

Latches und Flipflops

- RS- Latch
- D-Latch
- D-Flipflop
- JK-Flipflop
- T-Flipflop
- Zweispeicher Flipflop
- RTL-Modellierung synchroner Schaltungen



Prof. Dr. J. Reichardt
Lehrbuch Digitaltechnik
Eine Einführung mit VHDL, 3. A
Oldenbourg Wissenschaftsverlag
München 2013
ISBN 978-3-486-72765-4

Latches und Flipflops in synchronen Schaltungen

- Charakteristisch für das sequenzielle Verhalten einer Digitalschaltung ist die Fähigkeit einer Erinnerung.
- Die Fähigkeit dieser „Erinnerung“ haben **Latches und Flipflops**. Dabei wird ein einzelnes Bit gespeichert. Falls mehrere Bits gespeichert werden, so spricht man von einem **Register**.
- Charakteristisch für eine Speicherschaltung ist die Rückführung eines Ausgangs auf den Eingang (Problem dabei: Schaltung darf nicht ins Schwingen kommen!)

Latches werden durch Pegel angesteuert. Die Ausgänge eines Latches können sich zu jedem Zeitpunkt ändern. **Flipflops** bzw. Register werden hingegen durch Flanken, also Pegelübergänge angesteuert. Die Ausgänge von Flipflops können sich nur nach Wechsel eines Taktsignals ändern.

- In (voll)synchronen Schaltungen werden alle Speicherschaltungen zum gleichen Zeitpunkt mit einem (meist) gemeinsamen Takt angesteuert.

Zustand und Folgezustand

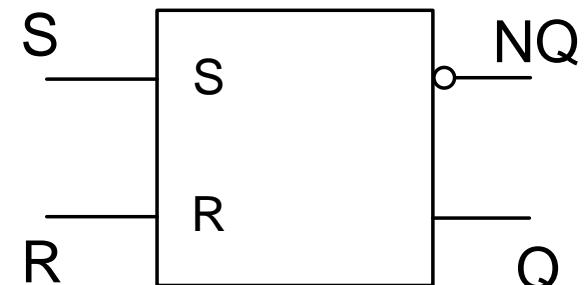
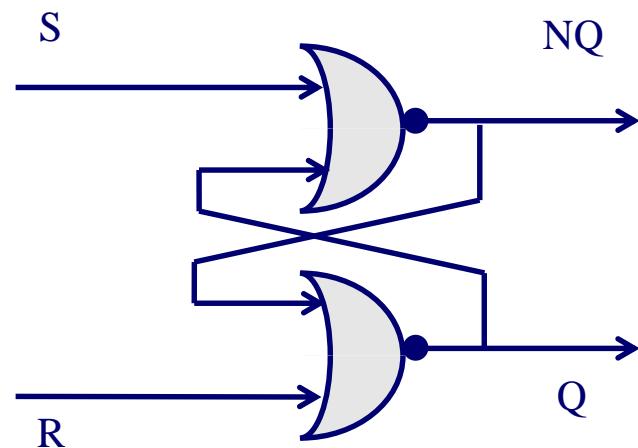
- Die Analyse der in den Speicherschaltungen vorhandenen kombinatorischen Rückkopplung erfordert zwingend die Berücksichtigung einer Zeitverzögerung, die in den VHDL-Modellen als symbolische Verzögerung modelliert wird. Formal wird dies durch die Begriffe „Zustand“ und „Folgezustand“ beschrieben.

Zu jedem Zeitpunkt befindet sich eine sequenzielle Schaltung in einem definierten Zustand Z , der durch Zustandsbits in einem Zustandsregister gespeichert wird. Nach der Ansteuerung wird der vorausberechnete Folgezustand Z^+ zum neuen aktuellen Zustand. Der jeweilige Folgezustand berechnet sich aus dem aktuellen Zustand, den aktuellen Eingangssignalen und einer kombinatorischen Übergangslogik.

- Ähnlich wie eine Wahrheitstabelle das Verhalten der kombinatorischen Logik beschreibt, dient eine Folgezustandstabelle zur Beschreibung eines endlichen Zustandsautomaten .

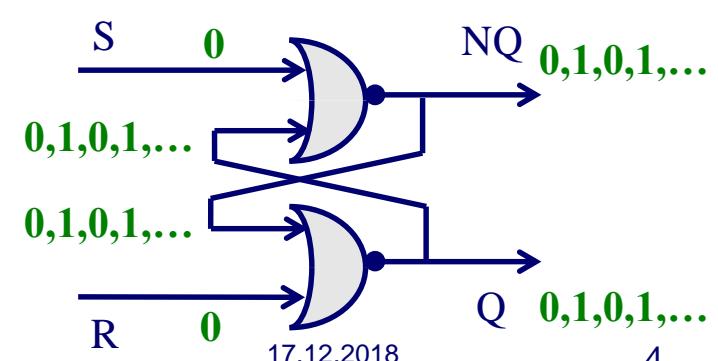
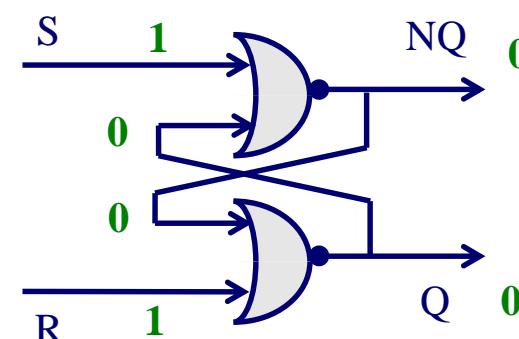
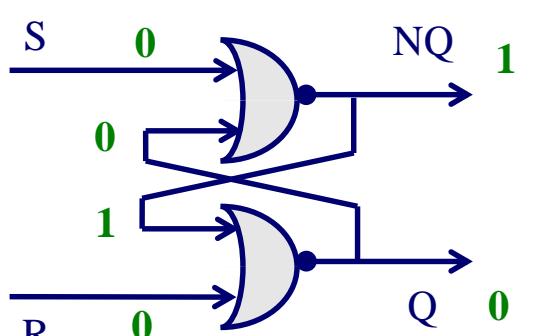
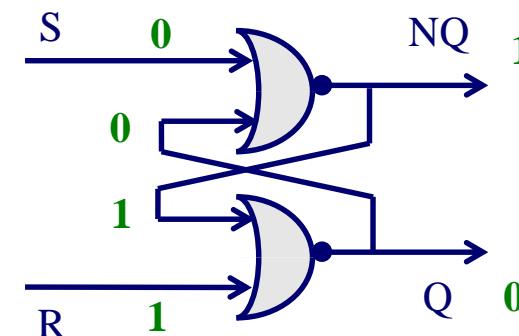
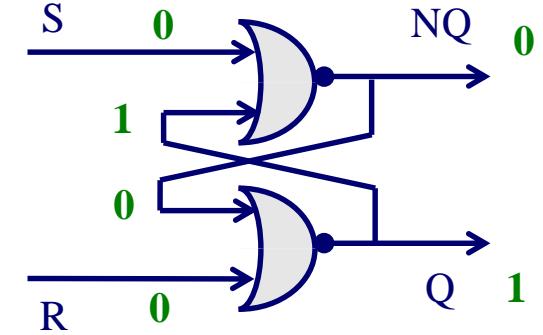
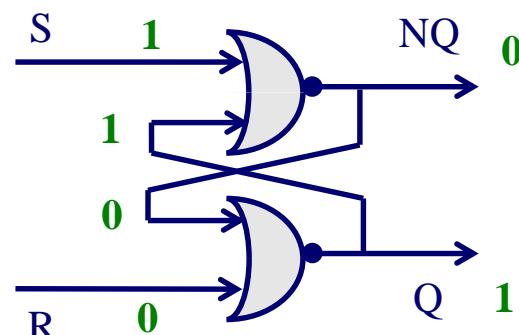
RS-Latch

- Dient als Ausgangspunkt für alle weiteren Speicherschaltungen (Basis-RS-Latch).
- Innerer Aufbau mit rückgekoppelten NOR-Gattern und Schaltsymbol:



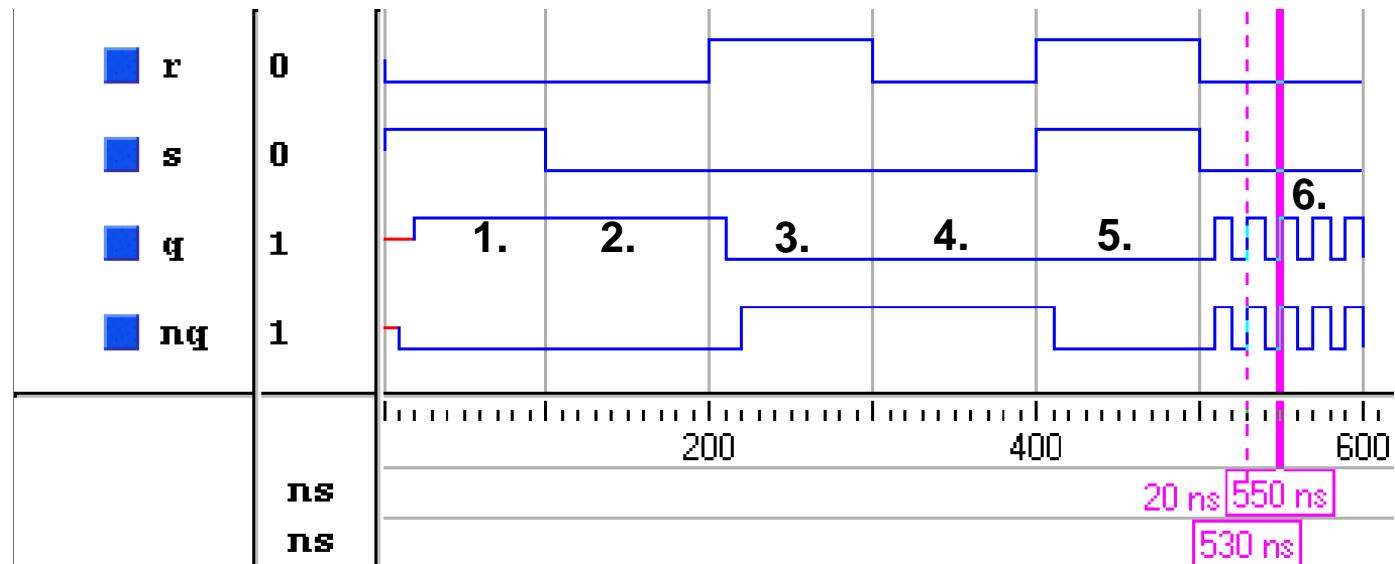
Analyse des sequenziellen Verhaltens beim RS-Latch

1. Setzen mit $S=1$
2. Speichern mit $R=S=0$.
Die Schaltung ist
stabil für $Q=0$ und $Q=1$
3. Löschen mit $R=1$
4. Speichern wie 2
5. Irreguläres Verhalten
bei $S=1$ und $R=1$
6. Speichern mit $R=S=0$
nach irregulärer
Ansteuerung. Die
Schaltung ist
instabil/gerät ins
Schwingen



Analyse des sequenziellen Verhaltens beim RS-Latch

1. Setzen mit $S=1$
2. Speichern mit $R=S=0$.
Die Schaltung ist
stabil für $Q=0$ und $Q=1$
3. Löschen mit $R=1$
4. Speicher wie 2
5. Irreguläres Verhalten
bei $S=1$ und $R=1$
6. Speichern mit $R=S=0$
nach irregulärer
Ansteuerung. Die
Schaltung ist
instabil/gerät ins
Schwingen



Folgezustandstabelle und Synthesetabelle des NOR-Latches

Folgezustandstabelle:

Nr	S	R	Q	Q ⁺	NQ ⁺	Bedeutung
4.	0	0	0	0	1	Speichern
2.	0	0	1	1	0	Speichern
3.	0	1	X	0	1	Löschen
1.	1	0	X	1	0	Setzen
5.	1	1	X	0	0	Irregulär

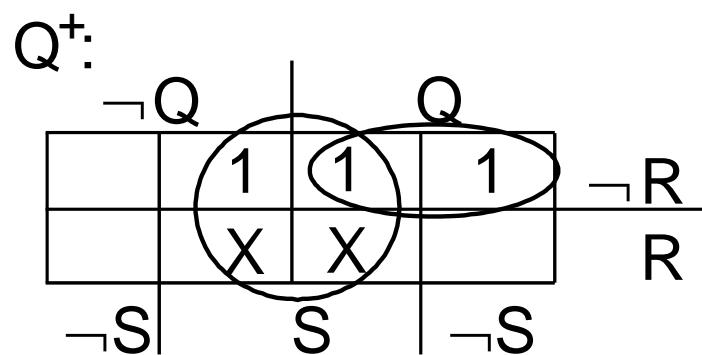
Arbeitstabelle

S	R	Q ⁺
0	0	Q
0	1	0
1	0	1
1	1	0

Um den irregulären Zustand des RS-Latches zu vermeiden, darf dieses niemals mit $R = S = 1$ angesteuert werden..

KV-Diagramm und charakteristische Gleichung

- KV-Diagramm zur Bestimmung des Folgezustands unter der Annahme (Nebenbedingung), dass der irreguläre Zustand niemals eingenommen wird ($R = S = 1$ wird mit Don't-Care-Termen berücksichtigt):



- Daraus erhält man die charakteristische Gleichung des RS-Latches:

$$Q^+ = S \vee (Q \wedge \overline{R}) \text{ unter der Nebenbedingung } S \wedge R = 0$$

VHDL-Datenflussmodell des RS-Latches

- Besteht aus kreuzgekoppelten NOR-Gattern

```
entity RSLATCH is
    port( R, S : in bit; -- Setzen/Ruecksetzen
          Q, NQ: out bit);-- Ausgaenge
end RSLATCH;
-----
architecture DATENFLUSS of RSLATCH is
signal Q_TEMP, NQ_TEMP: bit;
begin
    NQ_TEMP <= S nor Q_TEMP after 10 ns;
    Q_TEMP <= R nor NQ_TEMP after 10 ns;
    NQ <= NQ_TEMP; -- Kopie an den Ausgang
    Q <= Q_TEMP;   -- Kopie an den Ausgang
end DATENFLUSS;
```

Ermöglicht Vermeidung
von buffer
Ausgangsports !

- Alternativ sind auch kreuzgekoppelte NAND-Gatter möglich. Dann wird das RS-Latch durch L-aktive Signale angesteuert.

