Zeitverhalten von sequentiellen Schaltungen

TECHNISCHE UNIVERSITÄT DARMSTADT

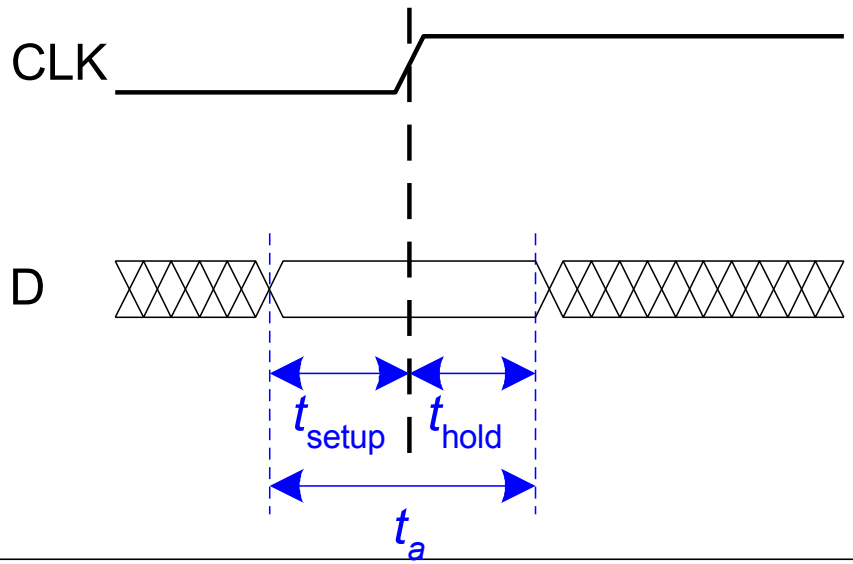
- Flip-Flop übernimmt Daten von *D* zur Taktflanke
- *D* darf sich nicht ändern, wenn es übernommen wird (*sampled*)
  - Muss stabil sein
- Ähnlich zu Fotographie: Keine Bewegung zum Auslösezeitpunkt
  - Sonst unscharf
- Also: *D* darf sich nicht zur Taktflanke ändern
  - Sonst möglicherweise *metastabil*

- Genauer:
  - *D* darf sich nicht in Zeitfenster um Taktflanke herum ändern

Zeitanforderungen an Eingangssignale

TECHNISCHE UNIVERSITÄT DARMSTADT

- Setup-Zeit
  - t_{setup} = Zeitintervall *vor Taktflanke,* in dem *D* sich nicht ändern darf (=stabil sein muss)
- Hold-Zeit
  - t_{hold} = Zeitintervall *nach Taktflanke* in dem *D* stabil sein muss
- Abtastzeit: t_a = Zeitintervall um Taktflanke herum in dem *D* stabil sein muss
  - $t_a = t_{setup} + t_{hold}$

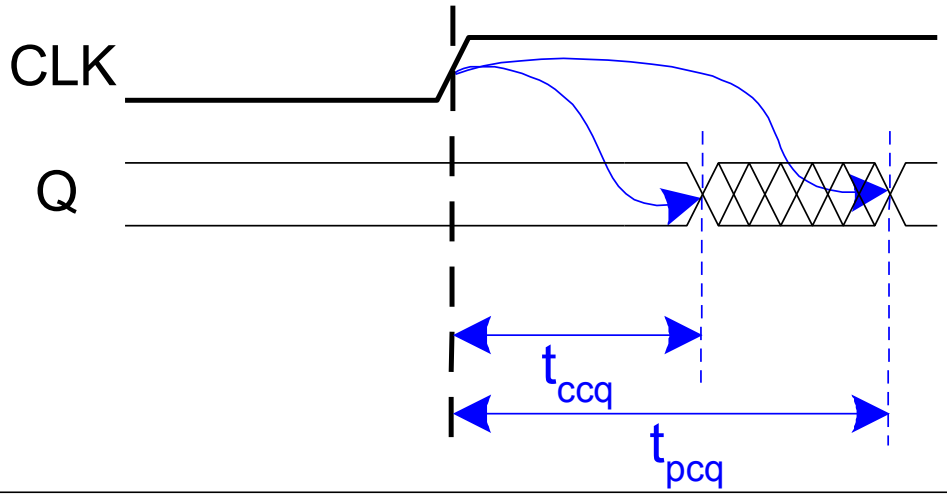


Zeitanforderungen an Ausgangssignale

TECHNISCHE UNIVERSITÄT DARMSTADT

- Laufzeitverzögerung (*propagation delay*)
  - t_{pcq} = Zeitintervall nach Taktflanke, nach dem Q garantiert stabil ist
    - sich also nicht mehr ändert!

- Kontaminationsverzögerung (*contamination delay*)
  - t_{ccq} = Zeitintervall nach Taktflanke, nach dem Q beginnen könnte, sich zu ändern



