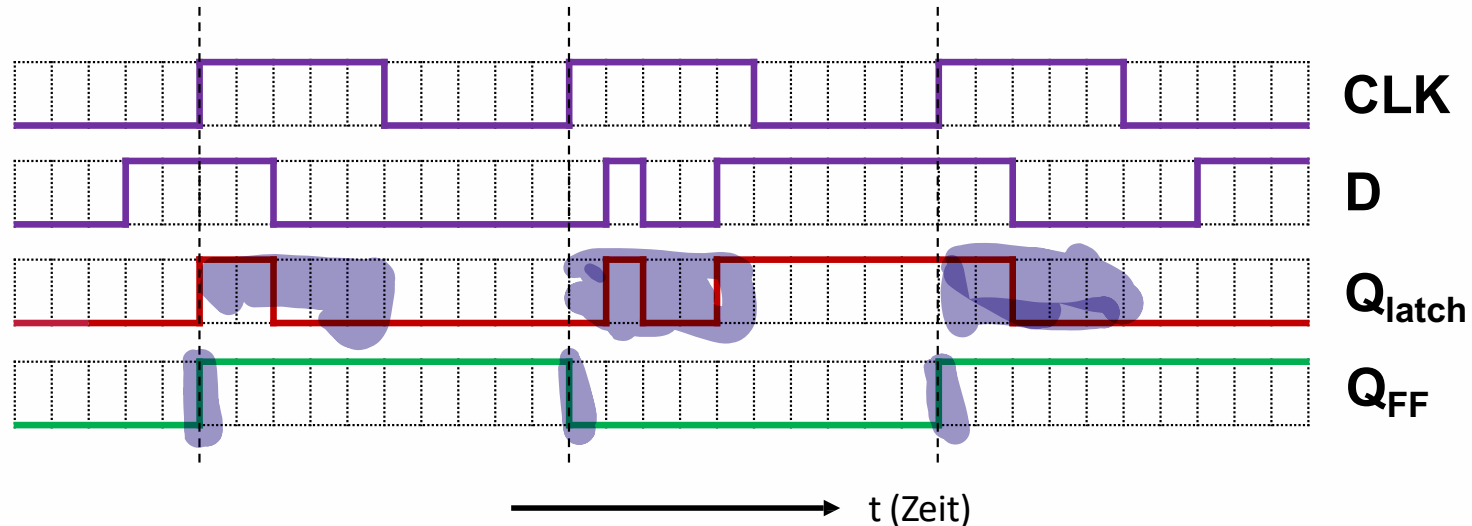


D-Latch vs. D-Flipflop

7.10

D-Latch: (*takt*)zustandsgesteuerte Schaltung. Änderungen am Eingang können während der ganzen aktiven Taktphase den Ausgang beeinflussen, da das Latch transparent ist

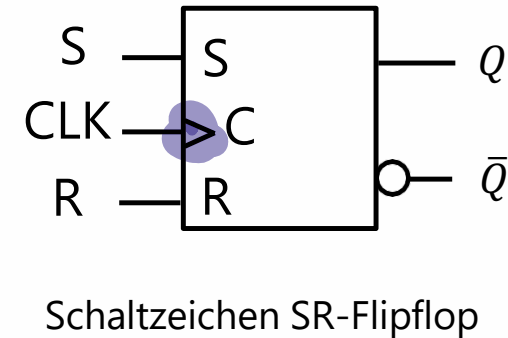
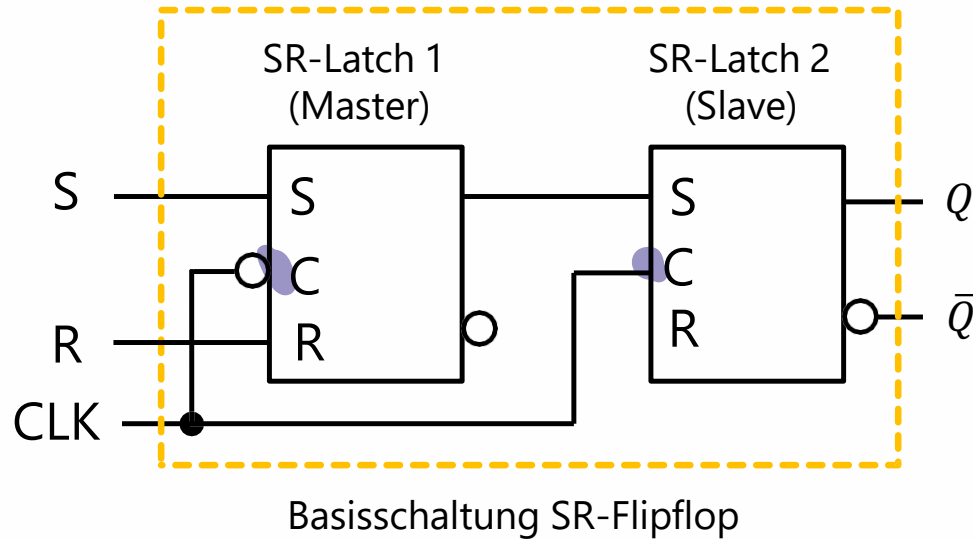
D-Flipflop: *taktflankengesteuerte* Kippschaltung. Der Eingangszustand zum Zeitpunkt des Taktwechsels (Flanke) wird wirksam, nicht während der Taktphase



Eigenschaften des SR-Flipflops

7.11

Die **Reihenschaltung** zweier SR-Latches ergibt ein SR-Flipflop



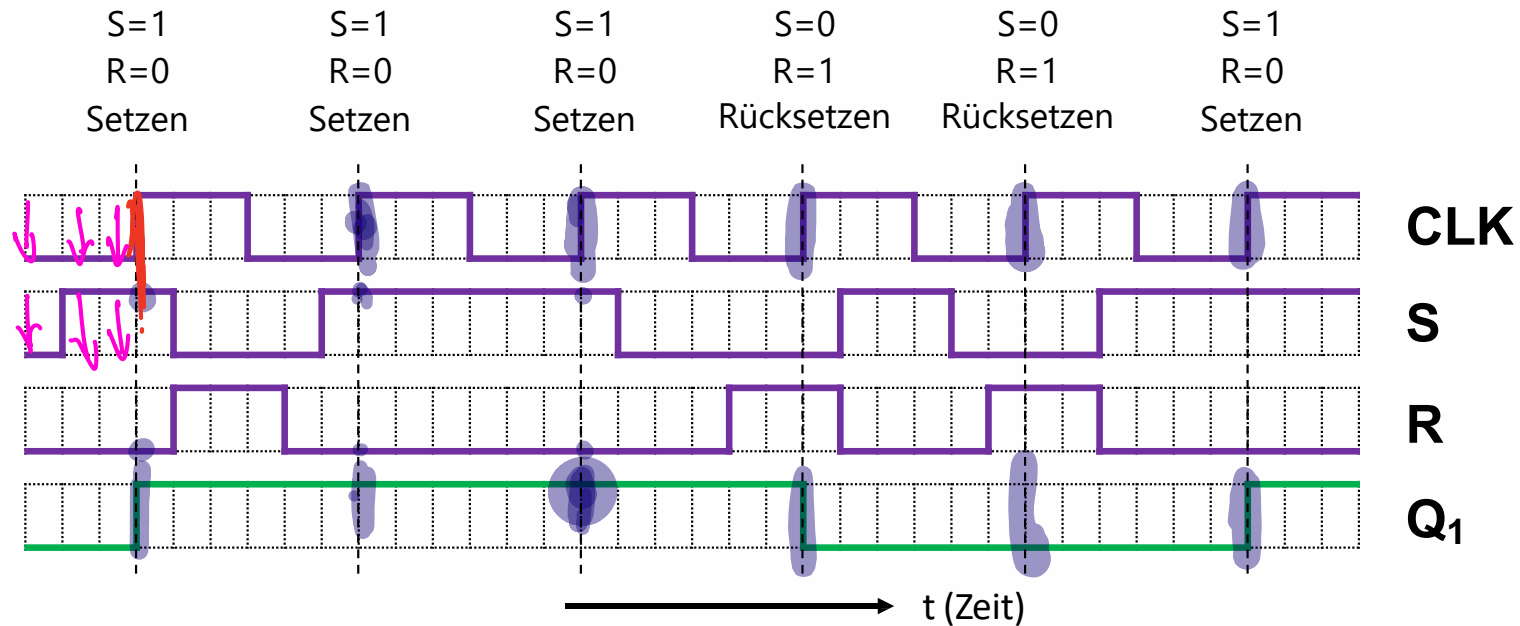
Mit der steigenden Flanke des Taktsignals CLK werden die aktuell anliegenden **Eingangswerte S und R** übernommen

$$Q_{1,n+1} = (S \vee (\bar{R} \wedge Q_1))_n \text{ unter der Bedingung } R \wedge S = 0$$