VHDL-Verhaltensmodell

- Verwendet den Datentyp `std_logic` zur Modellierung des irregulären Zustands (`U`)

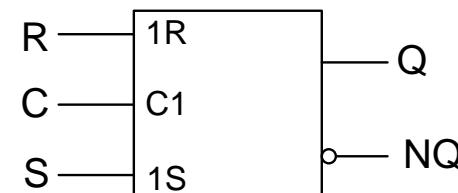
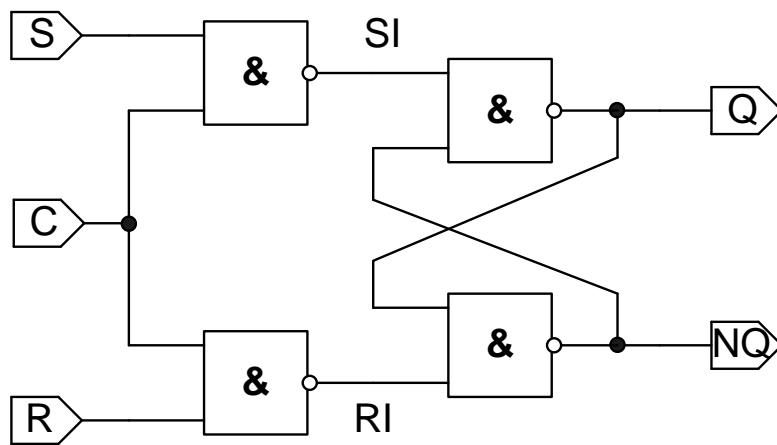
```
library ieee;
use ieee.std_logic_1164.all;
entity RSLATCHX is
  port( R, S : in std_logic;      -- Setzen/Ruecksetzen
        Q, NQ: out std_logic);
end RSLATCHX;
architecture VERHALTEN of RSLATCHX is
signal Q_TEMP: std_logic;
begin
process(R, S, QTEMP)
begin
  if      (S='1' and R='0') then Q_TEMP <= '1' after 10 ns;          --Setzen
  elsif  (S='0' and R='1') then Q_TEMP <= '0' after 10 ns;          --Rücksetzen
  elsif  (S='0' and R='0') then Q_TEMP <=  Q_TEMP after 10 ns;        --Speichern
  else   Q_TEMP <= 'U';                                              --Irregulaer
  end if;
end process;
Q <= Q_TEMP;
NQ <=  not Q_TEMP;
end VERHALTEN;
```

Verhaltensmodellierung
durch einen Prozess mit
`if`!

gewünschte
kombinatorische Schleife !

Taktzustandsgesteuertes RS-Latch

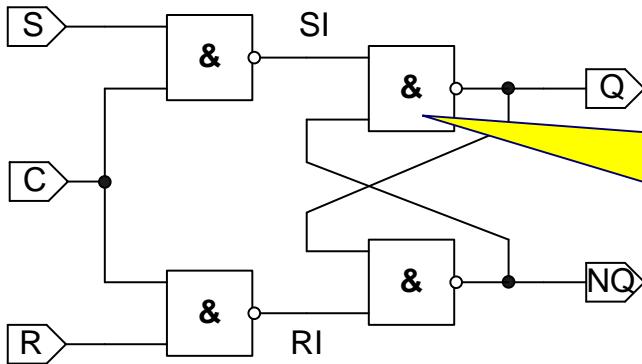
- Eine Torschaltung aus zwei NANDs schaltet die R- und S-Funktionen frei => zusätzlicher Eingang C erforderlich.
- Das innere RS-Latch muss L-aktiv angesteuert werden => besteht aus NANDs
- Struktur a) und Schaltsymbol b):



Abhängigkeitsnotation in Schaltsymbolen:

- Ziffern, die sich *hinter* Buchstaben befinden, sind **steuernde** Signale. Die Ziffern werden aufsteigend durchnummieriert.
- Wenn sich an Signalein- oder -ausgängen Ziffern vor Buchstaben befinden, werden die entsprechenden Signale durch das Signal **gesteuert**, welches diese Ziffer trägt.
- Der Buchstabe C bedeutet „erlaubt Aktion“.
- Der Buchstabe S bedeutet „Setzen“ und R bedeutet „Rücksetzen“.

Wahrheits- bzw. Folgezustandstabelle



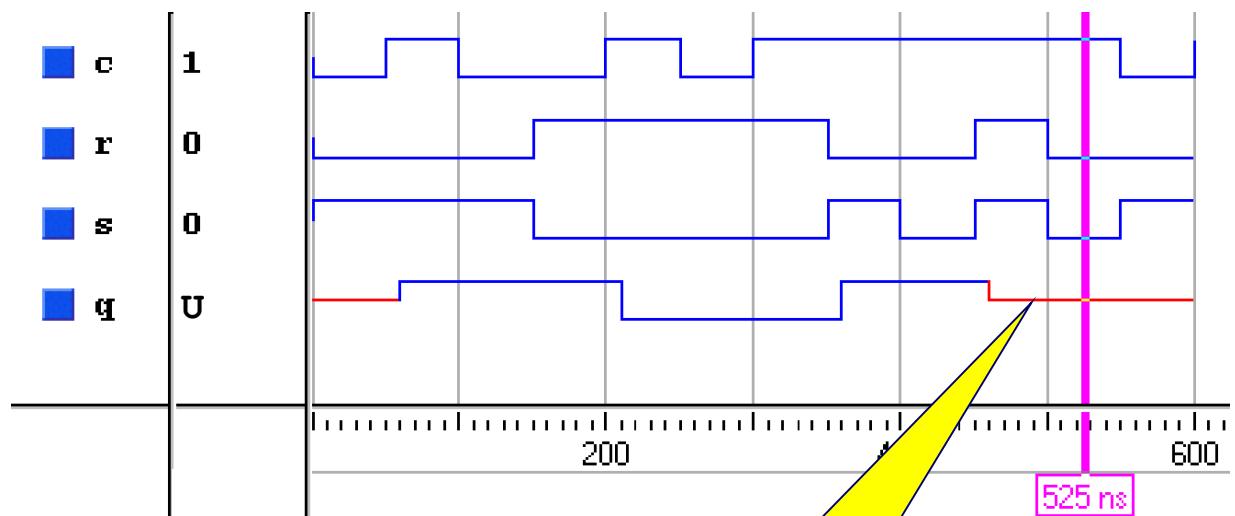
Erinnere: Das aus NAND-Gattern aufgebaute RS-Latch wird durch L-aktive Signale angesteuert !!

C	S	R	SI	RI	Q	Q ⁺	Bedeutung
0	X	X	1	1	0	0	Speichern / Takttor geschlossen
0	X	X	1	1	1	1	Speichern / Takttor geöffnet
1	0	0	1	1	0	0	Setzen
1	0	0	1	1	1	1	Löschen
1	1	0	0	1	X	1	Irregulär
1	0	1	1	0	X	0	
1	1	1	0	0	X	U	

VHDL-Prozess und Zeitverhalten

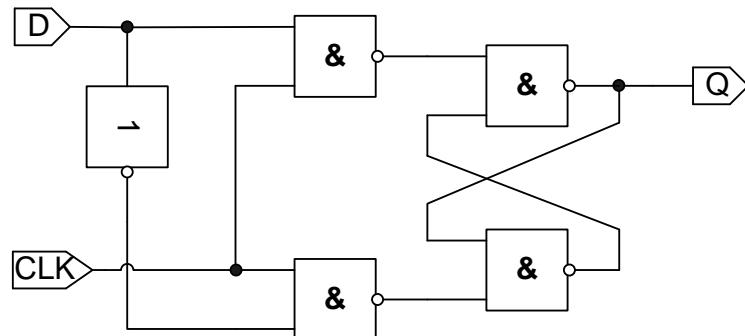
```
RS: process(C, R, S, Q)
begin
  if C = '1' then
    if  (S='1' and R='0') then Q <= '1' after 10 ns;          --Setzen
      elsif (S='0' and R='1') then Q <= '0' after 10 ns;        --Ruecksetzen
      elsif (S='0' and R='0') then Q <=  Q after 10 ns;          --Speichern
      else Q <= 'U' after 10 ns;                                --Irregulaer
    end if;
  end if;
```

Die erste if-Anweisung ist unvollständig => das Synthesewerkzeug inferriert ein D-Latch !

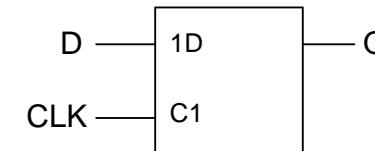


D-Latch (Data-Latch)

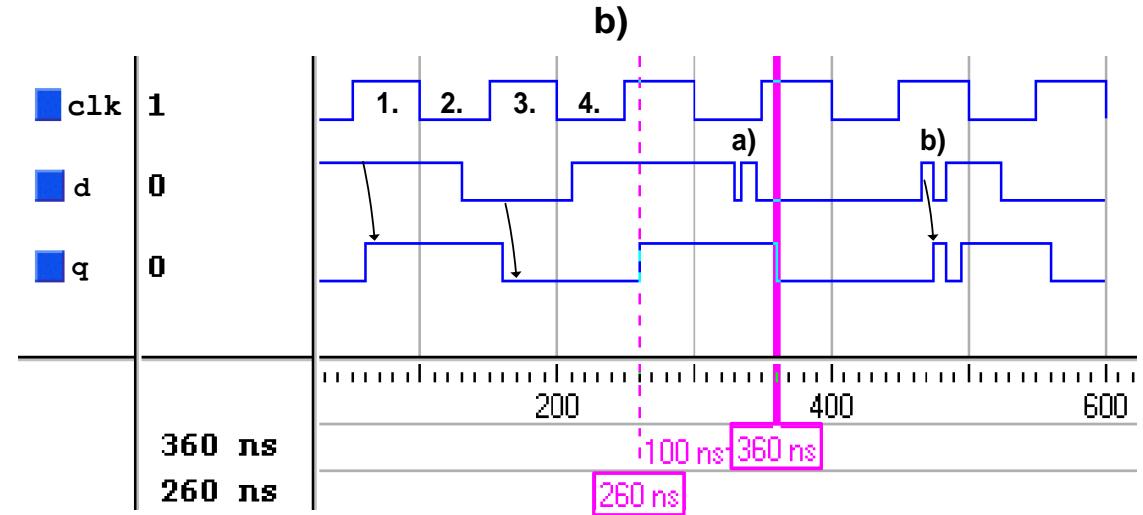
- Entsteht aus dem taktzustandsgesteuerten RS-Latch durch komplementäre Ansteuerung der R- und S-Eingänge aus einem gemeinsamen D-Eingang.
- Struktur a) und Schalsymbol b) mit Kennzeichnung der Abhängigkeit des D-Eingangs.



a)



b)



- Phasen 1 und 3: Während $\text{CLK} = 1$ wird der Wert des Dateneingangs übernommen.
- Phasen 2 und 4: Während $\text{CLK} = 0$ ist das Eingangstor geschlossen und der Wert des D-Latches bleibt unverändert.
- Eingangsstörsignale erscheinen am Ausgang, nur sofern das Takttor geöffnet ist b).

Folgezustandstabelle

Charakteristisch und nachteilig für das Verhalten des D-Latches ist sein transparentes Verhalten für $CLK = 1$, bei dem das Ausgangssignal mit geringer Verzögerung an den Ausgang weitergereicht wird. Am Eingang vorhandene Hazards werden also ebenfalls an den Ausgang durchgereicht.

- Folgezustandstabelle:

CLK	D	Q	Q^+
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

- Charakteristische Gleichung: $Q^+ = (D \wedge CLK) \vee (Q \wedge \overline{CLK})$

VHDL-Modellierung von D-Latches

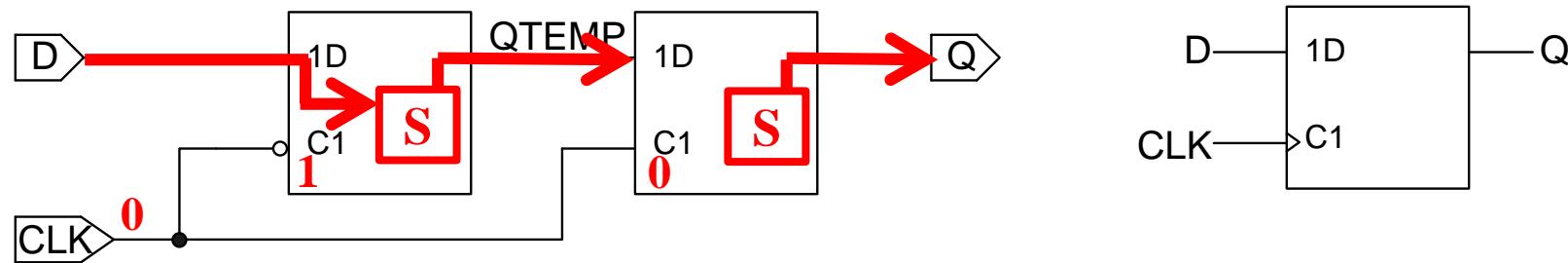
```
entity DLATCH is
    port( CLK, D : in bit;
          Q: out bit);
end DLATCH;
architecture VERHALTEN of DLATCH is
begin
process(CLK, D)
begin
    if CLK = '1' then
        Q <= D after 5 ns; --Daten übernehmen
    end if;
end process;
end VERHALTEN;
```

CLK und D müssen sich in der Sensitivityliste befinden, da beide Signale sofort einen Signalwechsel von Q bewirken können.

In VHDL wird ein Signal oder eine Variable dann zum D-Latch synthetisiert, wenn dem Signal bzw. der Variablen in einer if-Anweisung nicht in allen möglichen Verzweigungen ein Wert zugewiesen wird. Um sicher zu sein, dass ein Latch nicht versehentlich erzeugt wird, sollte allen Ausgangssignalen bzw. Variablen eines kombinatorischen Prozesses vor der ersten if-Verzweigung ein Defaultwert zugewiesen werden. Ein D-Latch wird auch synthetisiert, wenn Variable zuerst verwendet werden, bevor sie eine Wertzuweisung erfahren.