Dynamische Entwurfsdisziplin

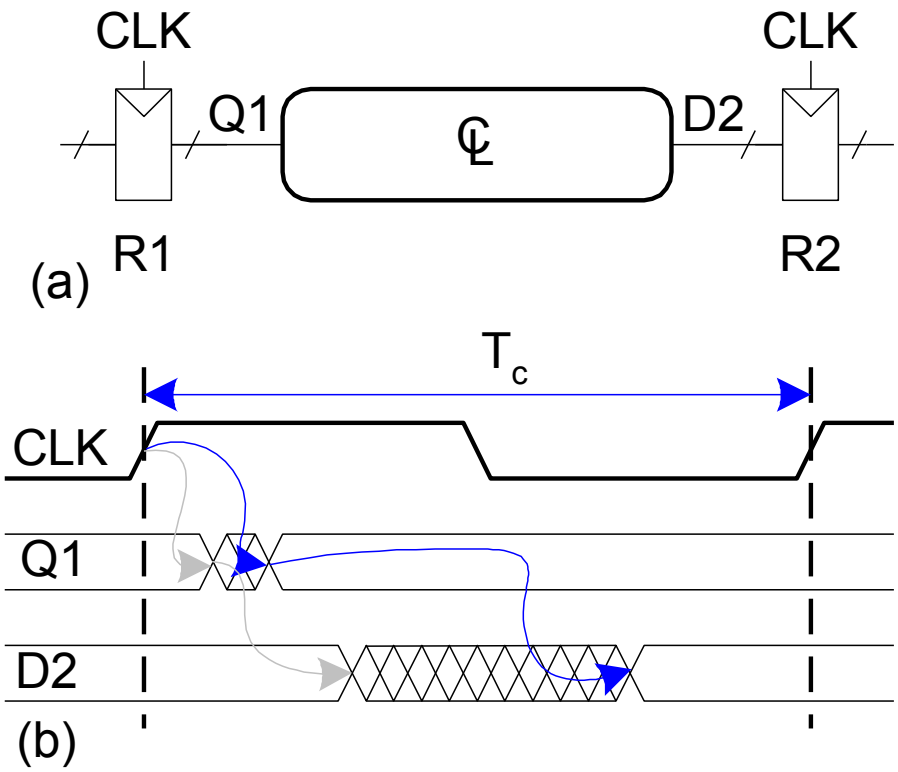
TECHNISCHE UNIVERSITÄT DARMSTADT

- Die Eingänge in eine synchrone sequentielle Schaltung müssen in der ganzen Abtastzeit stabil sein

- Genauer: Stabil mindestens
  - … ab t_{setup} vor der Taktflanke
  - … bis t_{hold} nach der Taktflanke

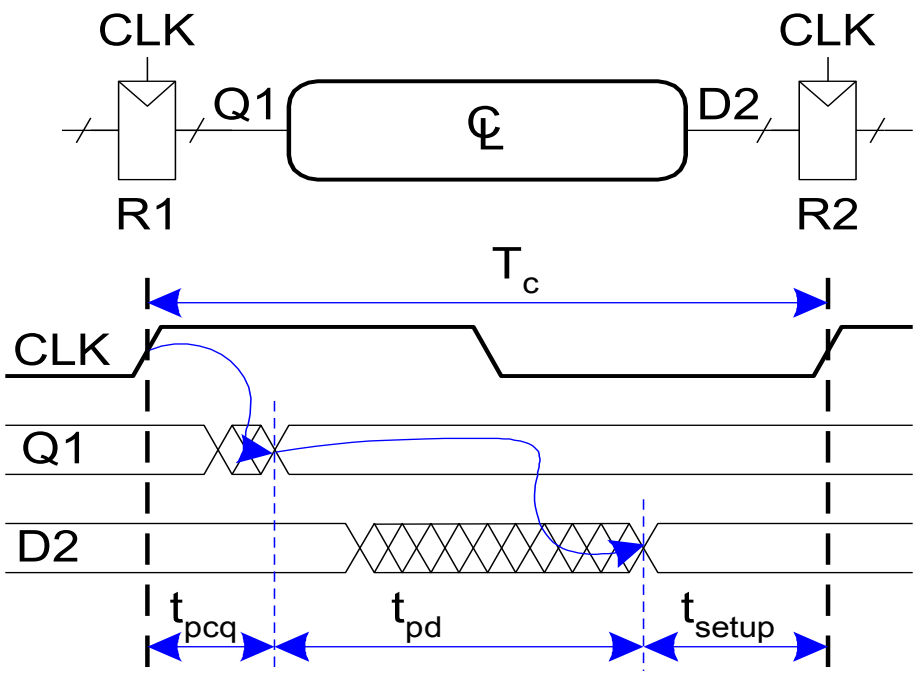
Dynamische Entwurfsdisziplin

- Verzögerung zwischen Registern hat Maximal- und Minimalwert
  - Abhängig von den Verzögerungen der kombinatorischen Schaltelemente



Anforderungen an Setup-Zeit

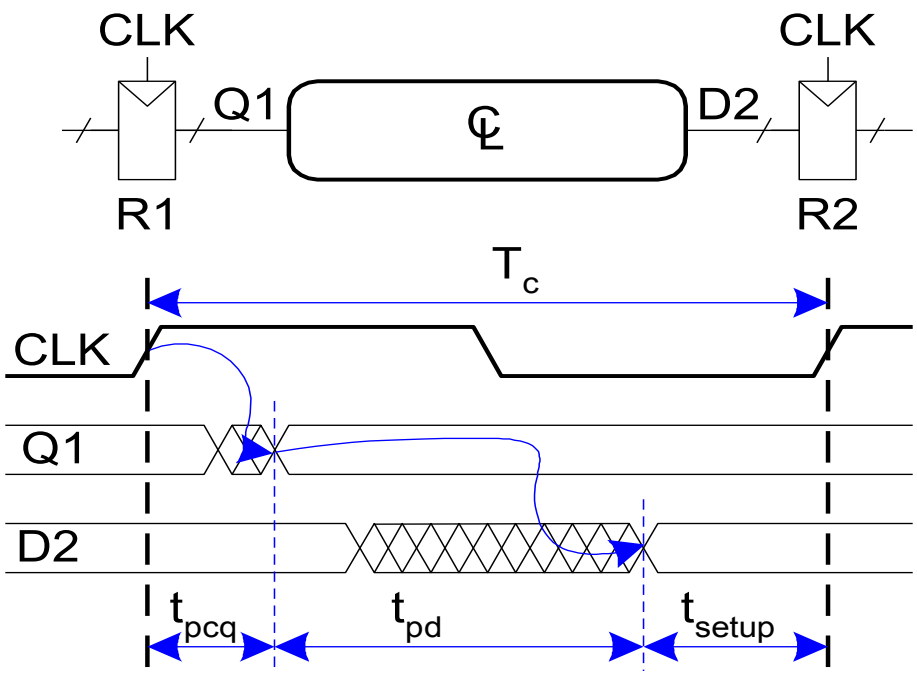
- Einhalten der Setup-Zeit hängt von der Maximal-Verzögerung von Register R1 durch kombinatorische Logik ab
- Eingang zu Register muss mindestens ab t_{setup} vor Taktflanke stabil sein