Zeitverlauf des SR-Flipflops

7.12

Zeitdiagramm eines SR-Flipflops als Funktion von **S** und **R**

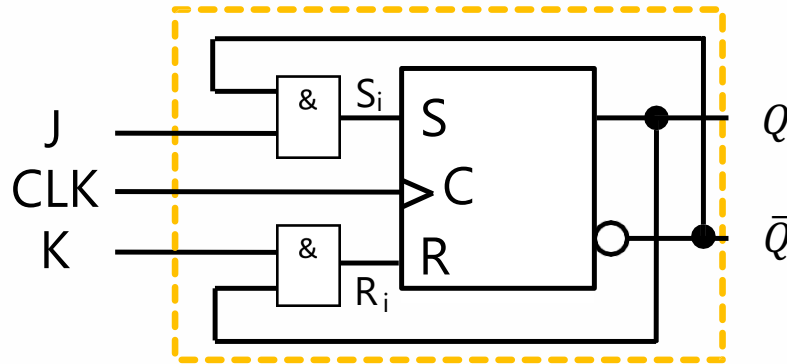


Mit der **steigenden Taktflanke** von CLK sind die Eingänge R und S wirksam, sonst ist das Flipflop verriegelt. Ausnahme: wenn $S=R=0$ bei einer aktiven Taktflanke. In diesem Fall ist die letzte Kombination vor der aktuellen Taktflanke mit $CLK=0$ und $(S=1, R=0)$ oder $(S=0, R=1)$ wirksam.

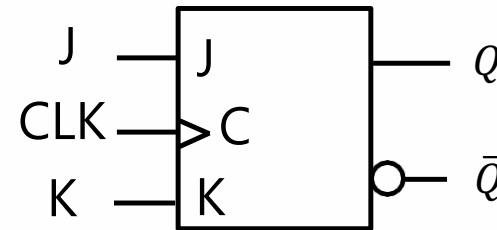
Grundlagen des JK-Flipflops (Jump-Kill Flipflops)

7.13

Das **JK-Flipflop** ist eine Erweiterung des SR-Flipflops, die bei einer aktiven Taktflanke, wenn an beiden Eingängen eine 1 liegt ($J=K=1$), den Ausgangszustand wechselt. Dieses Verhalten wird als Toggeln („kippen“) bezeichnet.



Basisschaltung JK-Flipflop



Schaltzeichen JK-Flipflop

JK-Flipflop: im Gegenteil zum SR-Flipflop können beim JK-Flipflop beide Eingänge J und K gleichzeitig 1 sein, ohne die Funktionalität der Schaltung zu stören.

Das (takt)zustandsgesteuerte **JK-Latch** existiert auch!

Funktionsgleichung des JK-Flipflops

7.14

Die Folgezustandstabelle des JK-Flipflops kann zusammengefasst werden ($Q = Q_1$; $\bar{Q} = Q_2$)

Fall	J	K	$Q_{1,n+1}$	$Q_{2,n+1}$	
1	0	0	$Q_{1,n}$	$Q_{2,n}$	speichern
2	0	1	0	1	rücksetzen
3	1	0	1	0	setzen
4	1	1	$\bar{Q}_{1,n}$	$\bar{Q}_{2,n}$	wechseln

} **SR-FF**

Aus dieser Tabelle kann die Gleichung des FFs abgeleitet werden

JK Q_{1n}	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Damit finden wir:

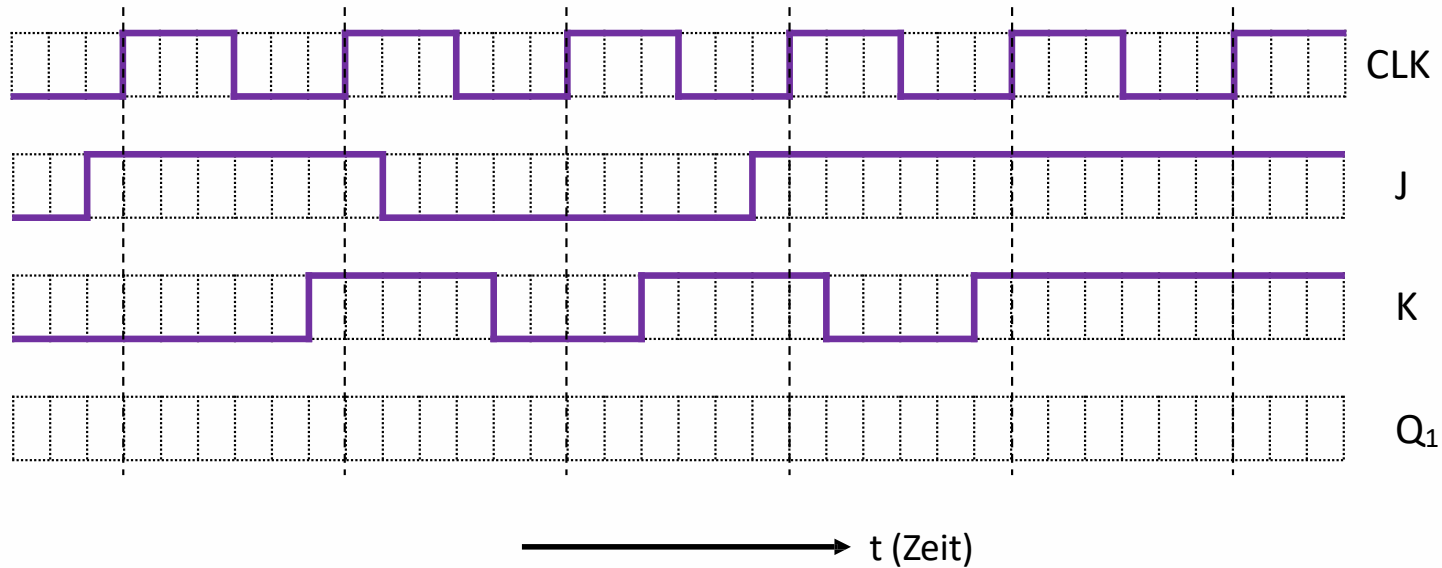
$$Q_{1,n+1} = (J \wedge \bar{Q}_{1,n}) \vee (\bar{K} \wedge Q_{1,n})$$

wenn CLK 0 \rightarrow 1

Zeitverhalten des JK-Flipflops

7.15

Wie verhält sich der Ausgang Q1 eines JK-Flipflops als Funktion von zwei zeitabhängigen Eingangsvariablen J und K?



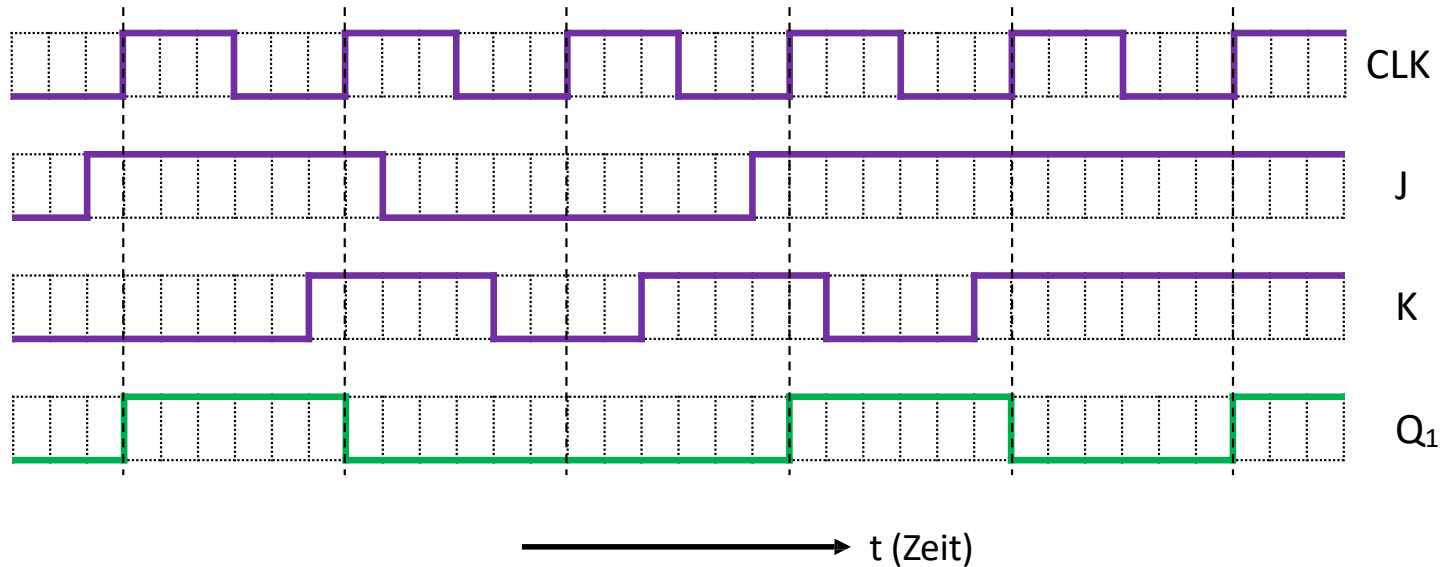
Versuchen Sie das Zeitdiagramm für den Ausgang Q1 zu vervollständigen?