D-Flipflops (DFFs)

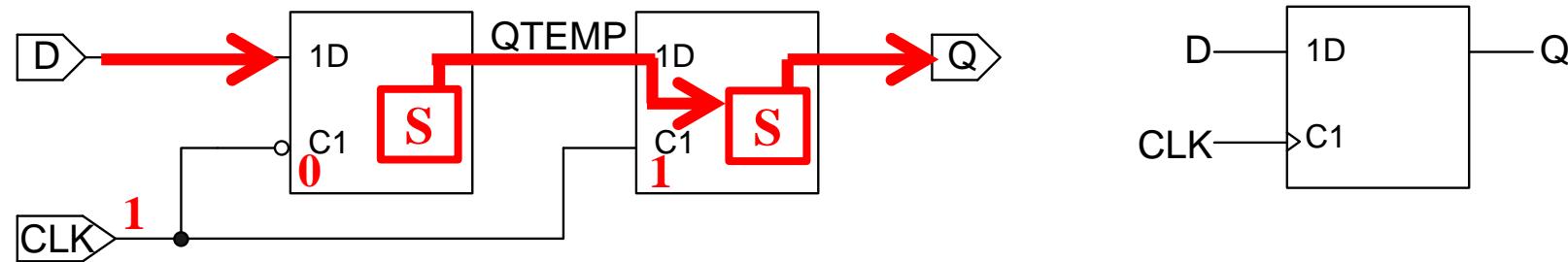
- Strukturmodell eines D-Flipflops aus zwei D-Latches und Schaltsymbol



CLK = 0

D-Flipflops (DFFs)

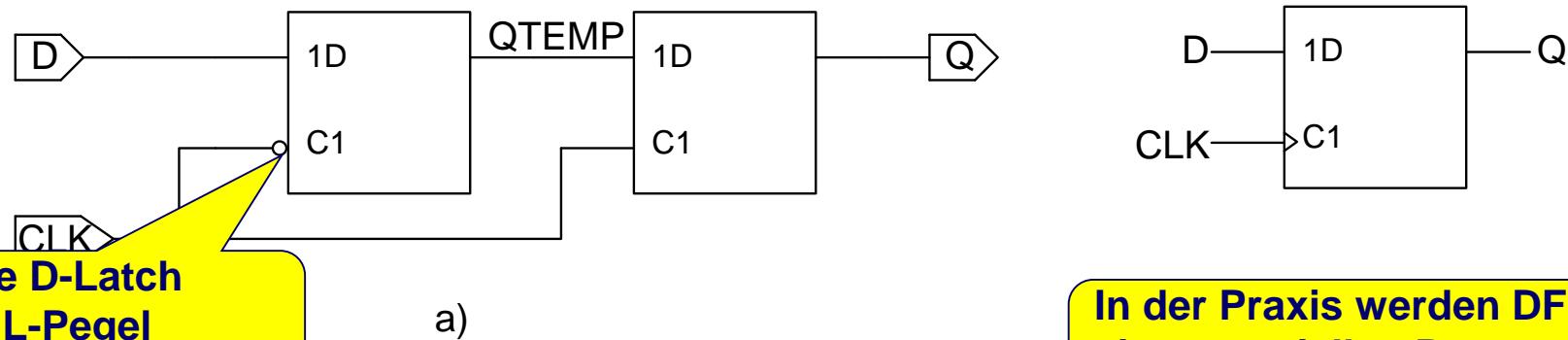
- Strukturmodell eines D-Flipflops aus zwei D-Latches und Schaltsymbol



CLK = 1

D-Flipflops (DFFs)

- Strukturmodell eines D-Flipflops aus zwei D-Latches und Schaltsymbol



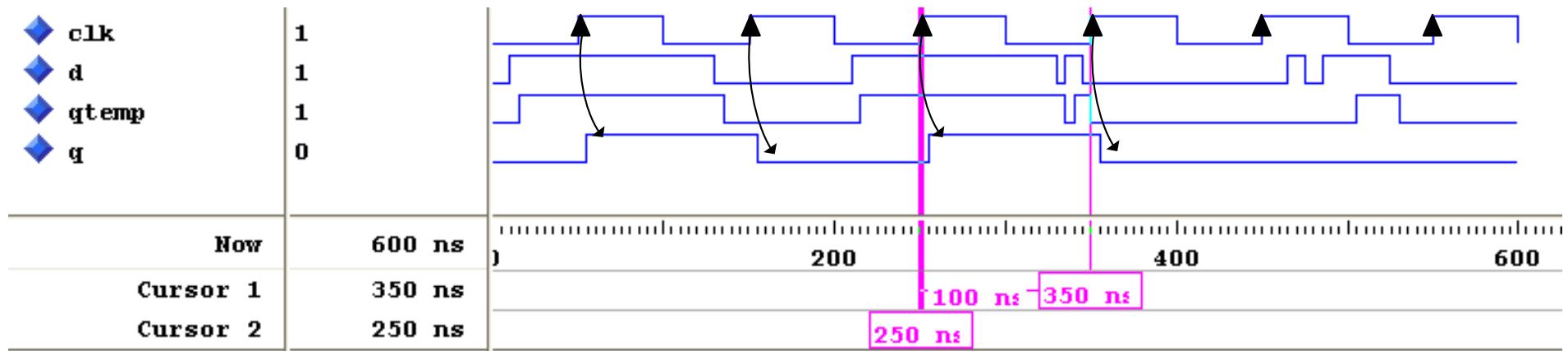
Das erste D-Latch
wird mit L-Pegel
angesteuert !

a)

In der Praxis werden DFFs mit
einer speziellen Prozesssyntax
modelliert !

```
architecture STRUKTUR of DFF_2_LATCH is
component DLATCH is
    port( CLK, D : in bit;
          Q: out bit);
end component;
signal INVCLK, QTEMP: bit;
begin
    INVCLK <= not CLK;                                -- Koppelsignale
    DLATCH1: DLATCH port map(INVCLK, D, QTEMP);     -- invertierter Takt
    DLATCH2: DLATCH port map(CLK, QTEMP, Q);         -- transparent bei CLK = 0
                                                       -- transparent bei CLK = 1
end STRUKTUR;
```

Verhalten und synthesegerechte Modellierung

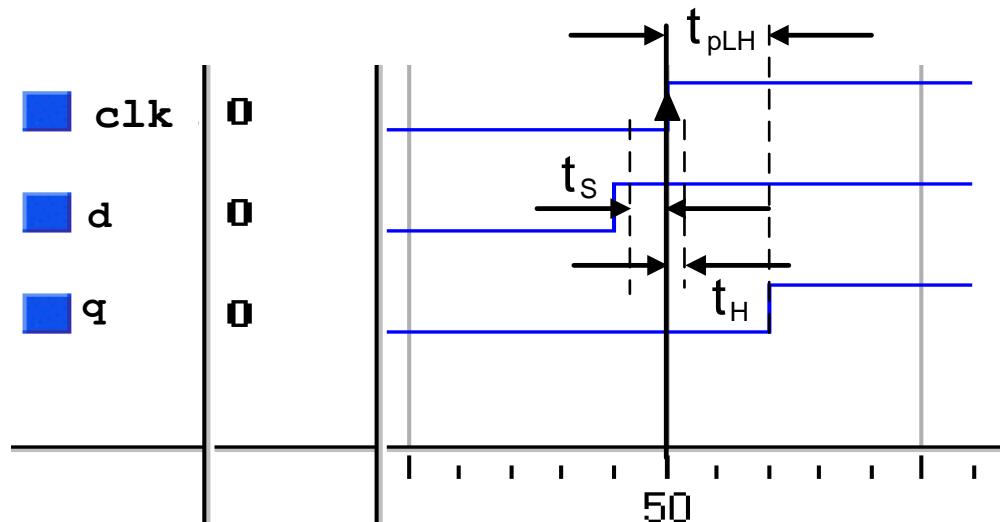


Die Wirkungspfeile zeigen, dass bei einem D-Flipflop das Eingangssignal nur während der aktiven (hier steigenden) Taktflanke ausgewertet werden muss.

```
architecture VERHALTEN of DFF is
begin
P1: process(CLK)
begin
    if CLK='1' and CLK'event then -- ansteigende Signalflanke
-- if CLK='0' and CLK'event then -- abfallende Signalflanke
        Q <= D after 5 ns;
    end if;
end process P1;
end VERHALTEN;
```

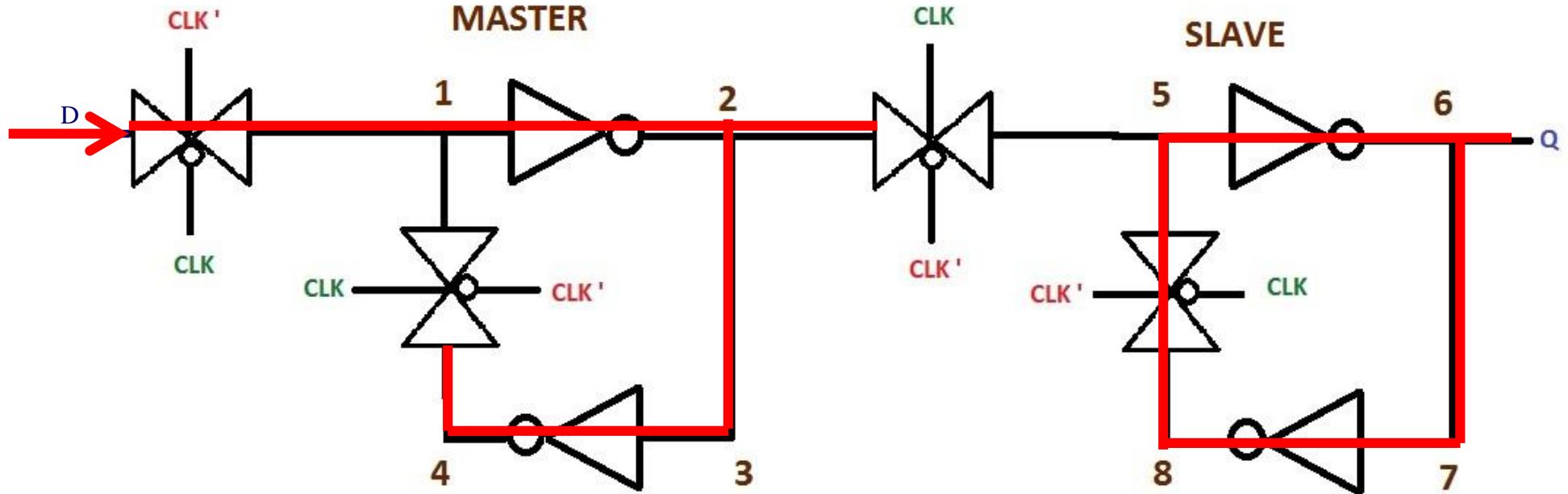
Entscheidungsintervall bei DFFs

- Was passiert, wenn das Eingangssignal während der aktiven Flanke geändert wird?
- Das Entscheidungsintervall setzt sich aus der Setup-Zeit t_s und der Hold-Zeit t_H zusammen.
- Wenn sich D während dieser Zeit ändert, so gerät der innere Aufbau des D-Flipflops ins Schwingen -> metastabiler Zustand.
- Der metastabile Zustand endet automatisch nach einer Erholungszeit t_R . Am Ende liefert der Ausgang entweder 0 oder 1 !



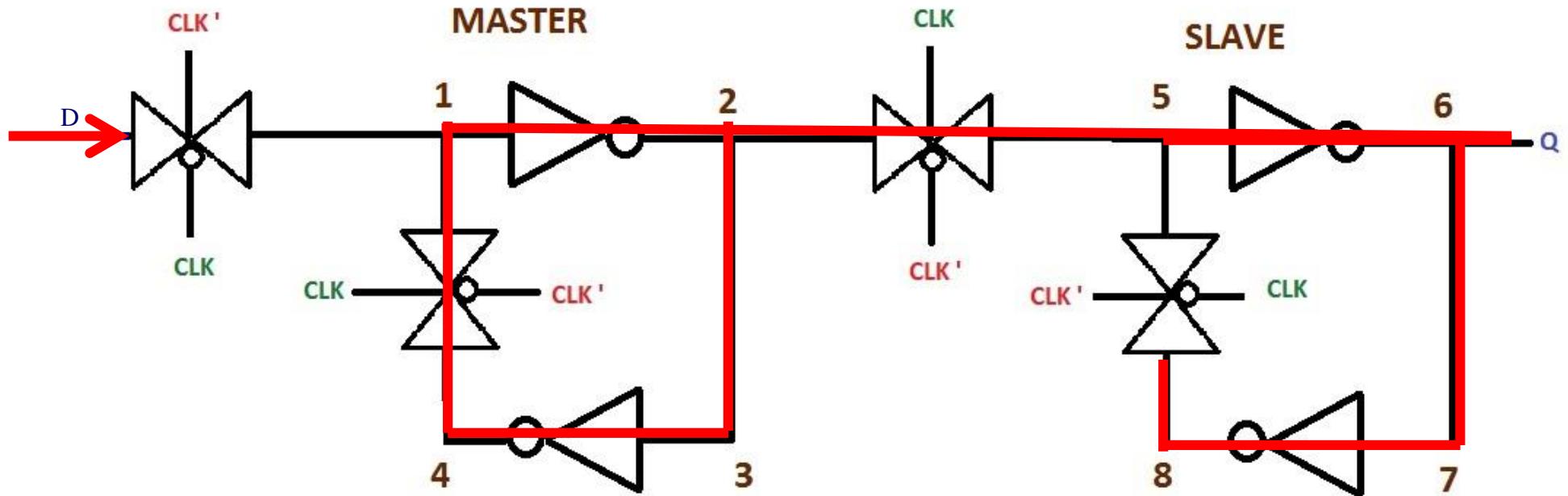
Bei getakteten Schaltungen dürfen sich die synchronen Eingangssignale nicht während des Entscheidungsintervalls nahe der aktiven Taktflanke ändern. Beim Entwurf von Testbenches wird daher empfohlen, alle Dateneingangssignale während der passiven Taktflanke (das ist meist die fallende) zu ändern.