$T_c \geq$

Anforderungen an Setup-Zeit

- Einhalten der Setup-Zeit hängt von der Maximal-Verzögerung von Register R1 durch kombinatorische Logik ab
- Eingang zu Register muss mindestens ab t_{setup} vor Taktflanke stabil sein

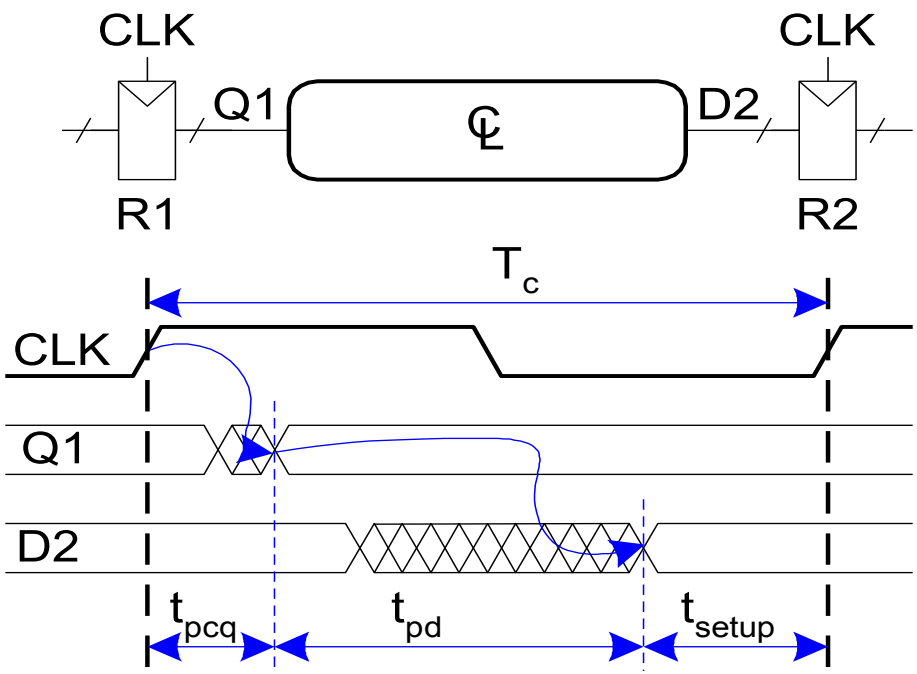


$T_c \geq t_{pcq} + t_{pd} + t_{setup}$

$t_{pd} \leq$

Anforderungen an Setup-Zeit

- Einhalten der Setup-Zeit hängt von der Maximal-Verzögerung von Register R1 durch kombinatorische Logik ab
- Eingang zu Register muss mindestens ab t_{setup} vor Taktflanke stabil sein