Was ist der Unterschied mit SR-Flipflops?

Zeitverhalten des JK-Flipflops

7.16

Wie verhält sich der Ausgang Q1 eines JK-Flipflops als Funktion von zwei zeitabhängigen Eingangsvariablen J und K?



Versuchen Sie das Zeitdiagramm für den Ausgang Q1 zu vervollständigen?

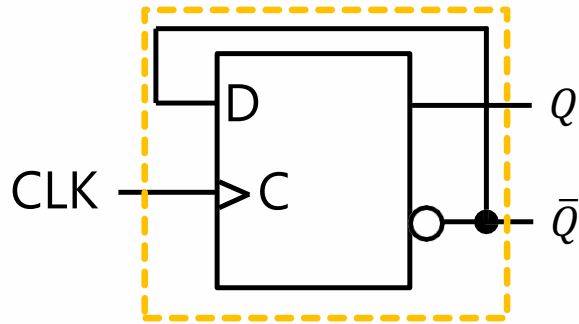
Was ist der Unterschied mit SR-Flipflops?

T(oggle)-Flipflop

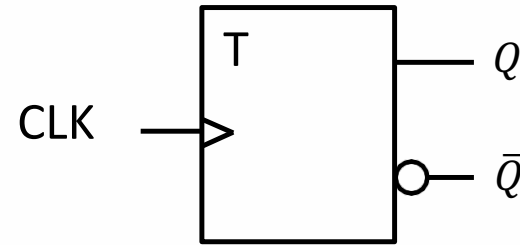
7.17

Gesucht wird eine Schaltung, die bei jeder aktiven Taktflanke in den anderen Zustand kippt (englisch: ‚toggelt‘)

Idee: der Ausgang Q_2 eines **D-FFs** wird mit dem Eingang D rückgekoppelt



Basisschaltung T-Flipflop



Schaltzeichen T-Flipflop

Funktionsgleichung: $Q_{n+1} = \bar{Q}_n$

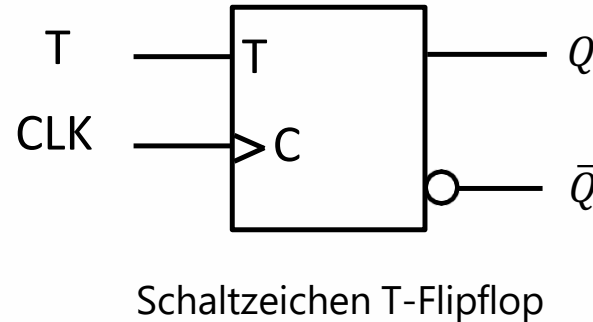
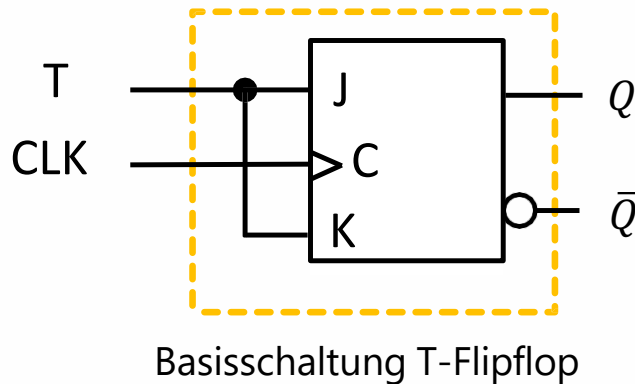
wenn Eingang CLK 0 → 1

Alternatives T-Flipflop mit 2 Eingängen

7.18

Eine **Variante** des T-Flipflops mit 2 Eingängen (CLK und T) existiert. Diese Schaltung kippt nur, wenn **T=1**. Wenn **T=0** bleibt der Ausgangszustand erhalten

Diese Schaltung kann aus einem **JK-Flipflop** realisiert werden, indem das Eingangssignal **T** mit J und K gekoppelt wird



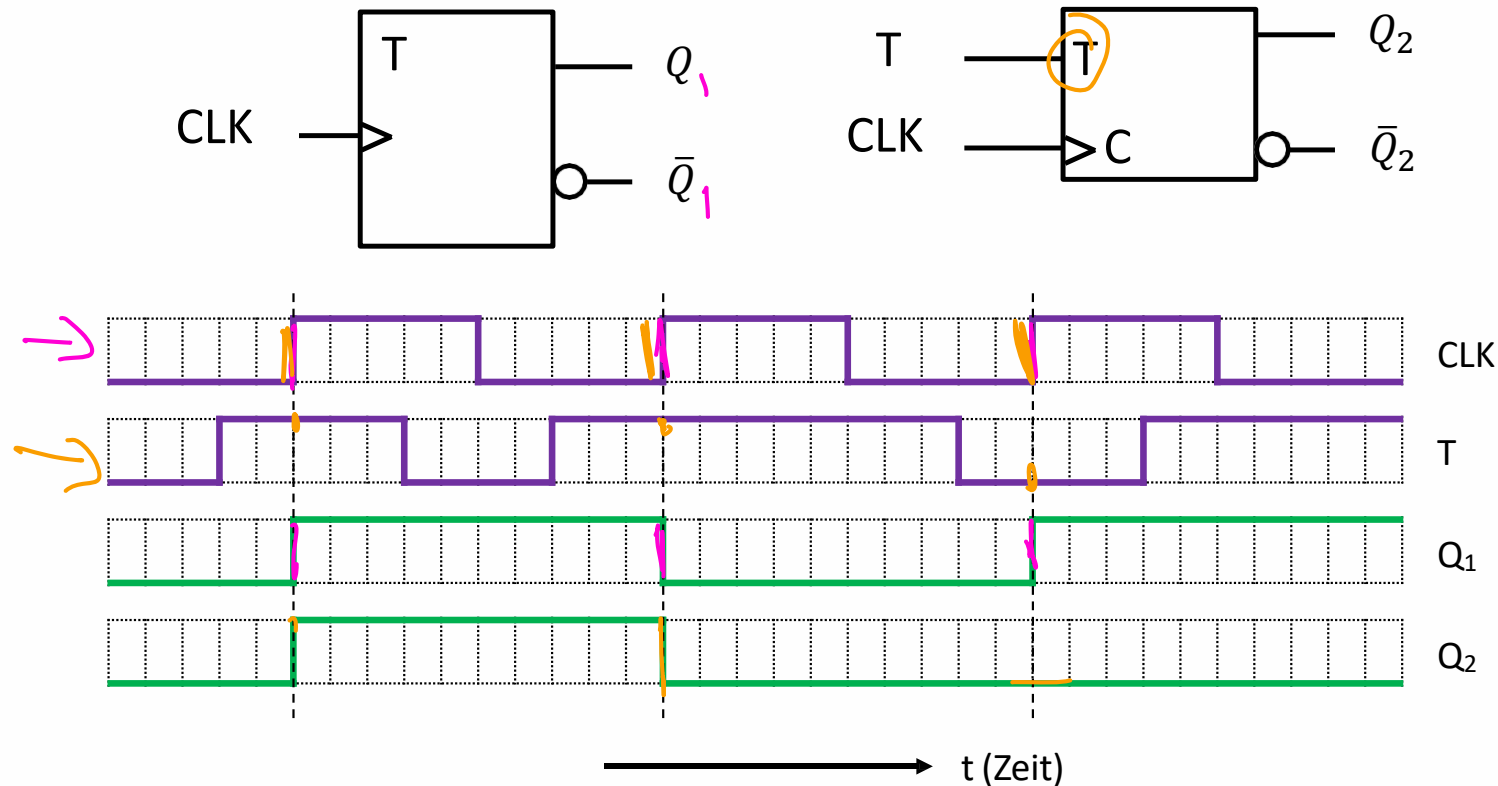
Funktionsgleichung: $Q_{n+1} = \bar{Q}_n$
wenn Eingang CLK 0 \rightarrow 1 **und** T = 1