Setup Time eines Flip-Flops



CLK = 0 _____

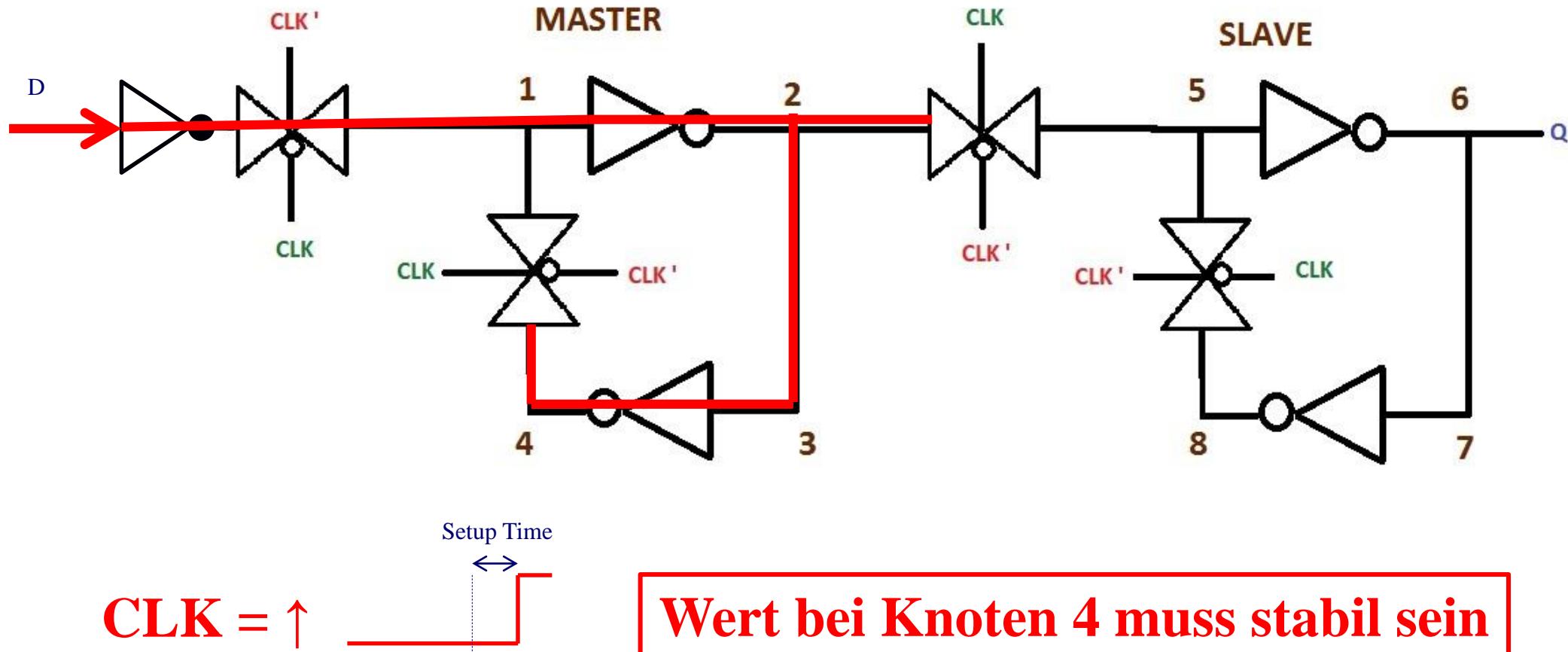
Setup Time eines Flip-Flops



CLK = 1

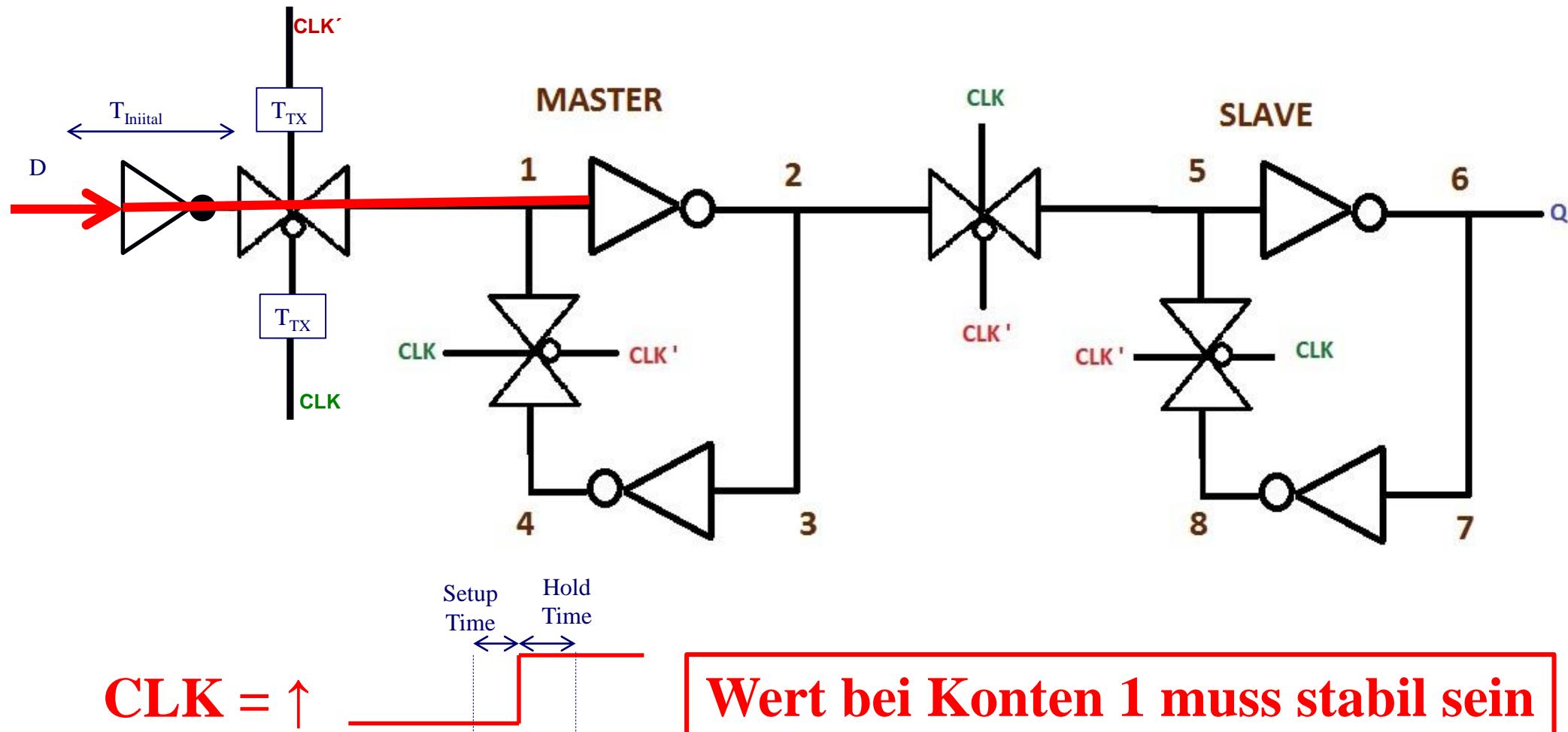


Setup Time eines Flip-Flops



Setup Time := D → 1 → 2 → 3 → 4

Hold Time eines Flip-Flops



CLK = ↑

Setup
Time

Hold
Time

Wert bei Konten 1 muss stabil sein

$$\text{Hold Time} := T_{TX} - T_{Initial}$$

Timing Analysis

Gegeben:

$$t_{su} = 0.6 \text{ ns}$$

$$t_h = 0.4 \text{ ns}$$

$$0.8 \text{ ns} \leq t_{cQ} \leq 1.0 \text{ ns}$$

$$t_{NOT} = 1 + 0.1k \text{ ns} \quad (k \text{ inputs})$$

Minimale Taktperiode:

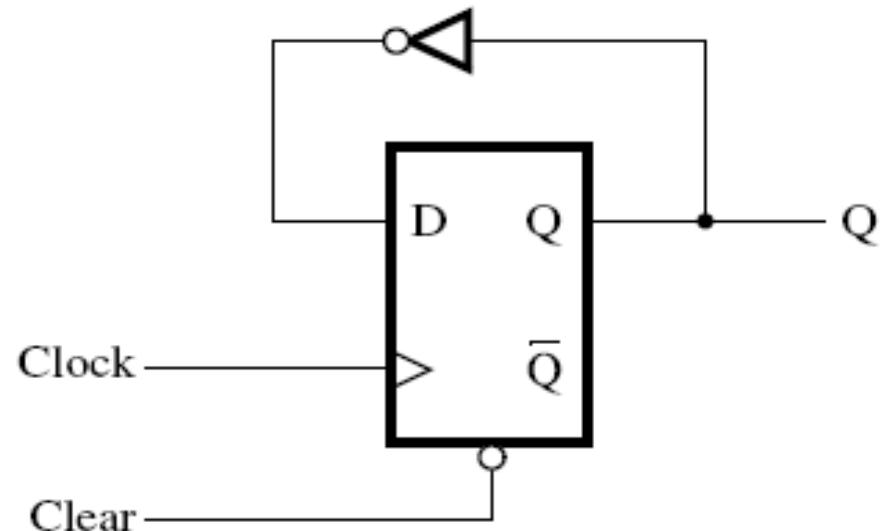
Längster Pfad zwischen FFs:

$$T_{\max} = t_{cQ} + t_{NOT}$$

$$T_{\max} + t_{su} \leq p_{clock} = 1/F_{\max}$$

$$T_{\max} + t_{su} = 1.0 + 1.1 + 0.6 = 2.7 \text{ ns}$$

$$\rightarrow F_{\max} = 1/2.7 \text{ ns} = 370.37 \text{ MHz}$$



Vermeiden von Holdtime Verletzungen:

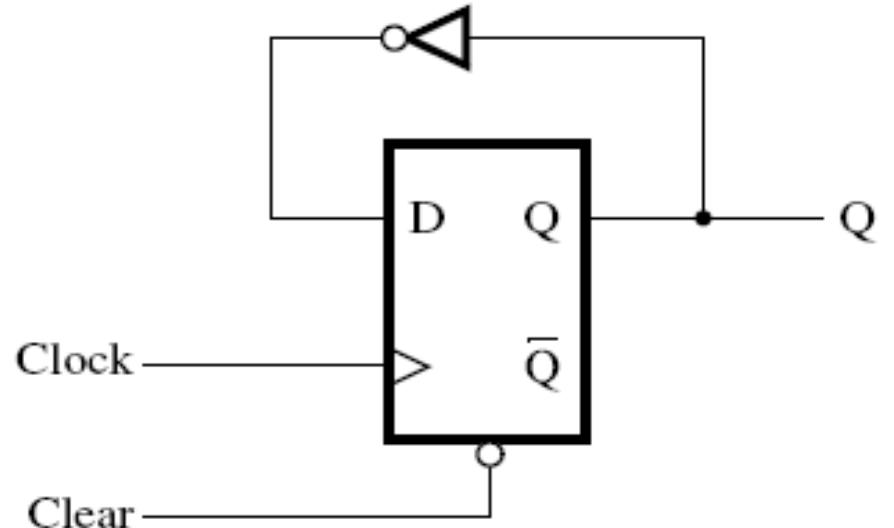
Kürzester Pfad zwischen FFs muss
länger sein als t_h :

$$T_{\min} = t_{cQ} + t_{NOT} \geq t_h$$

$$T_{\min} = 0.8 + 1.1 = 1.9 \text{ ns} \geq 0.4 \text{ ns}$$

\rightarrow keine Holdtime Verletzung

Timing Analysis



Minimale Taktperiode:

Längster Pfad zwischen FFs muss
kürzer als Taktperiode sein:

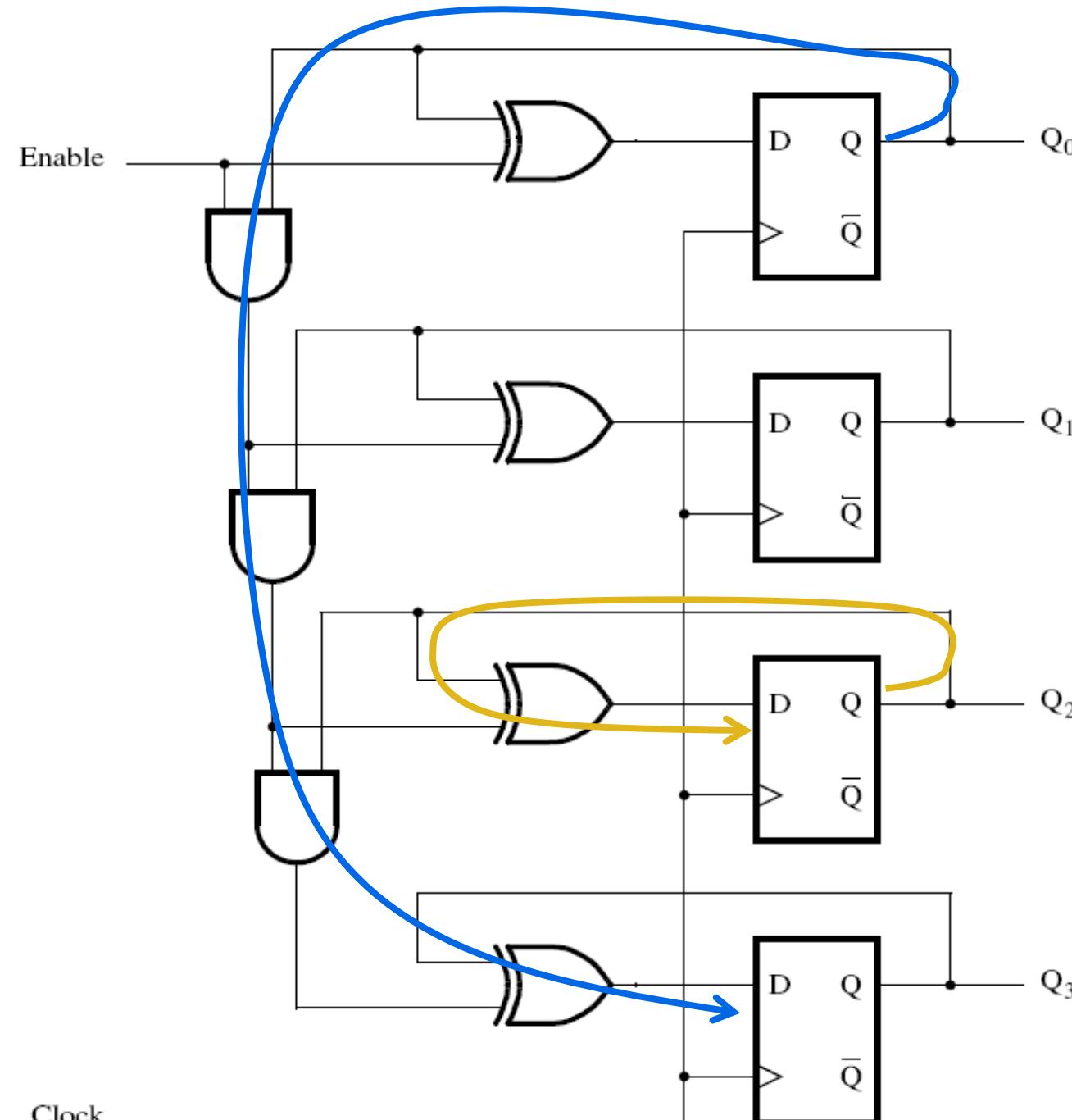
$$T_{\max} + t_{su} \leq p_{\min_clock} = 1/F_{\max}$$

Vermeiden von Holdtime Verletzungen:

Kürzester Pfad zwischen FFs muss
länger sein als t_h :

$$T_{\min} \geq t_h$$

Timing Analysis: 4 Bit Zähler



Minimal clock period:

Longest path between FFs:

$$T_{\max} = t_{cQ} + 3t_{AND} + t_{XOR}$$

$$T_{\max} + t_{su} = 1.0 + 3 \cdot 1.2 + 1.2 + 0.6 = 6.4 \text{ ns}$$

$$\rightarrow F_{\max} = 1/T_{\max} = 156.25 \text{ MHz}$$

Avoid hold time violation:

$$T_{\min} = t_{cQ} + t_{XOR} \geq t_h$$

$$T_{\min} = 0.8 + 1.2 = 2.0 \text{ ns} \geq 0.4 \text{ ns}$$

→ no hold time violation

Timing Analyse Beispiel

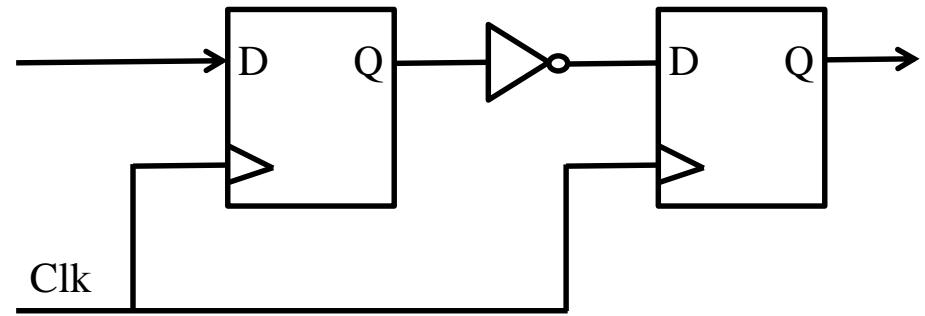
Gegeben:

$$t_{su} = 0.6 \text{ ns}$$

$$t_h = 0.4 \text{ ns}$$

$$0.8 \text{ ns} \leq t_{cQ} \leq 1.0 \text{ ns}$$

$$t_{NOT} = 1 + 0.1k \text{ ns} \quad (k \text{ inputs})$$



Minimale Taktperiode:

$$T_{\max} = t_{cQ} + t_{NOT} = 1 \text{ ns} + 1.1 \text{ ns} = 2.1 \text{ ns}$$

$$T_{\max} + t_{su} = 2.7 \text{ ns} \leq p_{clock}$$

$$F_{\max} = 370.37 \text{ MHz}$$

Vermeiden von Holdtime Verletzungen:

$$T_{\min} \geq t_h$$

$$T_{\min} = 0.8 \text{ ns} + 1.1 \text{ ns} = 1.9 \text{ ns} \geq 0.4 \text{ ns}$$

Timing Analyse Beispiel

Gegeben:

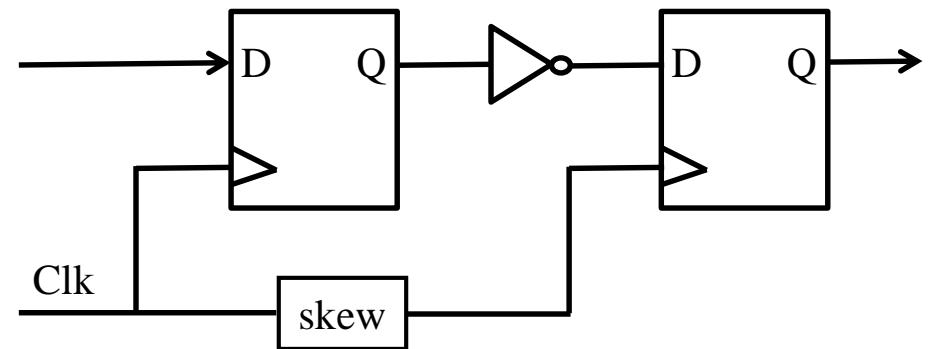
$$t_{su} = 0.6 \text{ ns}$$

$$t_h = 0.4 \text{ ns}$$

$$0.8 \text{ ns} \leq t_{cQ} \leq 1.0 \text{ ns}$$

$$t_{NOT} = 1 + 0.1k \text{ ns} \quad (k \text{ inputs})$$

$$t_{\text{skew}} = 0.6 \text{ ns}$$



Minimale Taktperiode:

$$T_{\max} + t_{su} - t_{\text{skew}} \leq p_{\text{clock}} = 1/F_{\max}$$

$$T_{\max} = t_{cQ} + t_{NOT} = 1 \text{ ns} + 1.1 \text{ ns} = 2.1 \text{ ns}$$

$$T_{\max} + t_{su} - t_{\text{skew}} = 2.1 \text{ ns} \leq p_{\text{clock}}$$

$$F_{\max} = 476.19 \text{ MHz}$$

Vermeiden von Holdtime Verletzungen:

$$T_{\min} - t_{\text{skew}} \geq t_h$$

$$T_{\min} = 0.8 \text{ ns} + 1.1 \text{ ns} = 1.9 \text{ ns}$$

$$T_{\min} - t_{\text{skew}} = 1.3 \text{ ns} \geq 0.4 \text{ ns}$$

Timing Analyse Beispiel

Gegeben:

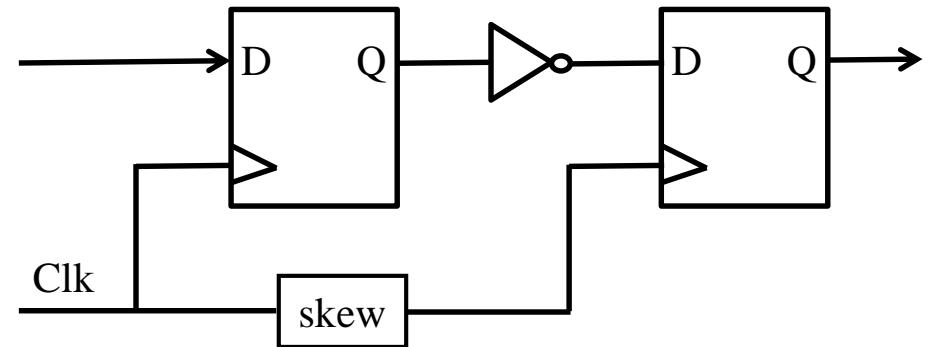
$$t_{su} = 0.6 \text{ ns}$$

$$t_h = 0.4 \text{ ns}$$

$$0.8 \text{ ns} \leq t_{cQ} \leq 1.0 \text{ ns}$$

$$t_{NOT} = 1 + 0.1k \text{ ns} \quad (k \text{ inputs})$$

$$t_{\text{skew}} = -0.6 \text{ ns}$$



Minimale Taktperiode:

$$T_{\max} + t_{su} - t_{\text{skew}} \leq p_{\text{clock}} = 1/F_{\max}$$

$$T_{\max} = t_{cQ} + t_{NOT} = 1 \text{ ns} + 1.1 \text{ ns} = 2.1 \text{ ns}$$

$$T_{\max} + t_{su} - t_{\text{skew}} = 3.3 \text{ ns} \leq p_{\text{clock}}$$

$$F_{\max} = 303.03 \text{ MHz}$$

Vermeiden von Holdtime Verletzungen:

$$T_{\min} - t_{\text{skew}} \geq t_h$$

$$T_{\min} = 0.8 \text{ ns} + 1.1 \text{ ns} = 1.9 \text{ ns}$$

$$T_{\min} - t_{\text{skew}} = 2.5 \text{ ns} \geq 0.4 \text{ ns}$$

Timing Analyse Beispiel

Gegeben:

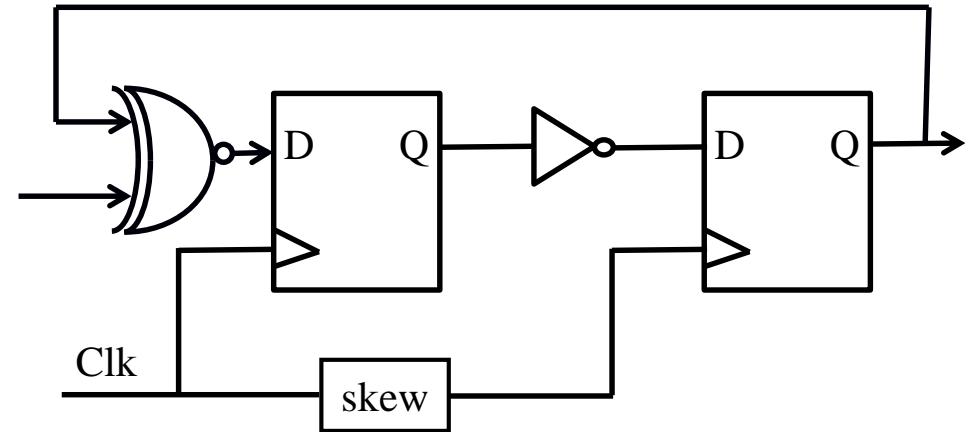
$$t_{su} = 0.6 \text{ ns}$$

$$t_h = 0.4 \text{ ns}$$

$$0.8 \text{ ns} \leq t_{cQ} \leq 1.0 \text{ ns}$$

$$t_G = 1 + 0.1k \text{ ns} \quad (k \text{ inputs})$$

$$t_{\text{skew}} = 0.6 \text{ ns}$$



Minimale Taktperiode:

$$p_1 = t_{cQ} + t_{not} + t_{su} - t_{\text{skew}} = 2.1 \text{ ns}$$

$$p_2 = t_{cQ} + t_{xnor} + t_{su} - (-t_{\text{skew}}) = 3.4 \text{ ns}$$

$$p_{clock} = p_2 = 3.4 \text{ ns}$$

$$F_{\max} = 294 \text{ MHz}$$

$$p_{clock}(t_{\text{skew}}) = 2.8 \text{ ns} - (-t_{\text{skew}})$$

Vermeiden von Holdtime Verletzungen:

$$T_{\min} - t_{\text{skew}} \geq t_h$$

$$T_1 = t_{cQ} + t_{not} - t_{\text{skew}} = 1.3 \text{ ns}$$

$$T_2 = t_{cQ} + t_{xnor} - (-t_{\text{skew}}) = 2.6 \text{ ns}$$

$$T_1, T_2 \geq 0.4 \text{ ns}$$

$$t_{\text{skew}} \leq 1.5 \text{ ns}$$

Setup-Time Überprüfung

- Überprüfung der Setup-Time durch eine assertion:

```
entity DFF_CHECK is
    generic(TS:time := 5 ns);
    port( CLK, D : in bit;
          Q : out bit);
end DFF_CHECK;
architecture VERHALTEN of DFF_CHECK is
begin
P1: process(CLK)
begin
    if CLK='1' and CLK'event then -- ansteigende Signalflanke
        -- synthesis off
        assert D'quiet(TS) report "DFF Setup Time Fehler";
        -- synthesis on
        Q <= D after 10 ns;
    end if;
end process P1;
end VERHALTEN;
```

Teile des Codes werden durch Synthese-Pragmas von der Synthese ausgeschlossen.

Mit dem 'quiet-Attribut wird überprüft, ob sich das Signal D während der Zeit TS geändert hat.

Meldung auf der Simulatorkonsole:

```
# ** Error: DFF Setup Time Fehler
#      Time: 150 ns  Iteration: 0  Instance: /dff_check
```

Setup- und Hold-Time Überprüfung

- Überprüfung der Setup-Time und Hold-Time durch eine assertion:

```
architecture behavioral of setup_hold_check is
  signal CLKTMP : bit;
begin
  p_check_setup_time : process (CLK, DATA)
  begin
    if CLK='1' and CLK'event then
      assert DATA'quiet(g_setup_time)
        report "Setup time violation" severity error;
    end if;
  end process;

  CLKTMP <= CLK'delayed(g_hold_time);
  p_check_hold_time : process (CLKTMP)
  begin
    if CLKTMP='1' and CLKTMP'event then
      assert DATA'last_event > g_hold_time
        report "Hold time violation" severity error;
    end if;
  end process;
end behavioral;
```

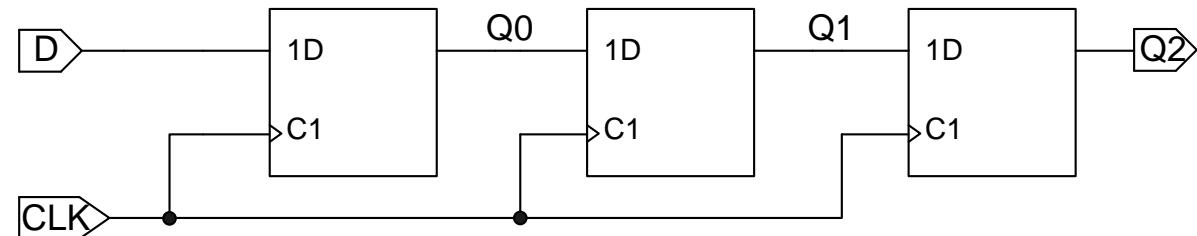
„Beschattung“ des Clocksignals.

Attribut, wann das Signal sich verändert hat.

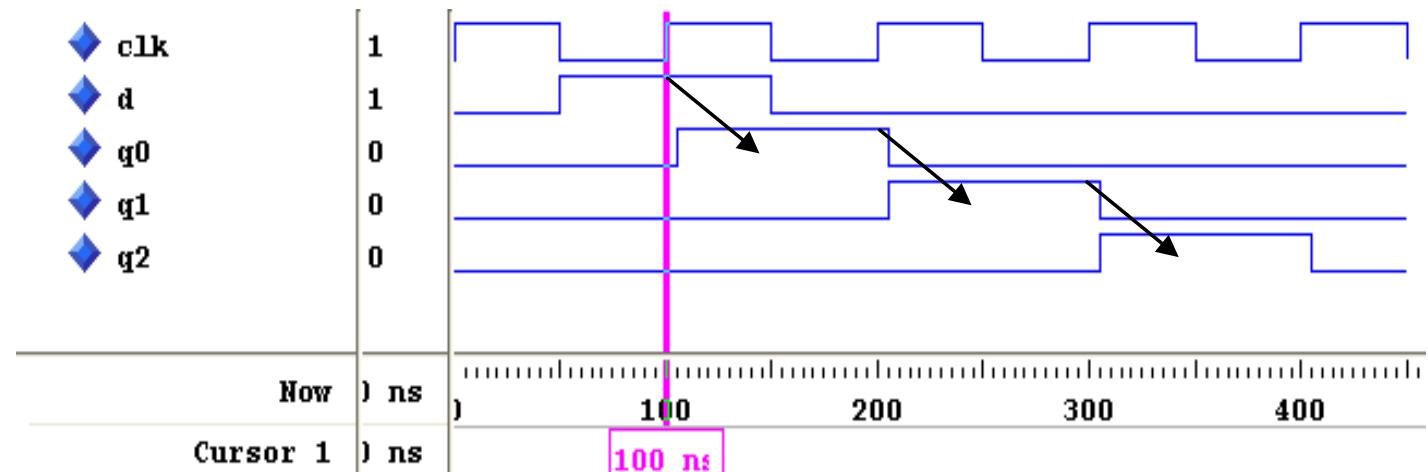
Test, ob sich DATA Signal im hold-time Interval geändert hat.

Zum Pipelineprinzip in Schieberegistern

- Schieberegisterschaltung → Flipflops in Reihe:

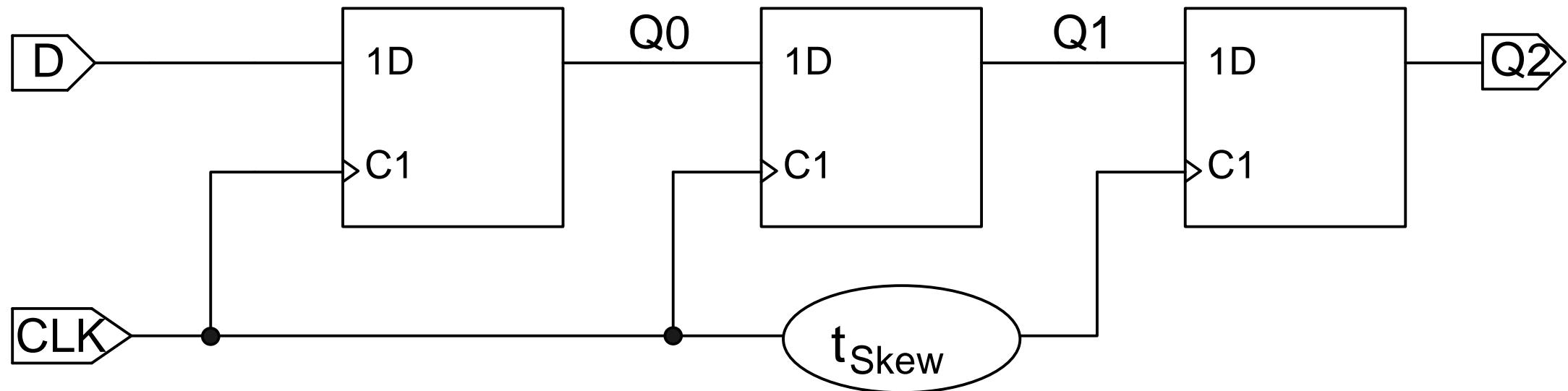


- Der Eingangsimpuls**
 $D=1$ wird bei jeder Flanke einer DFF-Stufe weitergeleitet.
- Diese Pipeline-Funktion erfordert,**
dass es keine Hold-Zeit Verletzungen gibt.