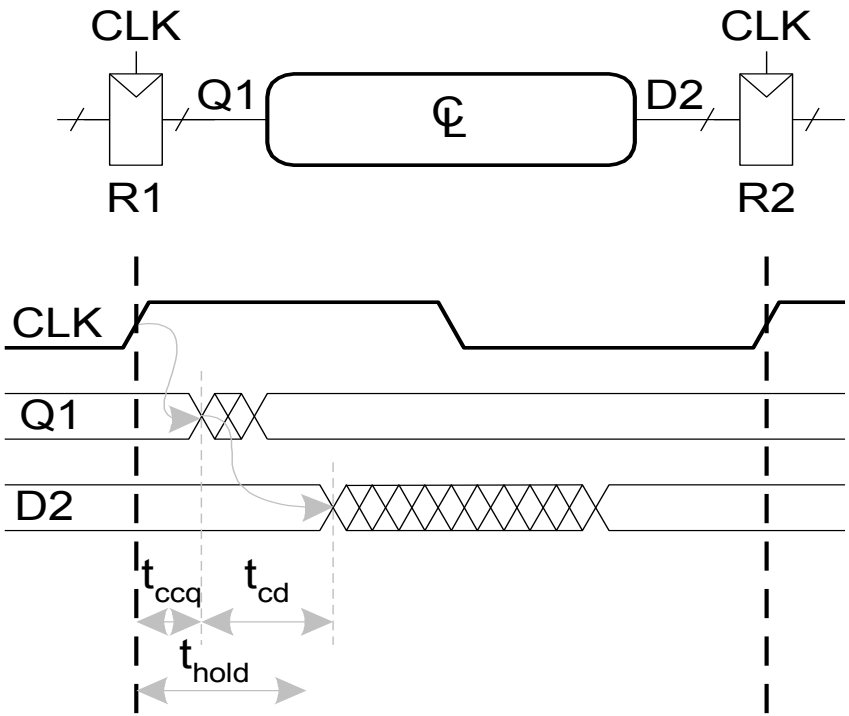


$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

$$t_{pd} \leq T_c - (t_{pcq} + t_{setup})$$

Anforderungen an Hold-Zeit

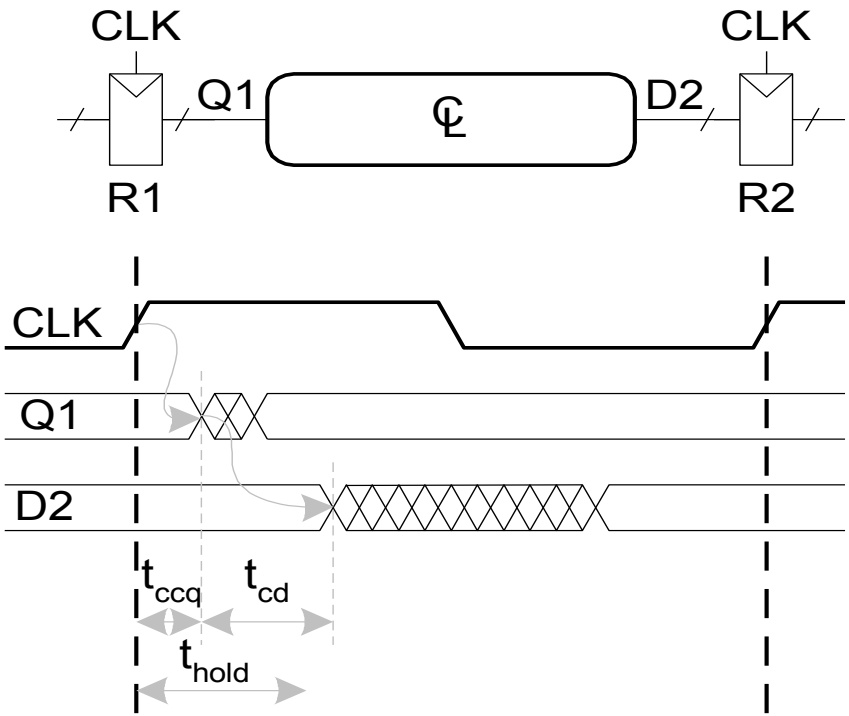
- Einhalten der Hold-Zeit hängt von der minimalen Verzögerung von Register R1 durch die kombinatorische Logik ab
- Der Eingang an Register R2 muss mindestens bis t_{hold} nach der Taktflanke stabil sein



t_{hold} <

Anforderungen an Hold-Zeit

- Einhalten der Hold-Zeit hängt von der minimalen Verzögerung von Register R1 durch die kombinatorische Logik ab
- Der Eingang an Register R2 muss mindestens bis t_{hold} nach der Taktflanke stabil sein

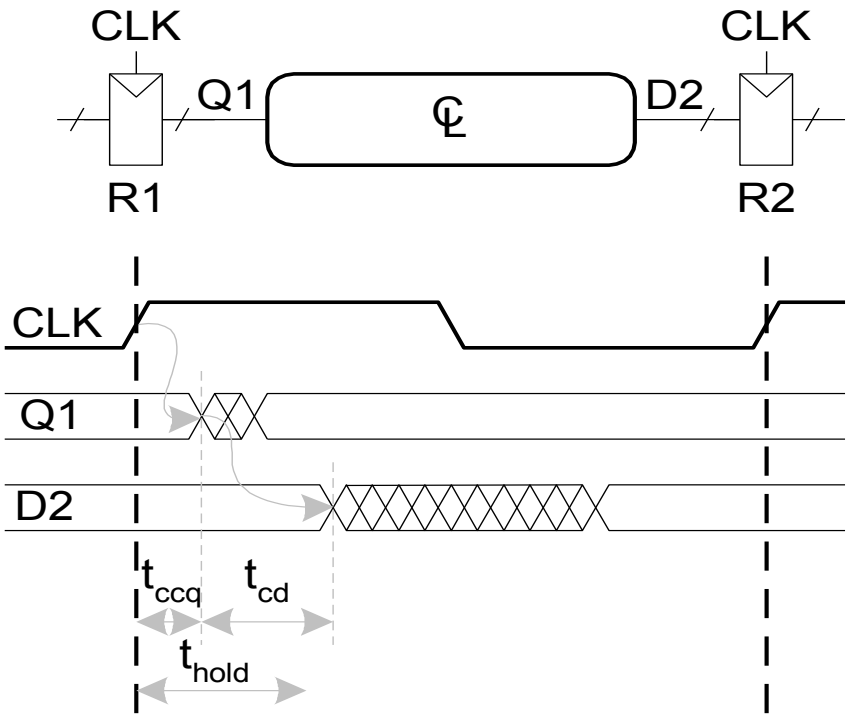


$t_{hold} < t_{ccq} + t_{cd}$

$t_{cd} >$

Anforderungen an Hold-Zeit

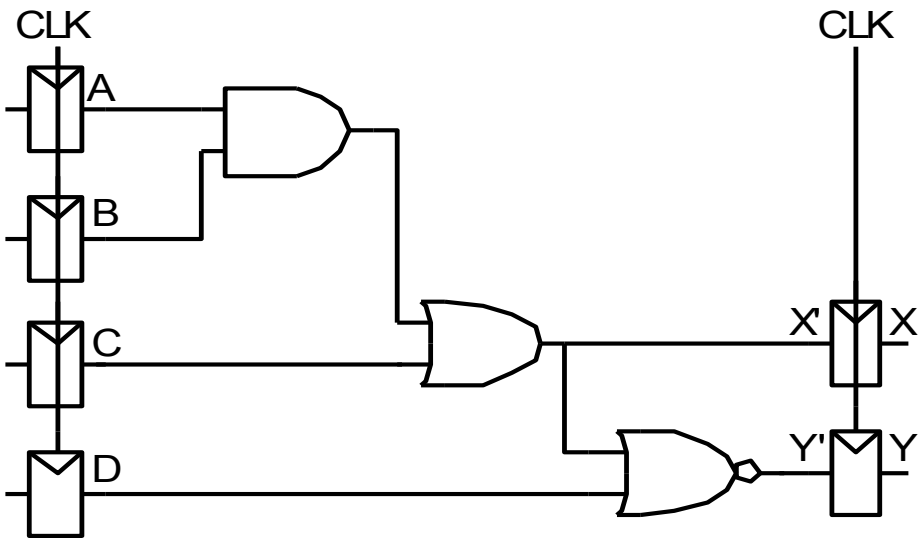
- Einhalten der Hold-Zeit hängt von der minimalen Verzögerung von Register R1 durch die kombinatorische Logik ab
- Der Eingang an Register R2 muss mindestens bis t_{hold} nach der Taktflanke stabil sein