Sonst wird das aktuelle Ausgangssignal gespeichert

Zeitverhalten des T-Flipflops

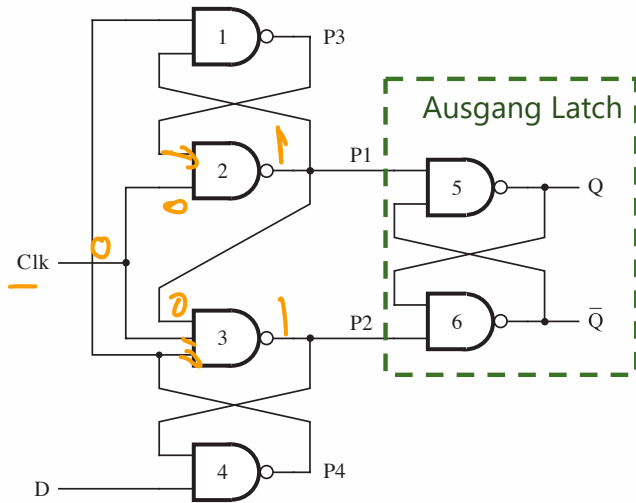
7.19

Die beiden Varianten des **T-Flipflops** weisen zwei leicht verschiedene Zeitverhalten auf, wie hier gezeigt



Taktflanken-gesteuerte D Flipflop

7.20



Detaillierte Analyse: [Video](#)

CLK = 0

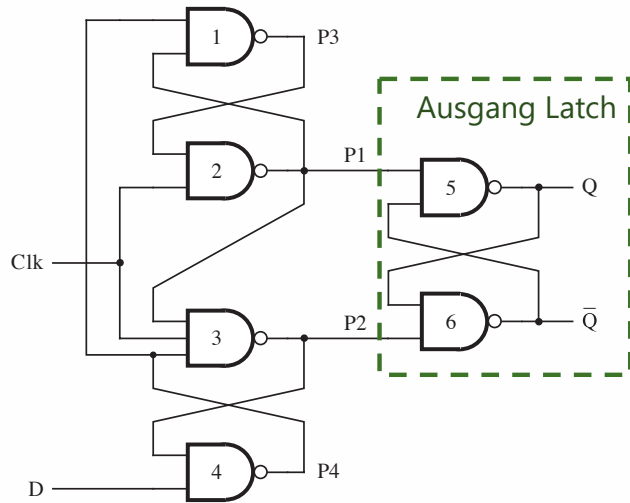
$P1 = P2 = 1 \rightarrow$ Ausgang Latch zu

$P3 = D$

$P4 = \bar{D}$

Taktflanken-gesteuerte D Flipflop

7.21



Detaillierte Analyse: [Video](#)

CLK = 0

$P1 = P2 = 1 \rightarrow$ Ausgang Latch zu

$P3 = D$

$P4 = \bar{D}$

CLK = 1

$P1 = \bar{D}$

$P2 = D$

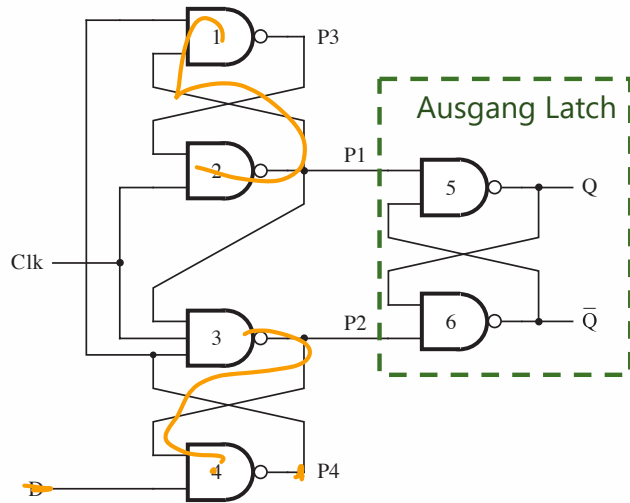


$Q = D$

$\bar{Q} = \bar{D}$

Taktflanken-gesteuerte D Flipflop

7.22



Detaillierte Analyse: [Video](#)

CLK = 0

$P1 = P2 = 1 \rightarrow$ Ausgang Latch zu

$P3 = D$

$P4 = \bar{D}$

CLK = 1

$P1 = \bar{D} \rightarrow Q = D$

$P2 = D \rightarrow \bar{Q} = \bar{D}$



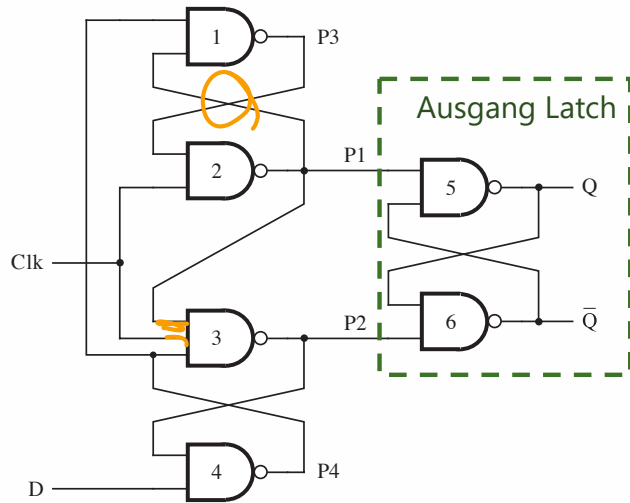
Ist dieser Flipflop transparent wenn CLK = 1?

Fall 1: D=0 während Taktflanken

P2 = 0 und
hält P4=1 solange CLK=1
(unabhängig von D)

Taktflanken-gesteuerte D Flipflop

7.23



Detaillierte Analyse: [Video](#)

CLK = 0

$P1 = P2 = 1 \rightarrow$ Ausgang Latch zu

$P3 = D$

$P4 = \bar{D}$

CLK = 1

$P1 = \bar{D} \rightarrow Q = D$

$P2 = D \rightarrow \bar{Q} = \bar{D}$



Ist dieser Flipflop transparent wenn CLK = 1?

Fall 1: D=0 während Taktflanken

P2 = 0 und
hält P4=1 solange CLK=1
(unabhängig von D)

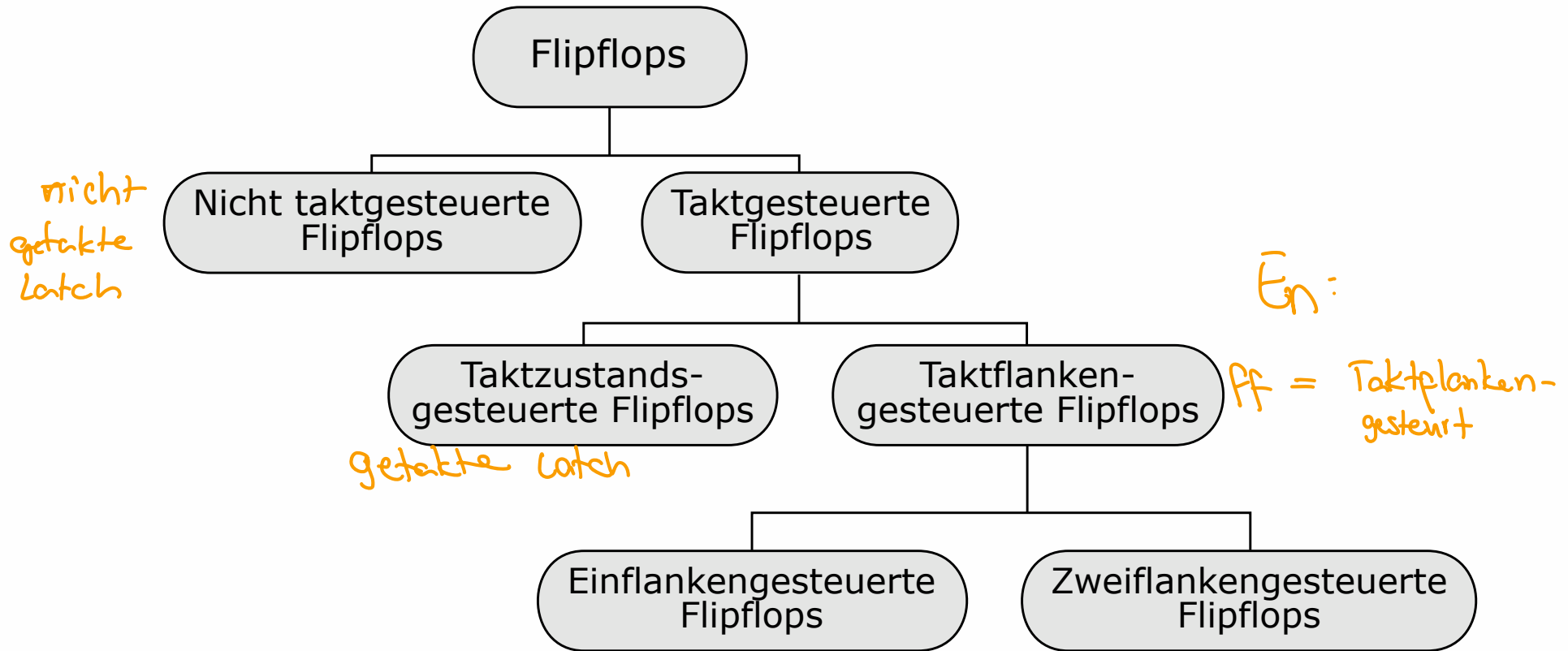
Fall 2: D=1 während Taktflanken

P1 = 0 und
hält P3=P2=1 solange CLK=1
(unabhängig von D)