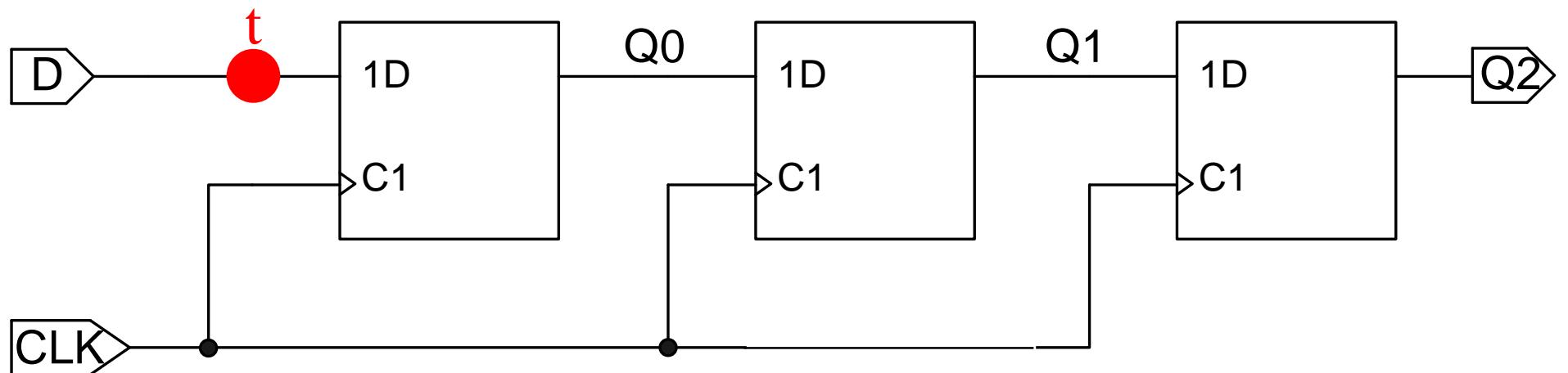
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



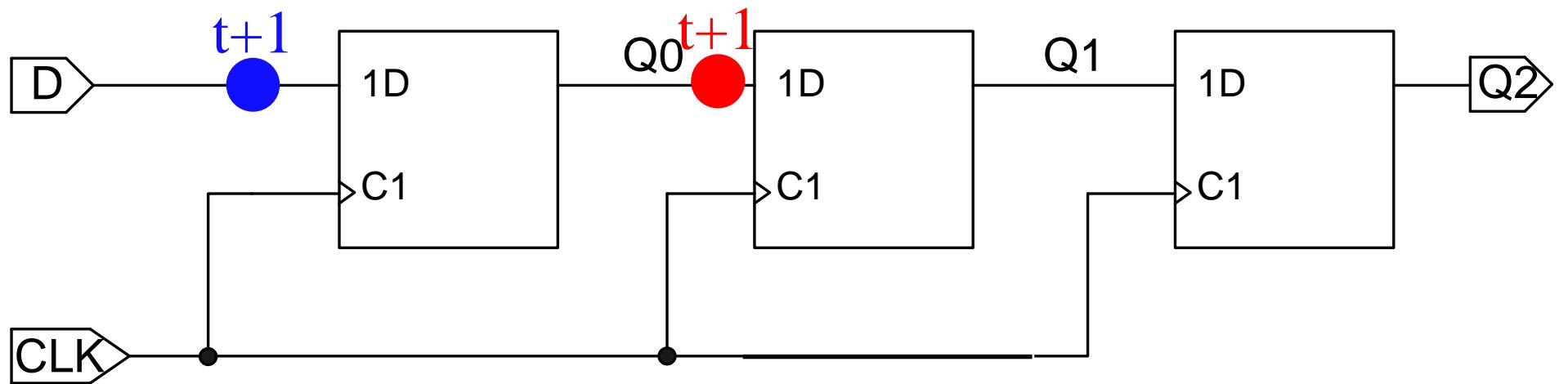
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



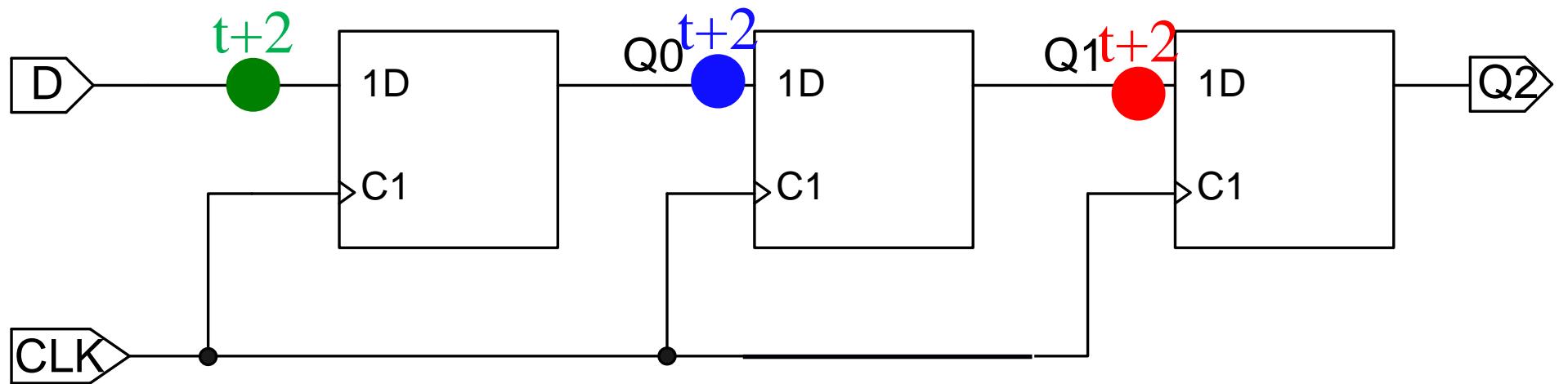
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



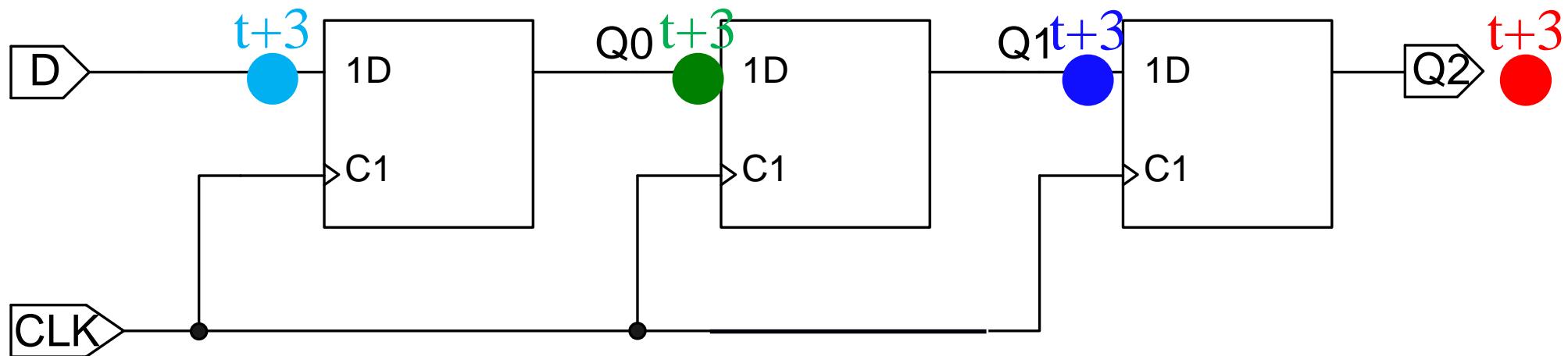
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



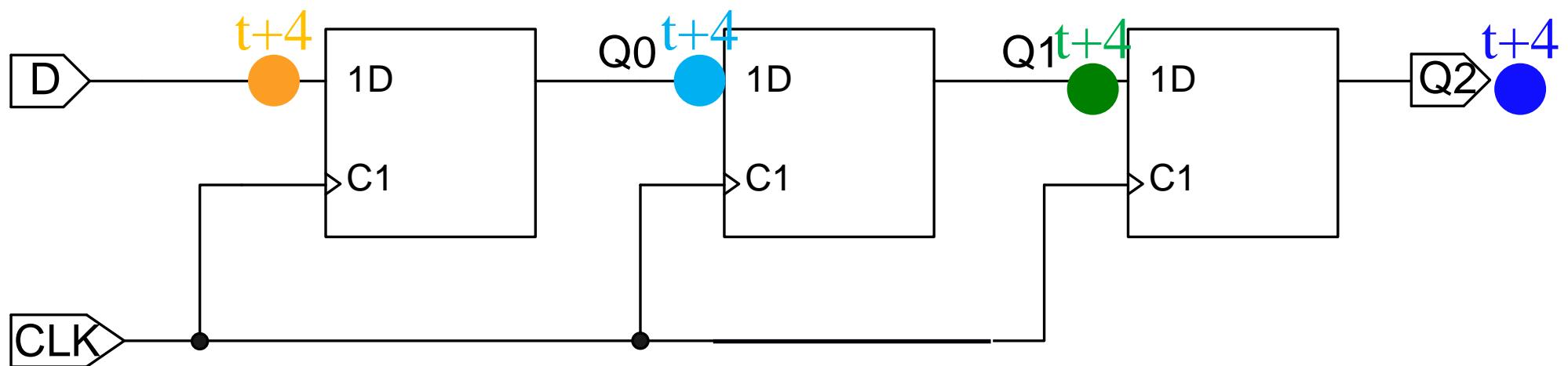
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



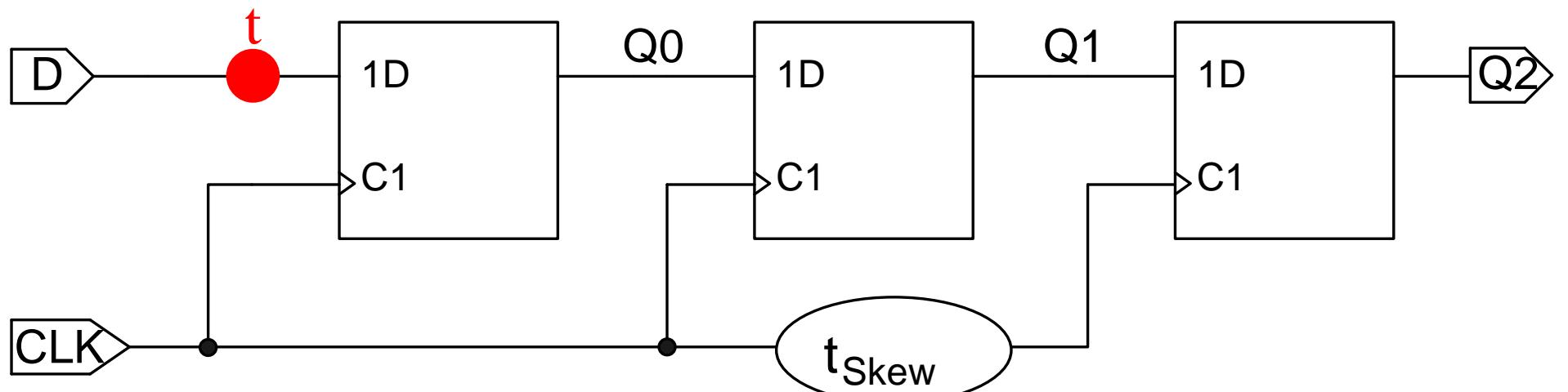
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



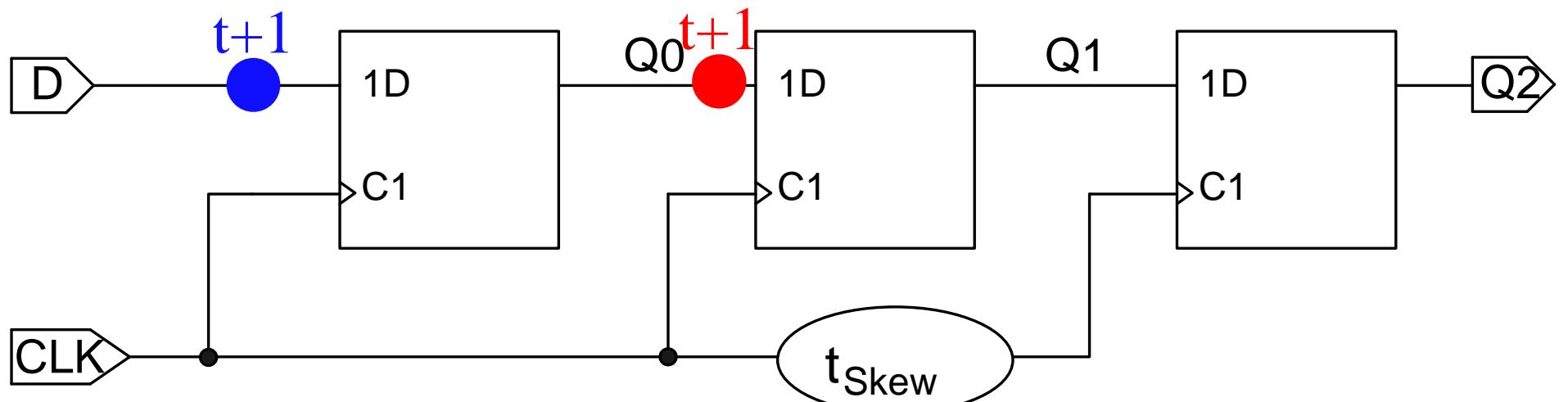
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



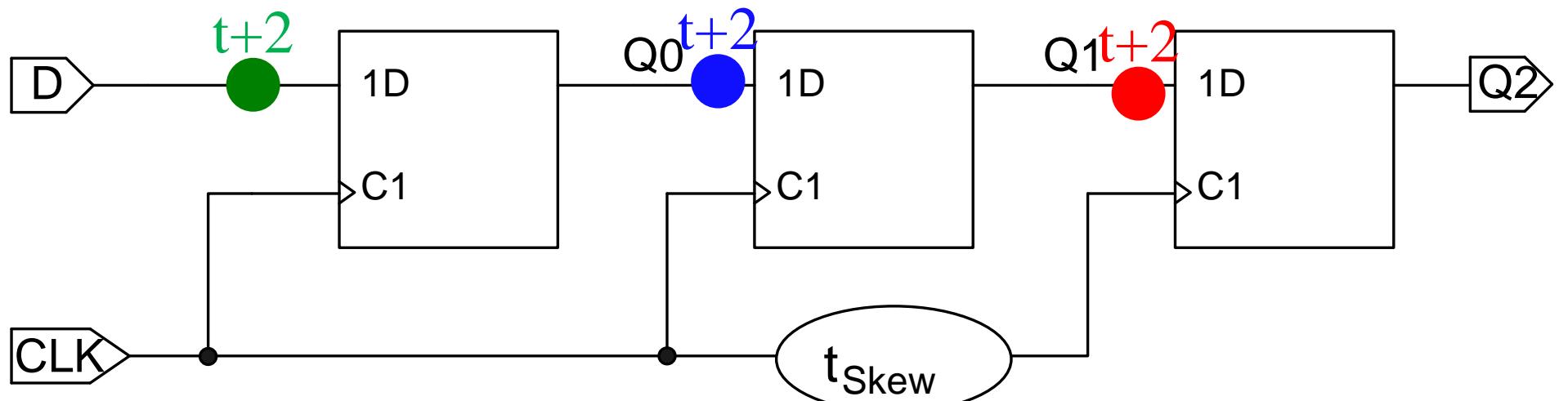
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