$$t_{hold} < t_{ccq} + t_{cd}$$

$$t_{cd} > t_{hold} - t_{ccq}$$

Analyse des Zeitverhaltens



Verzögerungsangaben

t_{ccq} = 30 ps

t_{pcq} = 50 ps

t_{setup} = 60 ps

t_{hold} = 70 ps

Pro Gatter

t_{pd} = 35 ps

t_{cd} = 25 ps

t_{pd} =

t_{cd} =

Einhalten von Setup-Zeitanforderung:

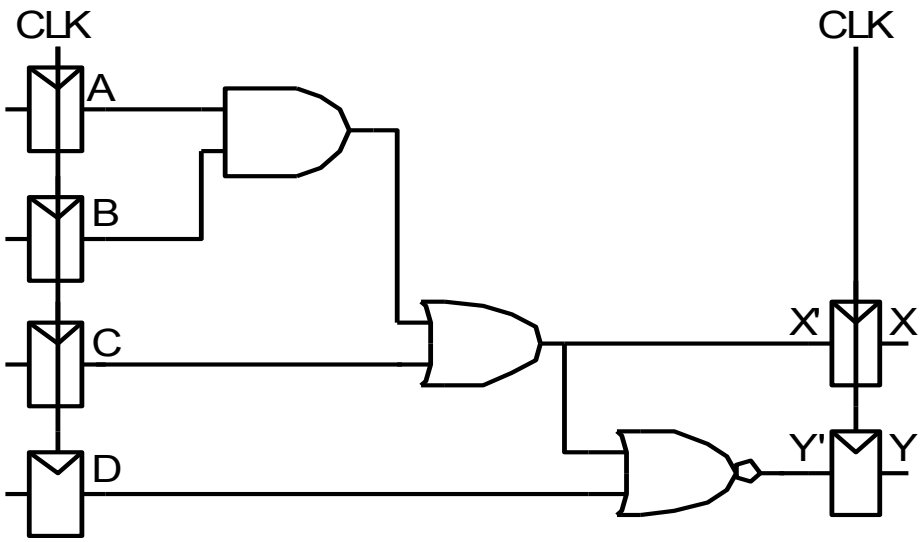
$T_c \geq$

$f_c = 1/T_c =$

Einhalten von Hold-Zeitanforderung:

$t_{ccq} + t_{cd} > t_{hold}$?

Analyse des Zeitverhaltens



Verzögerungsangaben

t_{ccq} = 30 ps

t_{pcq} = 50 ps

t_{setup} = 60 ps

t_{hold} = 70 ps

Pro Gatter

t_{pd} = 35 ps

t_{cd} = 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 25 ps

Einhalten der Setup-Zeitanforderung:

$T_c \geq (50 + 105 + 60)$ ps = 215 ps

$f_c = 1/T_c$ = 4,65 GHz

Einhalten der Hold-Zeitanforderung:

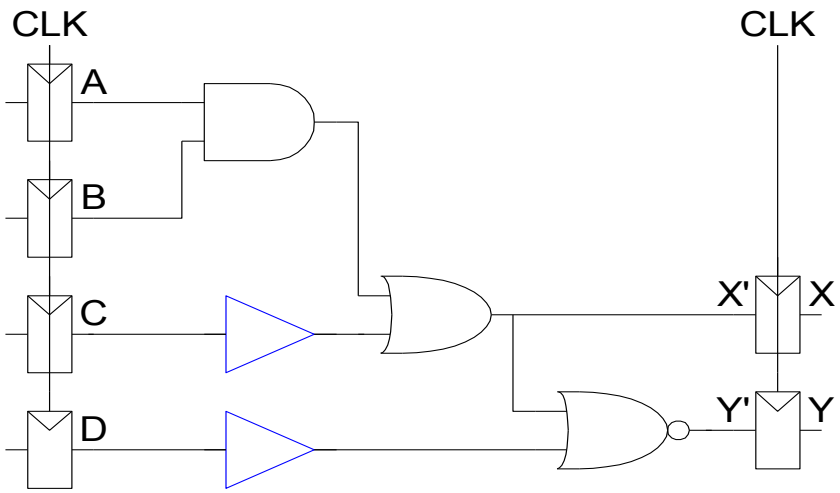
$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 25) ps > 70 ps ? Nein, verletzt!

Beheben der verletzten Hold-Zeitanforderung

TECHNISCHE UNIVERSITÄT DARMSTADT

Füge Puffer in zu kurze Pfade ein!



Verzögerungsangaben

t_{ccq} = 30 ps

t_{pcq} = 50 ps

t_{setup} = 60 ps

t_{hold} = 70 ps

Pro Gatter

t_{pd} = 35 ps

t_{cd} = 25 ps

t_{pd} =

t_{cd} =

Einhalten der Setup-Zeitanforderung:

$T_c \geq$

f_c =

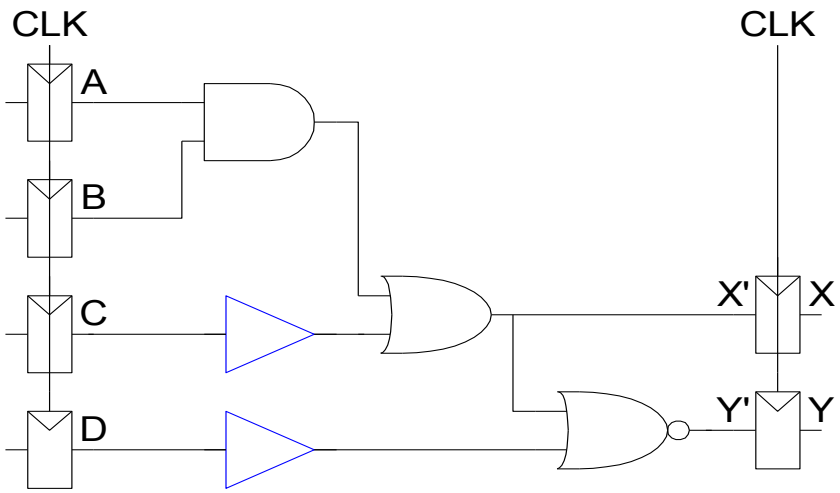
Einhalten der Hold-Zeitanforderung:

$t_{ccq} + t_{cd} > t_{hold}$?

Beheben der verletzten Hold-Zeitanforderung

TECHNISCHE UNIVERSITÄT DARMSTADT

Füge Puffer in zu kurze Pfade ein!



Verzögerungsangaben

t_{ccq} = 30 ps

t_{pcq} = 50 ps

t_{setup} = 60 ps

t_{hold} = 70 ps

Pro Gatter

t_{pd} = 35 ps

t_{cd} = 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Einhalten der Setup-Zeitanforderung:

$T_c \geq (50 + 105 + 60)$ ps = 215 ps

$f_c = 1/T_c$ = 4.65 GHz

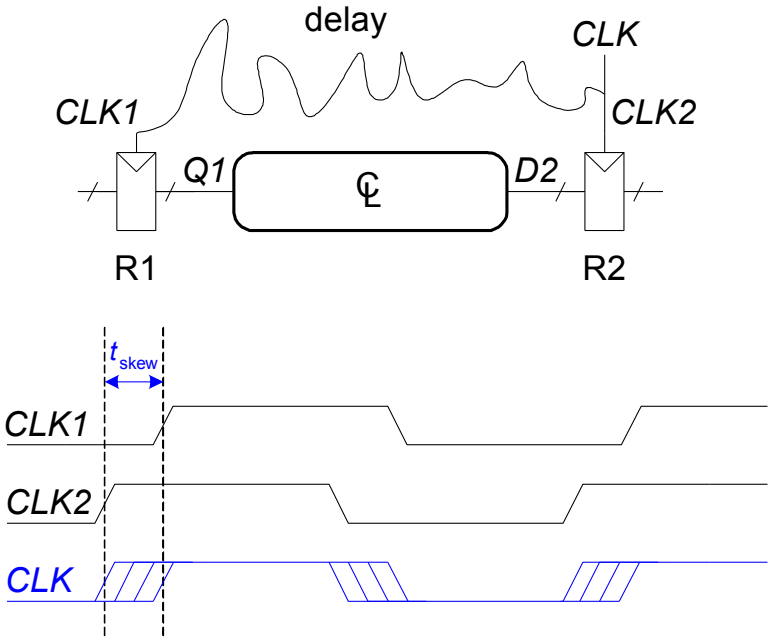
Einhalten der Hold-Zeitanforderung:

$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 50) ps > 70 ps ? Ja, eingehalten!

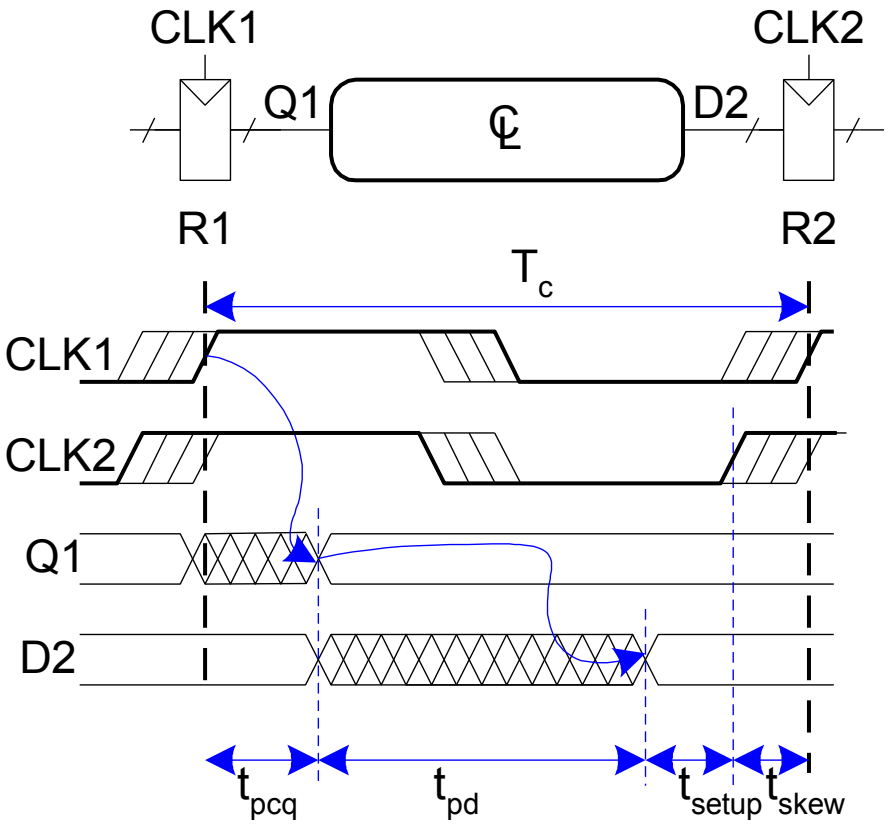
Taktverschiebung (*clock skew*)