Klassifizierung von Flipflops

De: ff = bistabile Schaltung

7.24



Dynamik von Flipflops (I)

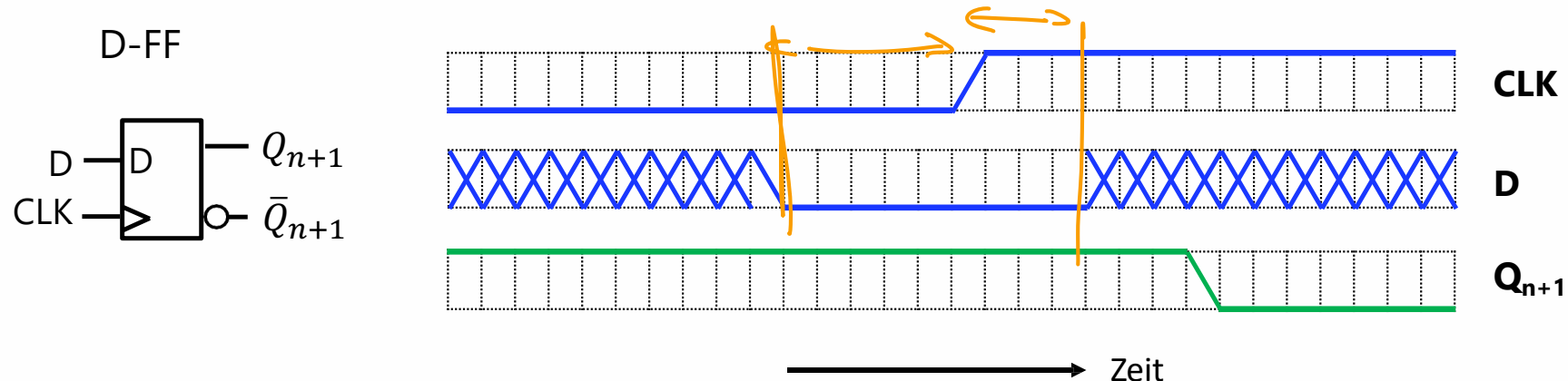
7.25

Fragen zur Dynamik von FFs, die wir beantworten wollen:

Wie schnell reagiert ein Flipflop?

Wie muss das zeitliche Zusammenspiel von den Clock (CLK) und Daten-Signalen organisiert werden, um die Funktions- sicherheit des Flipflops zu gewährleisten?

Als **Illustration**, verwenden wir das folgende Beispiel (**D-FF**)



Dynamik von Flipflops (I)

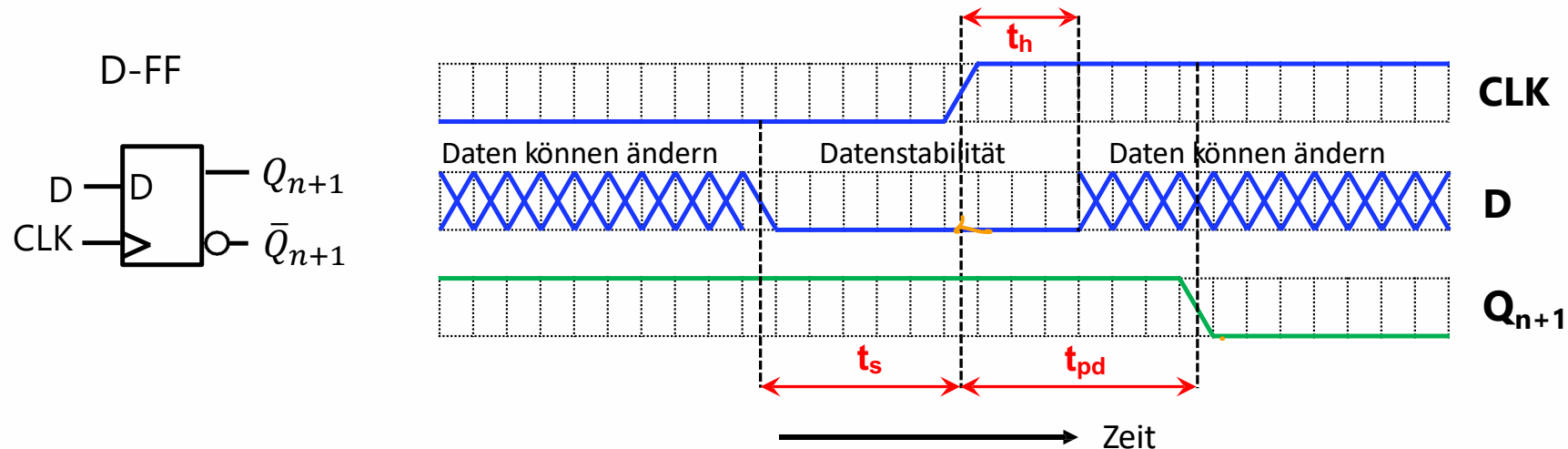
7.26

Fragen zur Dynamik von FFs, die wir beantworten wollen:

Wie schnell reagiert ein Flipflop?

Wie muss das zeitliche Zusammenspiel von den Clock (CLK) und Daten-Signalen organisiert werden, um die Funktions- sicherheit des Flipflops zu gewährleisten?

Als **Illustration**, verwenden wir das folgende Beispiel (**D-FF**)



Dynamik von Flipflops (II)

7.27

Die folgenden **dynamischen Kenndaten** werden definiert

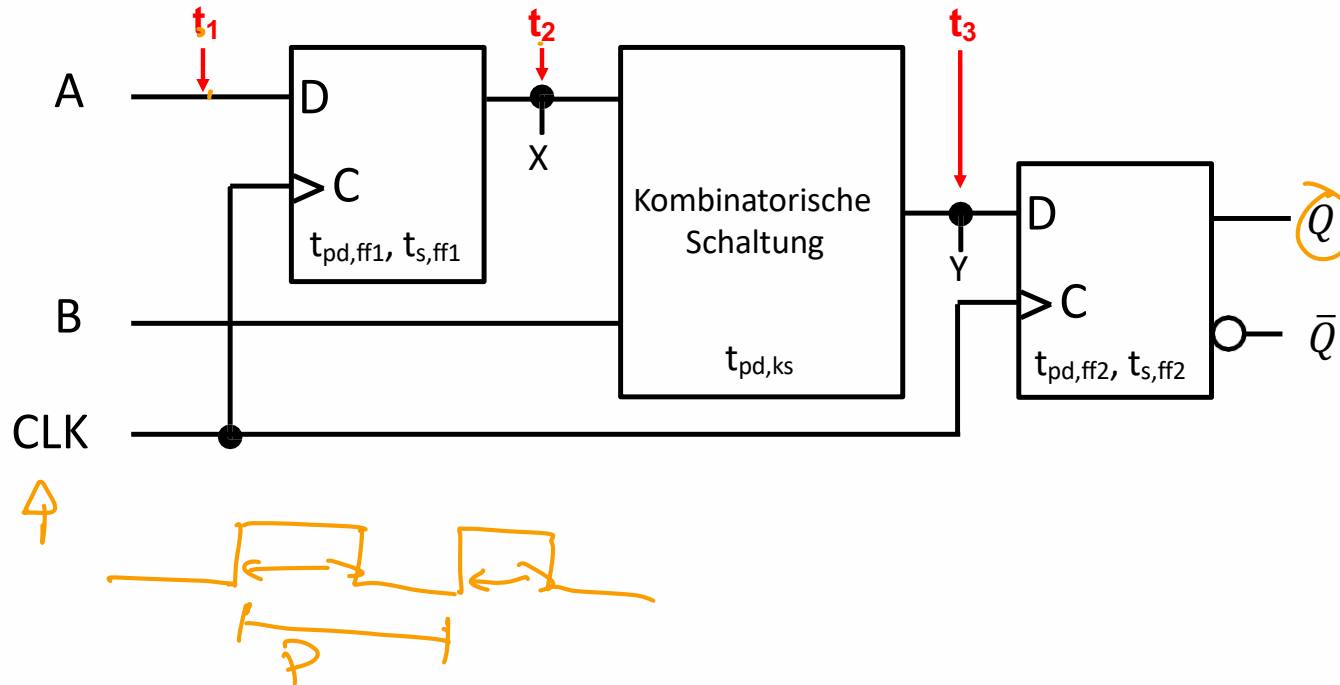
1. **t_{pd} : Verzögerungszeit** (propagation delay, Durchlaufzeit)
 t_{pd} misst die Verzögerungszeit zwischen einer aktiven Taktflanke am Clock und ihrer Reaktion am Ausgang des FFs
2. **t_s : Setup-Zeit** (Vorbereitungszeit, Einschwingzeit)
Die Setup-Zeit t_s bestimmt wie lange ein Daten-Signal (hier D) **vor** der aktiven Taktflanke unverändert anliegen muss, um sicher in das Flipflop übernommen zu werden
3. **t_h : Haltezeit** (hold time)
Mit der Haltezeit t_h wird angegeben, wie lange ein Daten Signal **nach** der aktiven Taktflanke unverändert anliegen muss, um sicher in das Flipflop übernommen zu werden