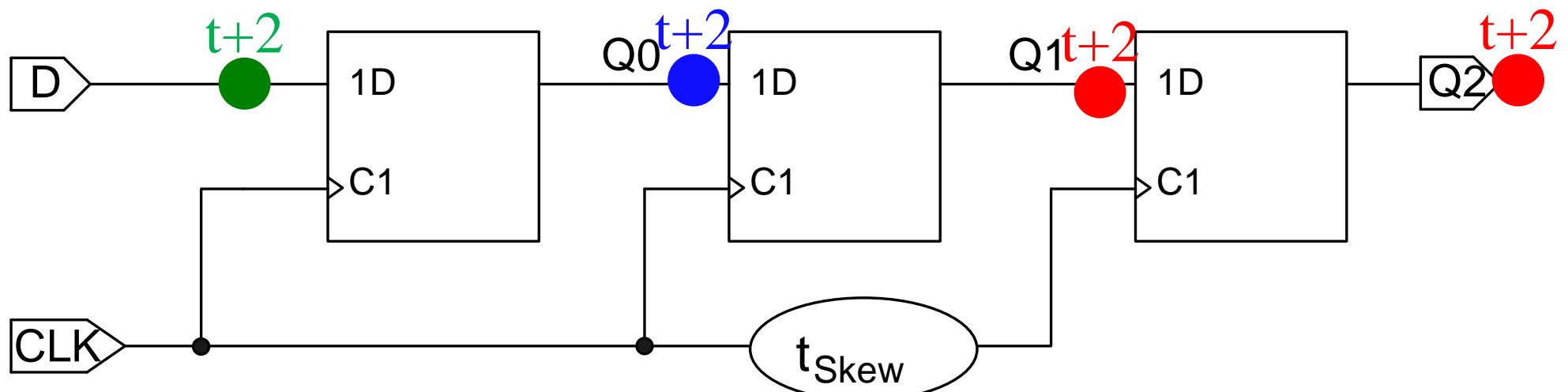
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



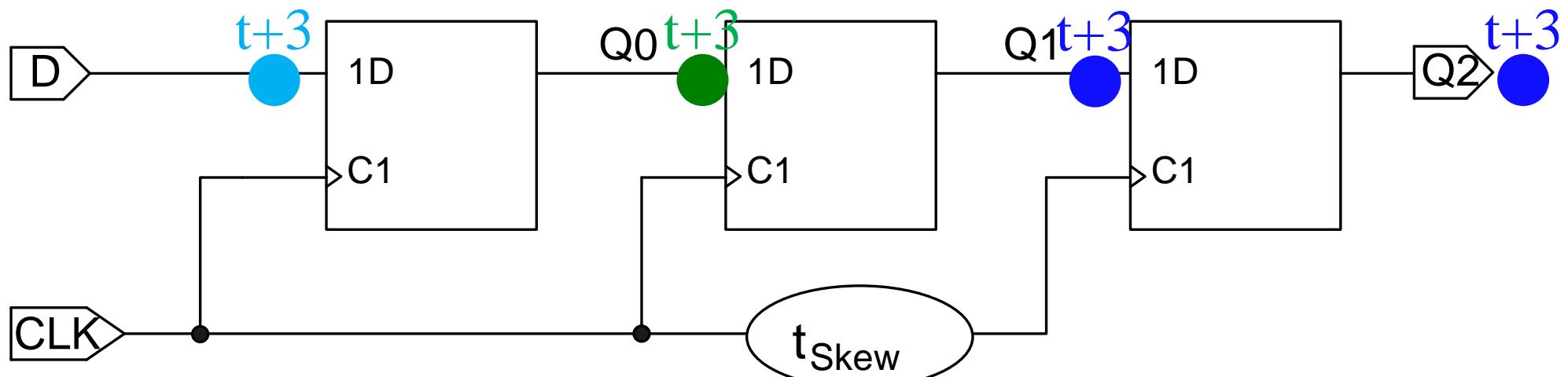
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



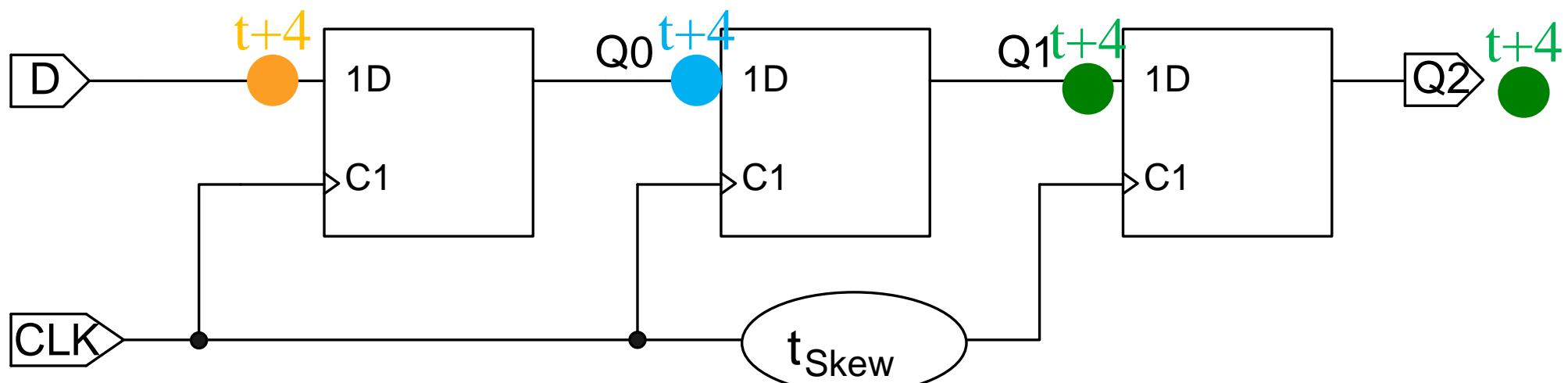
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



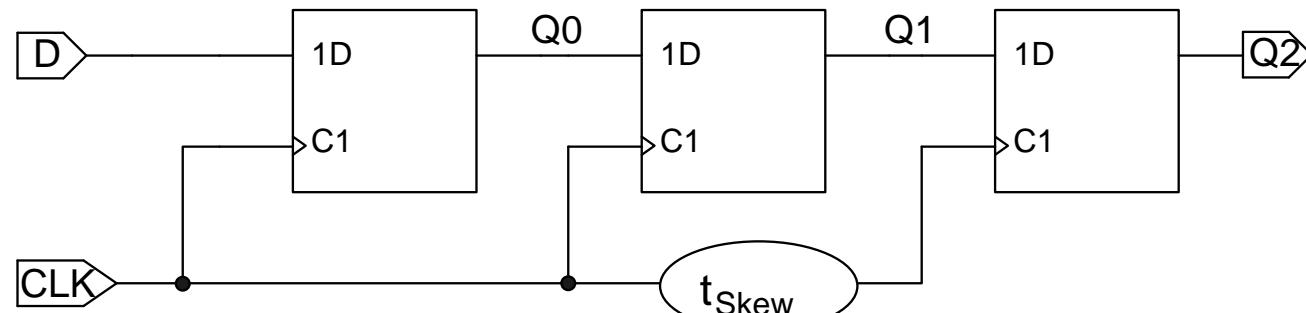
Zweispeicher-Flipflops

- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:

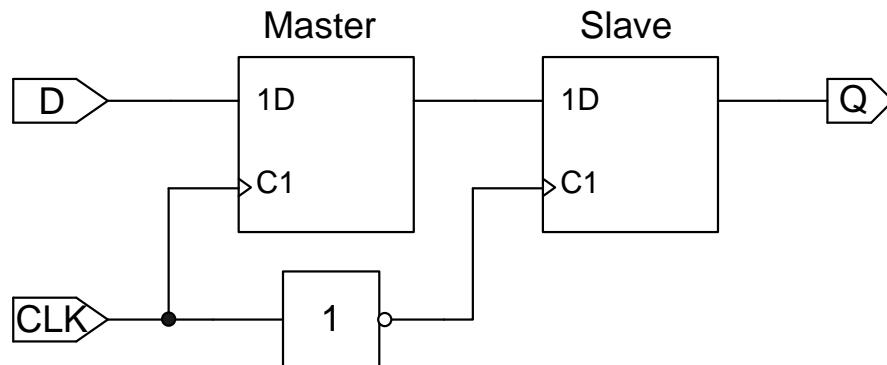


Zweispeicher-Flipflops

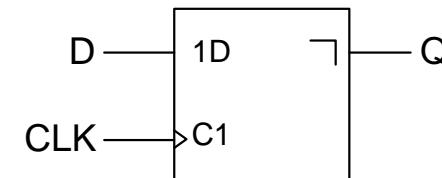
- Problem: (unvermeidbarer) Clock-Skew auf einem Board verhindert, dass alle DFFs einer Pipelinestruktur (Schieberegister) zum gleichen Zeitpunkt angesteuert werden:



- Abhilfe: Zweispeicher-(Master-Slave)-FF:
 - innerer Aufbau a) aus zwei gegenphasig angesteuerten DFFs
 - Schaltsymbol b)



a)

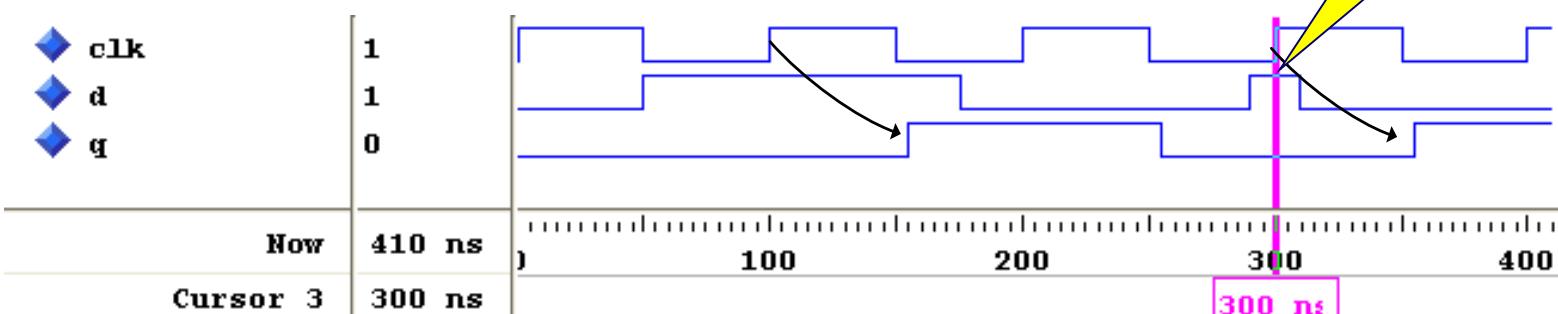


b)

VHDL-Modell und Simulation eines Zweispeicher-FFs

```
architecture VERHALTEN of MS_DFF is
signal TEMP: bit;
begin
P1: process(CLK)
begin
  if CLK='1' and CLK'event then -- steigende Signalflanke
    TEMP <= D;
  end if;
end process P1;
P2: process(CLK)
begin
  if CLK='0' and CLK'event then -- fallende Signalflanke
    Q <= TEMP after 5 ns;
  end if;
end process P2;
end VERHALTEN;
```

Das Eingangssignal wird bei steigender Flanke eingelesen, aber erscheint bei fallender Flanke am Ausgang.



Varianten von D-Flipflops

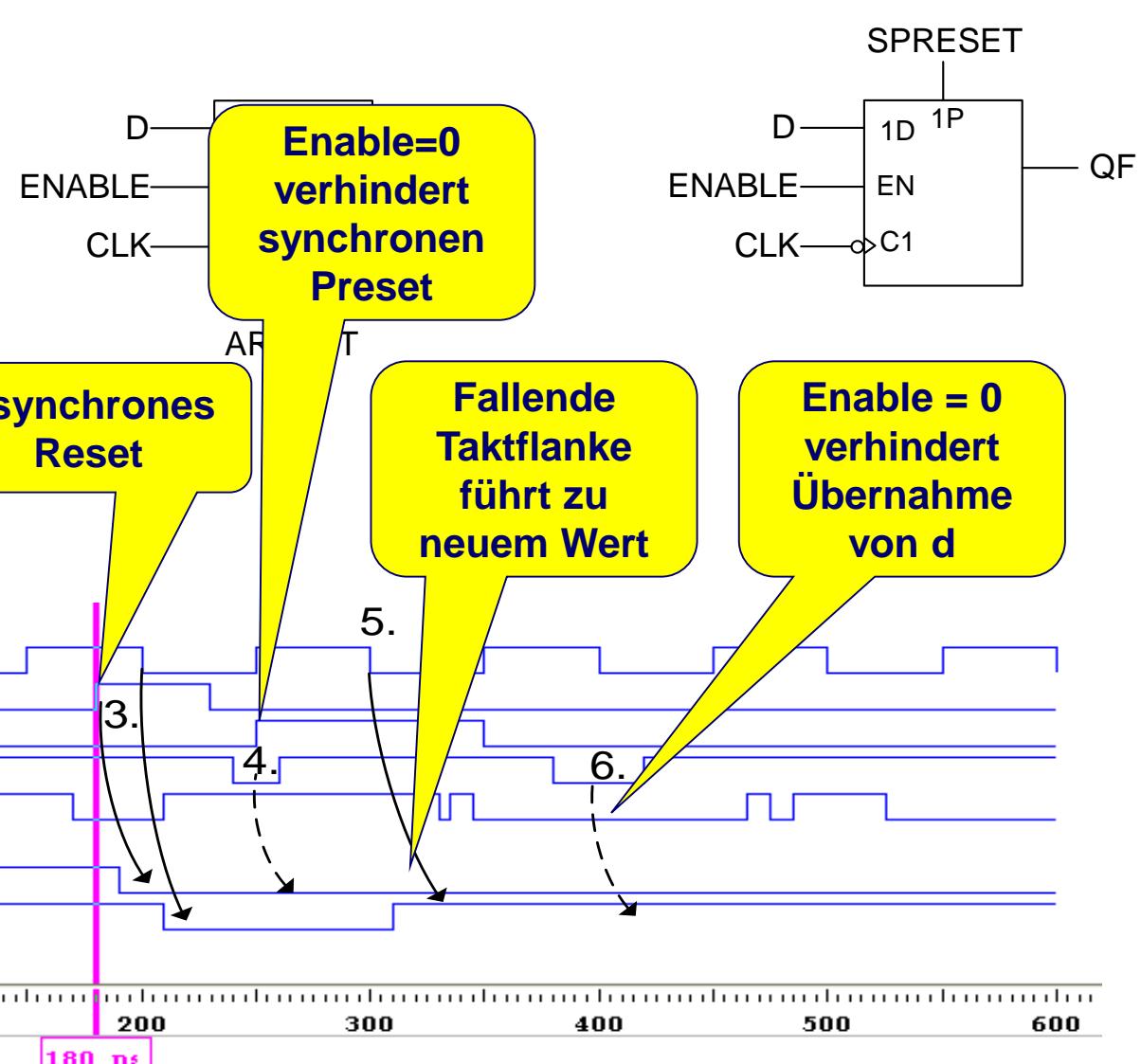
- Typische DFF-Varianten:
 - Asynchroner Reset- oder Preset-Eingang, der dafür sorgt, dass das Ausgangssignal sofort (asynchron) auf 0 (Reset) bzw. 1 (Preset) geht.
 - Freigabeeingang EN mit dem das Einlesen des Datensignals trotz aktiver Taktflanke verhindert werden kann, wenn EN nicht zuvor auf 1 gesetzt wurde.
 - Synchroner Reset- oder Preset-Eingang. Damit wird erreicht, dass die Reset- bzw. Preset-Funktion erst bei der aktiven Taktflanke erfolgt, wenn zuvor das Reset- bzw. Preset-Steuersignal gesetzt wurde.
- Synchrone Steuersignale (wie z.B. EN, SRESET) werden auch als Vorbereitungssignale bezeichnet.

Zur Vermeidung von Überraschungen bei der VHDL-Synthese wird dringend empfohlen, die in der Norm IEEE-1076.6 standardisierten Syntheserichtlinien einzuhalten, bzw. die in den Listings vorgestellten Flipflop-Entwurfsmuster zu verwenden.