- Der Takt kommt nicht bei allen Registern zur gleichen Zeit an
  - Unterschiedliche Verdrahtungswege auf dem Chip, Logik in Taktsignal (gated clock, vermeiden!)
- Verschiebung (oder Versatz, *skew*) ist die Differenz der Ankunftszeit zwischen zwei Registern
- Überprüfe, ob auch bei maximalem Versatz die dynamische Entwurfsdisziplin noch eingehalten ist



Setup-Zeitanforderungen mit Taktverschiebung

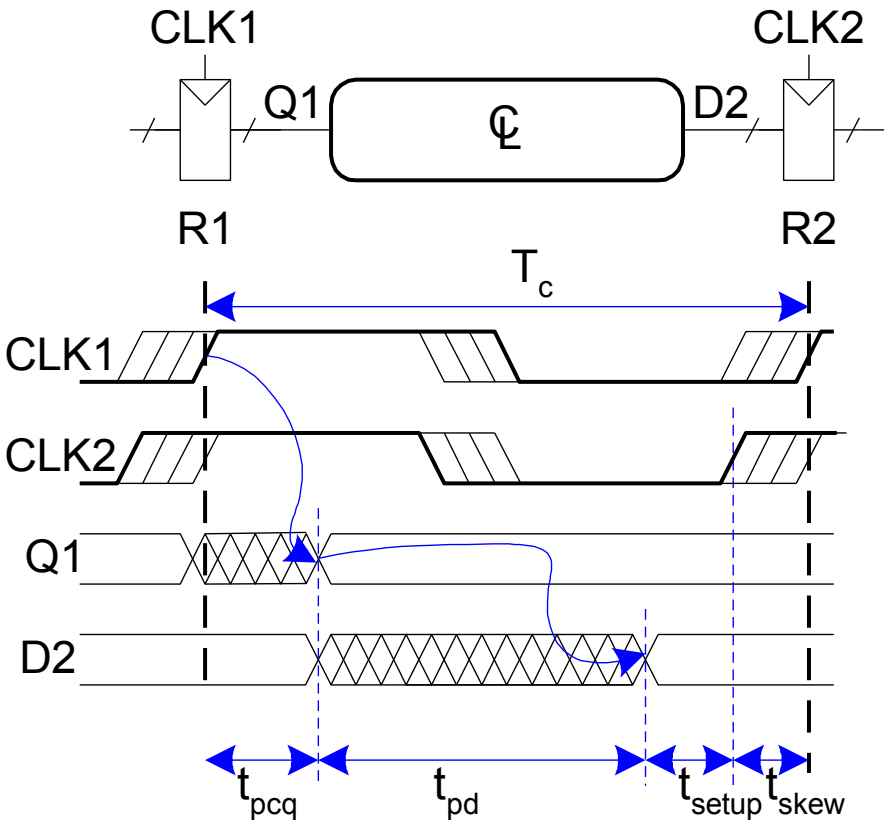
- In diesem Beispiel: CLK2 ist früher aktiv als CLK1