Maximale Taktfrequenz einer FF-Schaltung

7.28

In einem Schaltnetz mit mindestens zwei Flipflops in Serie, ist die **maximale Taktfrequenz** des Clocks begrenzt

Beispiel: Zeitverlauf eines Schaltnetzes mit zwei FFs in Serie

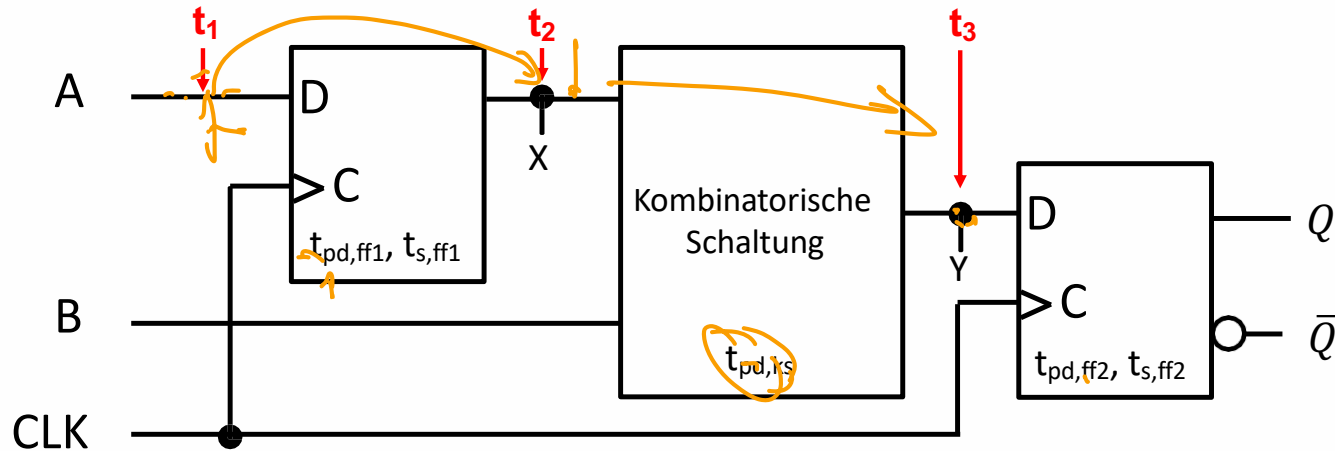


Maximale Taktfrequenz einer FF-Schaltung

7.29

In einem Schaltnetz mit mindestens zwei Flipflops in Serie, ist die **maximale Taktfrequenz** des Clocks begrenzt

Beispiel: Zeitverlauf eines Schaltnetzes mit zwei FFs in Serie



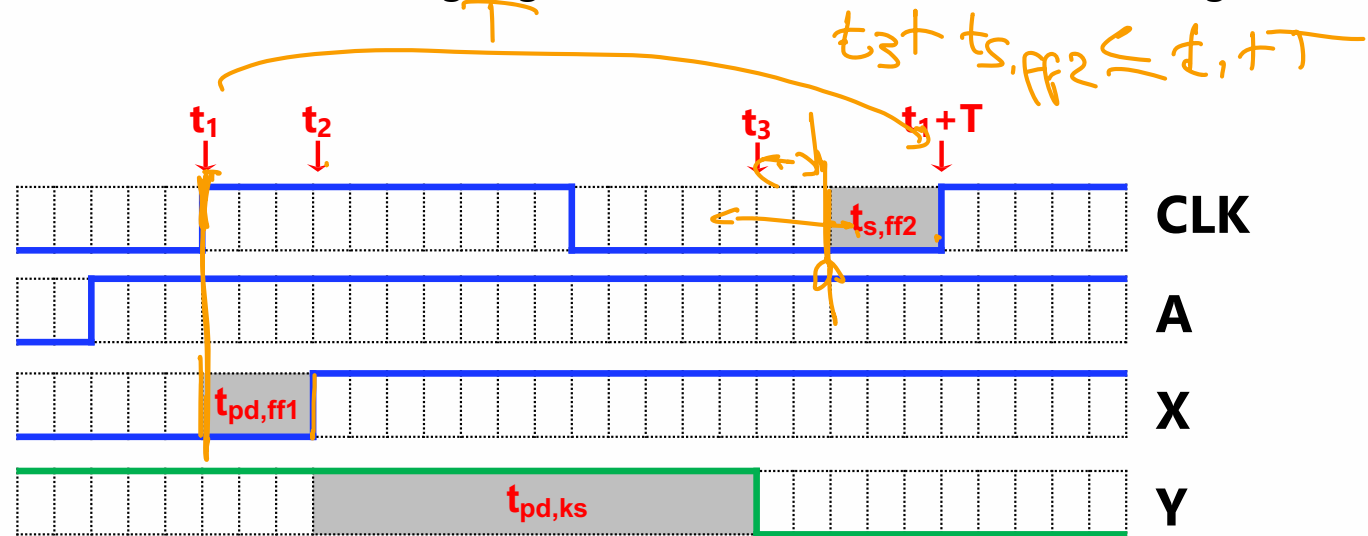
Eine Änderung am **Eingang A**, die bei einer aktiven Taktflanke detektiert wird, muss spätestens $t_{s,ff2}$ Sekunden vor der nächsten aktiven Taktflanke am **Eingang Y** des zweiten Flipflops sein, damit die Schaltung richtig funktioniert

Bestimmung der maximalen Taktfrequenz (I)

7.30

In einem Zeitdiagramm kann diese Bedingung besser visualisiert und zusammengefasst werden:

$$t_3 \leq t_1 + T - t_{s,ff2}$$



Definitionen:

t_{s,ff2} Setup-Zeit des 2. D-FFs

t_{pd,ff1} Verzögerungszeit des 1. D-FFs

t₂ t₁ + t_{pd,ff1}

t_{pd,ks} Verzögerungszeit der komb. Schaltung

t Periode des Taktsignals

t₃ t₁ + t_{pd,ff1} + t_{pd,ks}

Bestimmung der maximalen Taktfrequenz (II)

7.31

In einer mathematischen Form kann die Bedingung für die minimale Taktperiode T_{\min} so beschrieben werden

$$T_{\min} \geq t_{pd,ff1} + t_{pd,ks} + t_{s,ff2}$$

Aus dieser Gleichung kann die maximale Taktfrequenz $f_{\max} = 1/T_{\min}$ berechnet werden. In der Praxis muss der zeitlich längste Pfad innerhalb der kombinatorischen Schaltung, welche zwei Flipflops trennt, ($t_{pd,ks}$) identifiziert werden, um den Wert von T_{\min} zu erhalten.

Beispiel

Annahmen $t_{s,ff2} = 5 \text{ ns}$, $t_{pd,ff1} = 3 \text{ ns}$ und $t_{pd,ks} = 12 \text{ ns}$

$$T_{\min} \geq t_{s,ff2} + t_{pd,ff1} + t_{pd,ks} = 20 \text{ ns}$$

$$\text{Also } f_{\max} = 1/(20 \times 10^{-9}) = 50 \text{ MHz}$$

Endlich Automaten



Einführung in die sequenzielle Logik

7.33

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Es sei die Steuerung für den Antrieb eines Aufzuges zu entwerfen. Der Aufzug pendelt zwischen zwei Stockwerken (Paternosterprinzip).