Modellierungsbeispiele

a) ist ein DFF mit asynchronem Reset und Freigabeeingang, dessen aktive Flanke die steigende ist. Es wird durch einen Prozess R_EDGE (s.u.) modelliert, der das Ausgangssignal QR definiert.

b) ist ein DFF mit synchronem Reset und Freigabeeingang, welche bei fallender Flanke operiert. Das Flipflop wird durch einen Prozess F_EDGE und das Signal SPRESET QF modelliert.



VHDL-Code zu den Modellierungsbeispielen

```
entity DFLIPFLOPS is
    port( CLK, D, ARESET, SPRESET, ENABLE : in bit;
          QR, QF: out bit);           -- Zwei Flipfloptypen
end DFLIPFLOPS;
architecture VERHALTEN of DFLIPFLOPS is
begin
R_EDGE: process(CLK, ARESET)           -- 1. FF-Variante
begin
    if ARESET='1' then QR <= '0' after 10 ns; -- asynchroner Reset
    elsif (CLK='1' and CLK'event) then        -- ansteigende Flanke
        if ENABLE = '1' then QR <= D after 10 ns; -- Freigabe
        end if;
    end if;
end process R_EDGE;
F_EDGE: process(CLK)                  -- 2. FF-Variante
begin
    if (CLK='0' and CLK'event) then          -- abfallende Flanke
        if SPRESET='1' then QF <= '1' after 10 ns; -- synchroner Preset
        elsif ENABLE = '1' then QF <= D after 10 ns; -- Freigabe
        end if;
    end if;
end process F_EDGE;
end VERHALTEN;
```

VHDL-Entwurfsrichtlinien für getaktete Schaltungen

- Getaktete Schaltungselemente werden durch Prozesse modelliert. In der Sensitivityliste müssen sich das Taktsignal sowie alle eventuell vorhandenen **asynchronen** Steuersignale befinden.
- Durch die Klammer `if CLK='0' | '1' and CLK'event ... end if` wird ein taktsynchroner Rahmen definiert.
- Alle Signale denen im taktsynchronen Rahmen ein Wert zugewiesen wird, werden zu Flipflops bzw. Registern synthetisiert.
- Eventuell vorhandene asynchrone Steuersignale (hier ARESET) müssen vor der Taktflankenabfrage abgefragt werden.
- Signalen oder Variablen, denen in einem taktsynchronen Rahmen ein Wert zugewiesen wurde, darf nachträglich kein Signalwert mehr zugewiesen werden. Dies impliziert, dass die if-Anweisung des getakteten Rahmens die letzte innerhalb des Prozesses sein muss.
- Variable werden heraus optimiert oder als kombinatorische Logik synthetisiert, wenn sichergestellt ist, dass ihnen innerhalb des taktsynchronen Rahmens auf allen Pfaden durch den Prozess ein Wert zugewiesen wird, *bevor* die Variable gelesen oder im Bedingungsausdruck einer case- oder if-Anweisung verwendet wird. Andernfalls werden dafür ebenfalls Flipflops synthetisiert.
- Innerhalb des taktsynchronen Rahmens verwendete if-Anweisungen dürfen unvollständig modelliert sein, ohne dass dadurch ein zusätzliches D-Latch erzeugt wird.

Taktflankengesteuertes JK-Flipflop

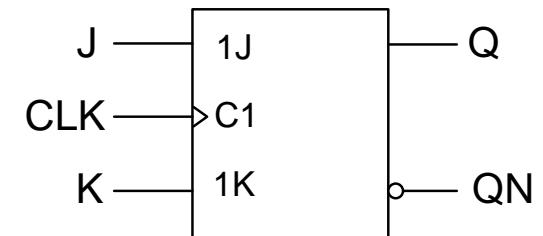
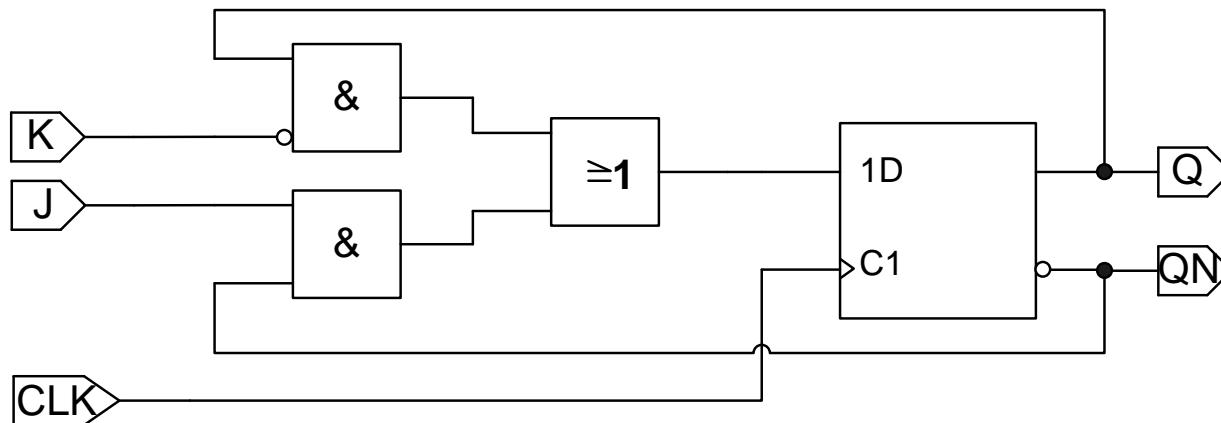
- Bei steigender Flanke funktioniert das JK-Flipflop ähnlich wie ein RS-Flipflop (J entspricht S und K entspricht R).
- Aber es toggelt für $J = K = 1$.
- Arbeitstabelle und charakteristische Gleichung:

CLK	J	K	Q^+
0	X	X	Q
1	X	X	Q
↑	0	0	Q
↑	0	1	0
↑	1	0	1
↑	1	1	\overline{Q}

$$Q^+ = (J \wedge \overline{Q}) \vee (\overline{K} \wedge Q)$$

Innerer Aufbau eines JK-FFs

- Innerer Aufbau mit DFF a) und Schalsymbol b)



a)

CLK	J	K	Q^+
0	X	X	Q
1	X	X	Q
\rightarrow	0	0	Q
\rightarrow	0	1	0
\rightarrow	1	0	1
\rightarrow	1	1	\overline{Q}

b)

$$Q^+ = (J \wedge \overline{Q}) \vee (\overline{K} \wedge Q)$$

T(oggle)-Flipflop

- Für $T=1$ invertiert der Ausgang, für $T=0$ ändert sich der Ausgang nicht
- Mit TFFs lassen sich im Vergleich zu DFFs beim Entwurf zyklischer Zähler Hardwareressourcen einsparen.
- Arbeitstabelle und charakteristische Gleichung:

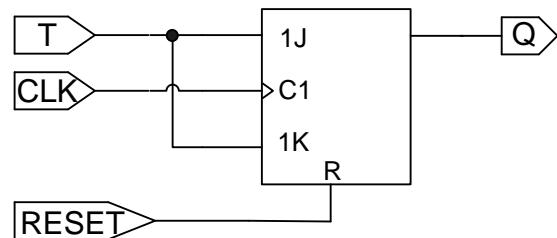
CLK	T	Q^+
0	X	Q
1	X	Q
↓	0	Q
↓	1	\overline{Q}

$$Q^+ = (\overline{Q} \wedge T) \vee (Q \wedge \overline{T})$$

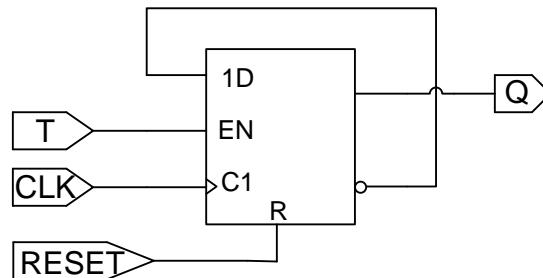
bzw.

$$Q^+ = \overline{Q} \text{ für } T = 1$$

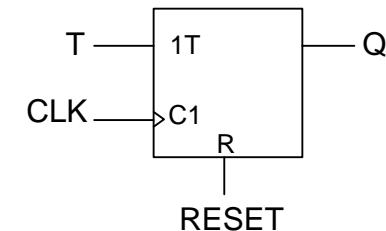
Innerer Aufbau eines TFFs und VHDL-Modell



TFF aufgebaut aus JK-FF



Basierend auf DFF



Schaltsymbol

VHDL-Modell zu DFF
basierendem TFF

```
architecture VERHALTEN of TFF is
signal QTEMP: bit;
begin
P1: process(CLK, RESET)
begin
  if RESET = '1' then
    QTEMP <= '0' after 5 ns;
  elsif CLK='1' and CLK'event then
    if T = '1' then
      QTEMP <= not QTEMP after 5 ns;
    end if;
  end if;
end process P1;
Q <= QTEMP;
end VERHALTEN;
```

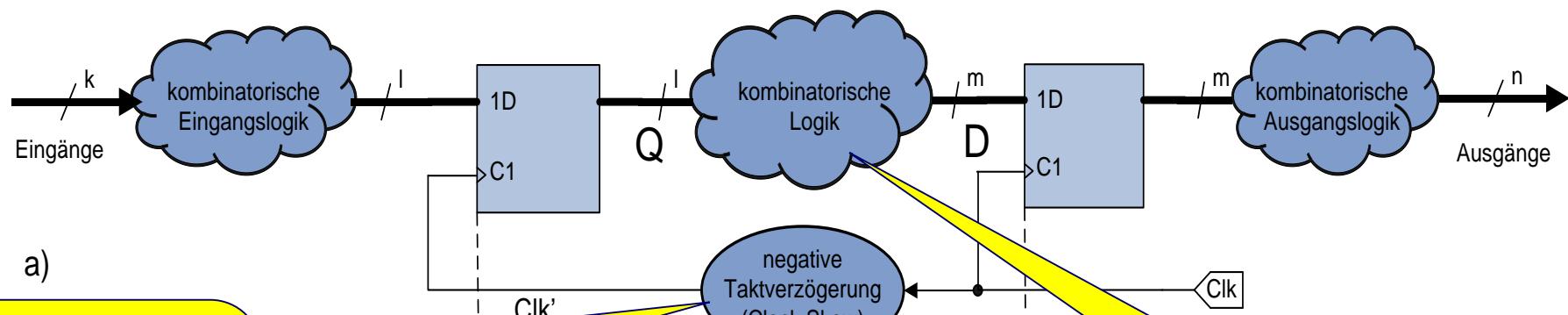
RTL-Modellierung synchroner Schaltungen

Ein Register-Transfer-Modell (RTL) beschreibt eine synchrone Digitalschaltung mit dem Strukturansatz, dass die Eingänge von D-Flipflops entweder Schaltungseingänge sind oder durch kombinatorische Logik angesteuert werden. Die Ausgänge der D-Flipflops sind entweder Schaltungsausgänge oder sie werden in einer weiteren kombinatorischen Logikstufe verarbeitet. Auf diese Weise lässt sich die komplette Schaltung in abwechselnd kombinatorische und getaktete Bauelemente strukturieren.

Beispiel (s. nächste Folie):

- k Eingangssignale werden über eine kombinatorische Eingangslogik an l Eingänge von D-Flipflops gelegt.
- Die l Ausgangssignale Q der ersten Registerstufe werden über einen weiteren kombinatorischen Logikblock mit m Ausgängen auf die Eingänge D einer zweiten Registerstufe gelegt.
- Eine kombinatorische Ausgangsstufe erzeugt aus den m Ausgängen der zweiten Registerstufe n Ausgangssignale.
- Durch interne Signalverzögerungen im programmierbaren Baustein bzw. auf einer Platine ist nicht sichergestellt, dass alle D-Flipflops zum gleichen Zeitpunkt getaktet werden, vielmehr existiert ein Taktversatz, der in der dargestellten Schaltung dazu führt, dass die erste Flipflopstufe etwas später angesteuert wird, als die zweite.

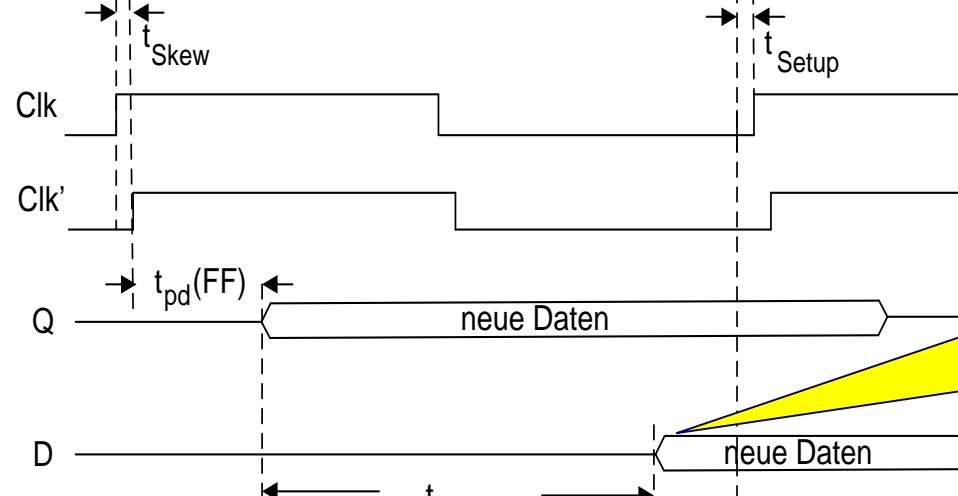
Maximale Taktfrequenz eines synchronen Systems



Definitionsgemäß ist der Skew negativ, wenn die zweite Stufe VOR der ersten Stufe getaktet wird!

Der „kritische Pfad“ wird durch eine statische Timinganalyse ermittelt.

b)



Eine Taktreserve >0 erlaubt die Erhöhung der Taktfrequenz.

$$f_{\max} = \left(-t_{Skew} + t_{pd(FF)} + t_{Komb} + t_{Setup} \right)^{-1}$$

$\rightarrow t$

Zusammenfassung

- Die Ansteuerung von RS-Latches ist wegen des unbedingt zu vermeidenden irregulären Zustands schwierig.
- D-Latches sind wegen ihres transparenten Verhaltens als Speicherelemente in sequenziellen Schaltungen weitaus fehleranfälliger als D-Flipflops. D-Latches tauchen bei der VHDL-Synthese durch semantische Fehler im VHDL-Code leider meist ungewollt auf.
- Das Grundbauelement taktsynchroner Logik ist das D-Flipflop. Nur dafür gibt es ein eigenes VHDL-Entwurfsmuster. Alle anderen Flipflops müssen durch zusätzlich zu modellierende Übergangslogik entweder als Datenfluss- oder Verhaltensmodell realisiert werden.
- Neben D-Flipflops kommen in modernen FPGA-Technologien nur T-Flipflops zum Einsatz, da sich damit unter bestimmten Umständen schnellere Zähler entwerfen lassen.
- Beim VHDL-Entwurf von Latches und Flipflops müssen die Syntheserichtlinien unbedingt eingehalten werden, um nicht durch unerwartet auftauchende Bauelemente überrascht zu werden.
- Die Analyse des Zeitbudgets auf Register-Transfer-Ebene erlaubt es, die maximale Taktfrequenz zu bestimmen, mit der eine synchrone Digitalschaltung sicher betrieben werden kann.