Die Synchronisation des Antriebes mit dem Einsteigevorgang und Tür öffnen sei hier der Einfachheit halber nicht berücksichtigt.

Die Steuerung habe jeweils zwei Eingänge und zwei Ausgänge:

Eingänge x_1 Stockwerkkontakt oben
 x_0 Stockwerkkontakt unten

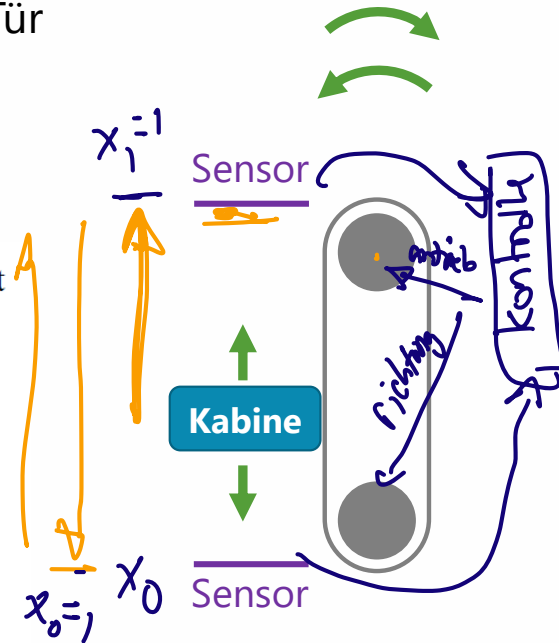
} = "1" für Kabine hat Stockwerk erreicht

Ausgänge y_0 Antrieb

"0" entspricht aus
"1" entspricht an

y_1 Antriebsrichtung

"0" entspricht nach unten
"1" entspricht nach oben



Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.34

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Für eine Schaltnetzrealisierung ergäbe sich folgende Wahrheitstabelle:

Situation	x_1	x_0	y_1	y_0
0	0	0	?	1
1	0	1	1	1
2	1	0	0	1
3	1	1	x	0

Im Falle, dass beide Stockwerkkontakte aktiv sind (**Fehlerfall, Situation 3**) soll der Antrieb abgeschaltet sein ($y_0=0$), die Richtung y_1 ist in diesem Falle don't care (x).

In den Situationen, in denen die Kabine die eindeutige Endposition erreicht hat (**Situationen 1 oder 2**) soll die Kabine in Richtung des anderen Stockwerkes fahren (nach hier nicht berücksichtigter Zeitverzögerung zum Ein-/Aussteigen). Die Antriebsrichtung ist somit klar definiert.

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.35

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Für eine Schaltnetzrealisierung ergäbe sich folgende Wahrheitstabelle:

Situation	x_1	x_0	y_1	y_0
0	0	0	?	1
1	0	1	1	1
2	1	0	0	1
3	1	1	x	0

Im Falle, dass beide Stockwerkkontakte aktiv sind (**Fehlerfall, Situation 3**) soll der Antrieb abgeschaltet sein ($y_0=0$), die Richtung y_1 ist in diesem Falle don't care (x).

In den Situationen, in denen die Kabine die eindeutige Endposition erreicht hat (**Situationen 1 oder 2**) soll die Kabine in Richtung des anderen Stockwerkes fahren (nach hier nicht berücksichtigter Zeitverzögerung zum Ein-/Aussteigen). Die Antriebsrichtung ist somit klar definiert.

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.36

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Für eine Schaltnetzrealisierung ergäbe sich folgende Wahrheitstabelle:

Situation	x_1	x_0	y_1	y_0
0	0	0	?	1
1	0	1	1	1
2	1	0	0	1
3	1	1	x	0

Im Falle, dass beide Stockwerkkontakte aktiv sind (**Fehlerfall, Situation 3**) soll der Antrieb abgeschaltet sein ($y_0=0$), die Richtung y_1 ist in diesem Falle don't care (x).

In den Situationen, in denen die Kabine die eindeutige Endposition erreicht hat (**Situationen 1 oder 2**) soll die Kabine in Richtung des anderen Stockwerkes fahren (nach hier nicht berücksichtigter Zeitverzögerung zum Ein-/Aussteigen). Die Antriebsrichtung ist somit klar definiert.

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.37

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Für eine Schaltnetzrealisierung ergäbe sich folgende Wahrheitstabelle:

Situation	x_1	x_0	y_1	y_0
0	0	0	?	1
1	0	1	1	1
2	1	0	0	1
3	1	1	x	0

Im Falle, dass beide Stockwerkkontakte aktiv sind (**Fehlerfall, Situation 3**) soll der Antrieb abgeschaltet sein ($y_0=0$), die Richtung y_1 ist in diesem Falle don't care (x).

In den Situationen, in denen die Kabine die eindeutige Endposition erreicht hat (**Situationen 1 oder 2**) soll die Kabine in Richtung des anderen Stockwerkes fahren (nach hier nicht berücksichtigter Zeitverzögerung zum Ein-/Aussteigen). Die Antriebsrichtung ist somit klar definiert.

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.38

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Situation	x_1	x_0	y_1	y_0
→ 0	0	0	?	1 ←
1	0	1	1	1
2	1	0	0	1
3	1	1	x	0

Wenn die Kabine zwischen zwei Stockwerken ist (Situation 0; $x_1 = x_0 = 0$), hängt die zu wählende Antriebsrichtung y_1 von der "Vorgeschichte" ab, d. h. ob die Kabine gerade nach oben oder nach unten fährt.

Dies ist mit einem reinem Schaltnetz nicht zu realisieren.

Man könnte aber z. B. die aktuelle Antriebsrichtung y_1 auch als Eingang für das Schaltnetz nutzen, d. h. **eine Rückkopplung** vom Ausgang auf den Eingang durchführen.

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.39

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Situation	x_1	x_0	y_1	y_0
0	0	0	?	1
1	0	1	1	1
2	1	0	0	1
3	1	1	x	0

Wenn die Kabine zwischen zwei Stockwerken ist (Situation 0; $x_1 = x_0 = 0$), hängt die zu wählende Antriebsrichtung y_1 von der "Vorgeschichte" ab, d. h. ob die Kabine gerade nach oben oder nach unten fährt.

Dies ist mit einem reinem Schaltnetz nicht zu realisieren.

Man könnte aber z. B. die aktuelle Antriebsrichtung y_1 auch als Eingang für das Schaltnetz nutzen, d. h. **eine Rückkopplung** vom Ausgang auf den Eingang durchführen.

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.40

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Dies würde in diesem Beispiel möglich sein, da sich durch die einfache Betriebsweise folgender Bewegungsablauf ergibt:

1. Aufzug in Fahrt nach unten
2. Aufzug unten angekommen, Antrieb umgeschaltet, Aufzug nach oben
3. Aufzug nach oben, gerade unten losgefahren
4. Aufzug in Fahrt nach oben
5. Aufzug oben angekommen, Antrieb umgeschaltet, Aufzug nach unten
6. Aufzug nach unten, gerade oben losgefahren

→ weiter bei 1. (Kombinationen 011 und 111 nur im Fehlerfall)

interne Zustand

$y_1(\text{alt})$	x_1	x_0	$y_1(\text{neu})$	y_0
0	0	0	0	1
0	0	1	1	1
1	0	1	1	1
1	0	0	1	1
1	1	0	0	1
0	1	0	0	1

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Die Zuordnung der Antriebsumschaltung ist richtungseindeutig, daher könnte direkt rückgekoppelt werden. Dies ist jedoch nicht bei allen Problemstellungen der Fall.

Um Oszillationen zu vermeiden ist es sicherer, die "Vorgeschichte" des Systems in eigenen Variablen, **den Zustandsvariablen**, zu beschreiben und zu realisieren.

Für das Beispiel benötigt man eine Zustandsvariable z_0 für die zwei Bewegungszustände:

$z_0 = "0"$ entspricht Fahrt nach unten

$z_0 = "1"$ entspricht Fahrt nach oben

Die Bewegungszustände werden in diesem Fall also binär codiert in der Zustandsvariable dargestellt.

Es sind auch andere Codierungen der Zustände möglich und gebräuchlich.

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.42

Beispiel: Antriebssteuerung eines einfachen Aufzuges:

vollständige Funktionstabelle, (die Indizes n , $n+1$ kennzeichnen den zeitlichen Zustandsablauf)

Ein- gänge		Zu- stand	Folge- zustand	Aus- gänge		Bewegungszustand
x_1	x_0	z_0^n	z_0^{n+1}	y_1	y_0	
0	0	0	0	0	1	Fahrt nach unten
0	1	0	1	1	1	unten angekommen, Antrieb umgeschaltet auf Fahrt nach oben
1	0	0	0	0	1	Fahrt nach unten, gerade oben losgefahren
1	1	0	-	-	0	Fehler: Aufzug angehalten
0	0	1	1	1	1	Fahrt nach oben
0	1	1	1	1	1	Fahrt nach oben, gerade unten losgefahren
1	0	1	0	0	1	oben angekommen, Antrieb umgeschaltet auf Fahrt nach unten
1	1	1	-	-	0	Fehler: Aufzug angehalten

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.43

Beispiel: Antriebssteuerung eines einfachen Aufzuges:

vollständige Funktionstabelle, (die Indizes n , $n+1$ kennzeichnen den zeitlichen Zustandsablauf)

Ein- gänge		Zu- stand	Folge- zustand	Aus- gänge		Bewegungszustand
x_1	x_0	z_0^n	z_0^{n+1}	y_1	y_0	
0	0	0	0	0	1	Fahrt nach unten
0	1	0	1	1	1	unten angekommen, Antrieb umgeschaltet auf Fahrt nach oben
1	0	0	0	0	1	Fahrt nach unten, gerade oben losgefahren
1	1	0	-	-	0	Fehler: Aufzug angehalten
0	0	1	1	1	1	Fahrt nach oben
0	1	1	1	1	1	Fahrt nach oben, gerade unten losgefahren
1	0	1	0	0	1	oben angekommen, Antrieb umgeschaltet auf Fahrt nach unten
1	1	1	-	-	0	Fehler: Aufzug angehalten

Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.44

Beispiel: Antriebssteuerung eines einfachen Aufzuges:

vollständige Funktionstabelle, (die Indizes n , $n+1$ kennzeichnen den zeitlichen Zustandsablauf)

Leistungssignal

Ein-gänge		Zu-stand	Folge-zustand	Aus-gänge		Bewegungszustand
x_1	x_0	z_0^n	z_0^{n+1}	y_1	y_0	
0	0	0	0	0	1	Fahrt nach unten
0	1	0	1	1	1	unten angekommen, Antrieb umgeschaltet auf Fahrt nach oben
1	0	0	0	0	1	Fahrt nach unten, gerade oben losgefahren
1	1	0	-	-	0	Fehler: Aufzug angehalten
0	0	1	1	1	1	Fahrt nach oben
0	1	1	1	1	1	Fahrt nach oben, gerade unten losgefahren
1	0	1	0	0	1	oben angekommen, Antrieb umgeschaltet auf Fahrt nach unten
1	1	1	-	-	0	Fehler: Aufzug angehalten

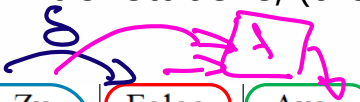
Einführung in die sequenzielle Logik

y_0 - Antrieb
 y_1 - Richtung

7.45

Beispiel: Antriebssteuerung eines einfachen Aufzuges:

vollständige Funktionstabelle, (die Indizes n , $n+1$ kennzeichnen den zeitlichen Zustandsablauf)



Ein-gänge		Zu-stand	Folge-zustand	Aus-gänge		Bewegungszustand
x_1	x_0	z_0^n	z_0^{n+1}	y_1	y_0	
0	0	0	0	0	1	Fahrt nach unten
0	1	0	1	1	1	unten angekommen, Antrieb umgeschaltet auf Fahrt nach oben
1	0	0	0	0	1	Fahrt nach unten, gerade oben losgefahren
1	1	0	-	-	0	Fehler: Aufzug angehalten
0	0	1	1	1	1	Fahrt nach oben
0	1	1	1	1	1	Fahrt nach oben, gerade unten losgefahren
1	0	1	0	0	1	oben angekommen, Antrieb umgeschaltet auf Fahrt nach unten
1	1	1	-	-	0	Fehler: Aufzug angehalten

Einführung in die sequenzielle Logik

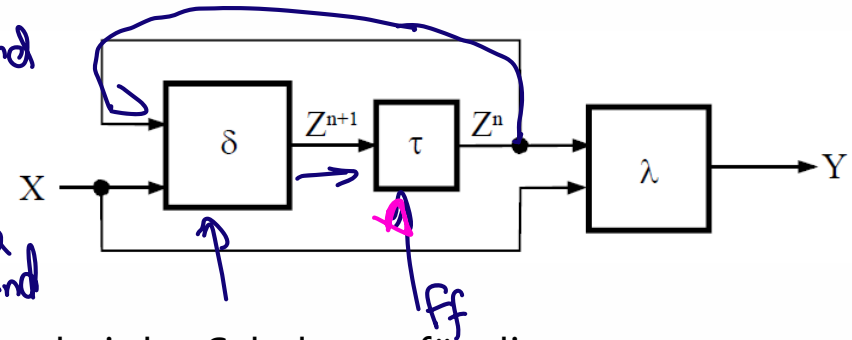
7.46

Beispiel: Antriebssteuerung eines einfachen Aufzuges

Die Steuerung lässt sich also ganz allgemein mit zwei folgenden kombinatorischen Logikelementen realisieren:

1. Ausgabefunktion: $Y = \lambda(X, Z^n)$

2. Zustandsfunktion: $Z^{n+1} = \delta(X, Z^n)$



Es ergibt sich eine Struktur aus zwei Schaltnetzen wobei das Schaltnetz für die Zustandsfortschaltung δ die Rückkopplung aufweist.

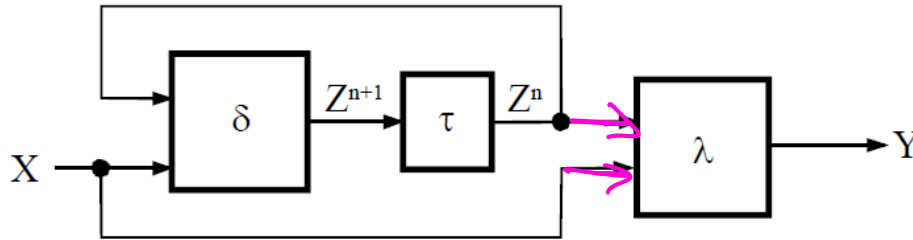
Um hier sicher zu stellen, dass die Zustände auch zeitlich voneinander getrennt wechseln ist eine Zeitverzögerung (τ) in die Rückkopplung eingebaut.

Ohne die Zeitverzögerung wäre die Zustandswechselzeit gleich der Durchlaufzeit durch das Schaltnetz δ (könnte aber minimal = 0 sein).

Einführung in die sequenzielle Logik

7.47

Eine solche Struktur mit zwei Schaltnetzen die teilweise rückgekoppelt sind und eine bestimmte Zeitverzögerung für die Zustandsfortschaltung aufweisen, werden auch **determinierte Automaten** genannt, in diesem Fall ein sogenannter „**Mealy-Automat**“.



Der **Mealy-Automat** ist durch seine Eingänge, Ausgänge, Zustände sowie den beiden Schaltnetzen λ für die Erzeugung der Ausgänge und δ für die Erzeugung der Folgezustände als „**Automatenfunktion A**“ vollständig definiert:

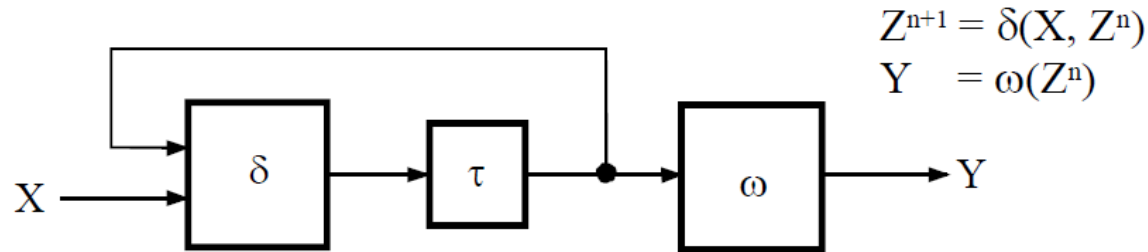
$$A_{\text{mealy}} = [X, Y, Z, \lambda, \delta]$$

Andere Automatentypen

7.54

„Moore Automat“

$$A_{\text{Moore}} = A[X, Y, Z, \delta, \omega]$$



Beim Moore Automaten basiert die Ausgangsfunktion nur auf dem aktuellen Zustand Z^n .

Moore und Mealy Automaten können ineinander überführt werden, da der Eingangsvektor X durch das Netz δ im Zustandsvektor Z enthalten ist. Die Ausganstransformation (hier ω) muss somit entsprechend gegenüber dem Mealy Automaten angepasst werden.