$T_c \geq$

Setup-Zeitanforderungen mit Taktverschiebung

- In diesem Beispiel: CLK2 ist früher aktiv als CLK1

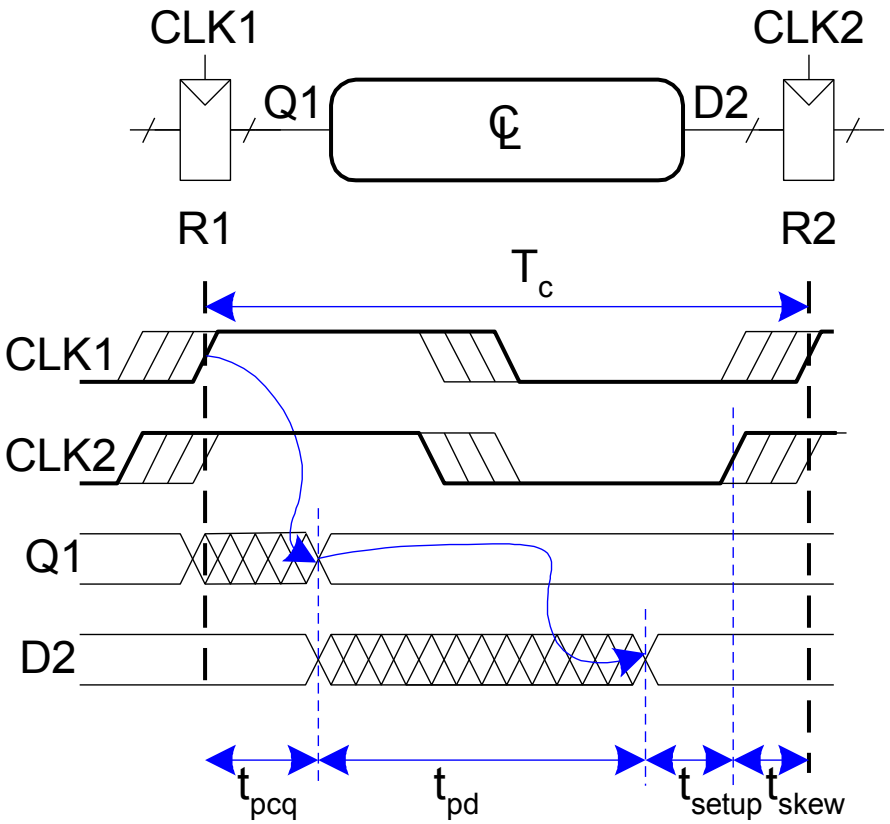


$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$t_{pd} \leq$$

Setup-Zeitanforderungen mit Taktverschiebung

- In diesem Beispiel: CLK2 ist früher aktiv als CLK1

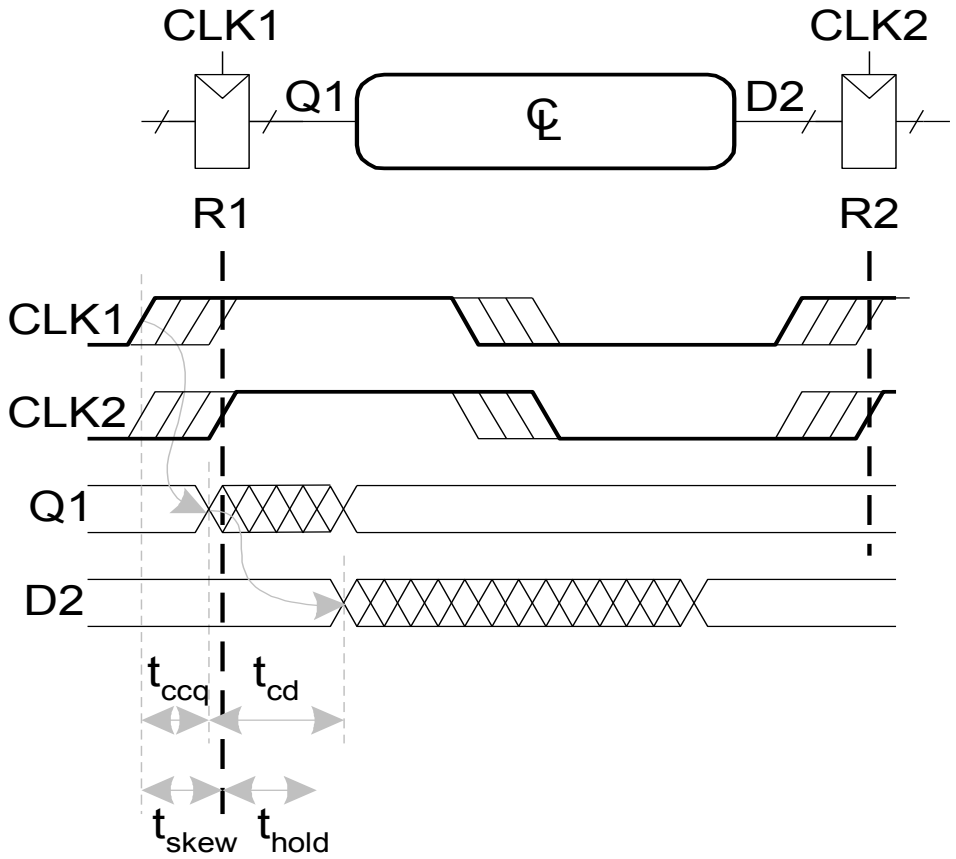


$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$t_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew})$$

Hold-Zeitanforderungen mit Taktverschiebung

- In anderem Fall: CLK2 könnte später als CLK1 aktiviert werden

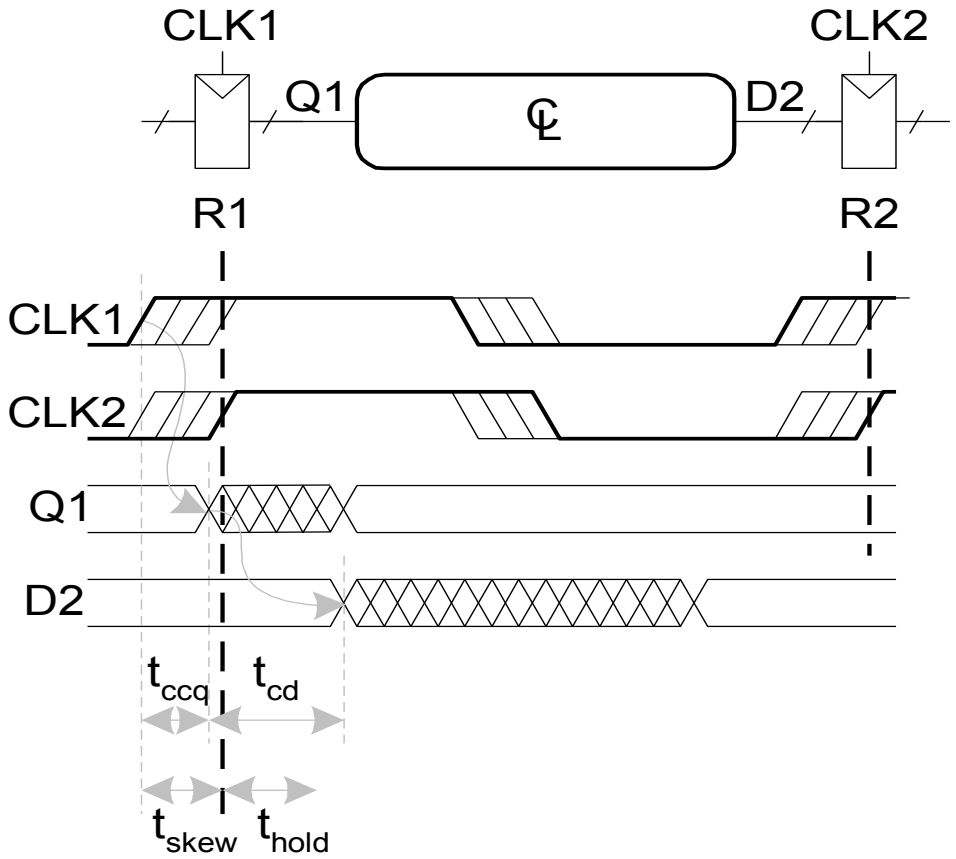


$t_{ccq} + t_{cd} >$
$t_{cd} >$

Hold-Zeitanforderungen mit Taktverschiebung

TECHNISCHE UNIVERSITÄT DARMSTADT

- In anderem Fall: CLK2 könnte später als CLK1 aktiviert werden



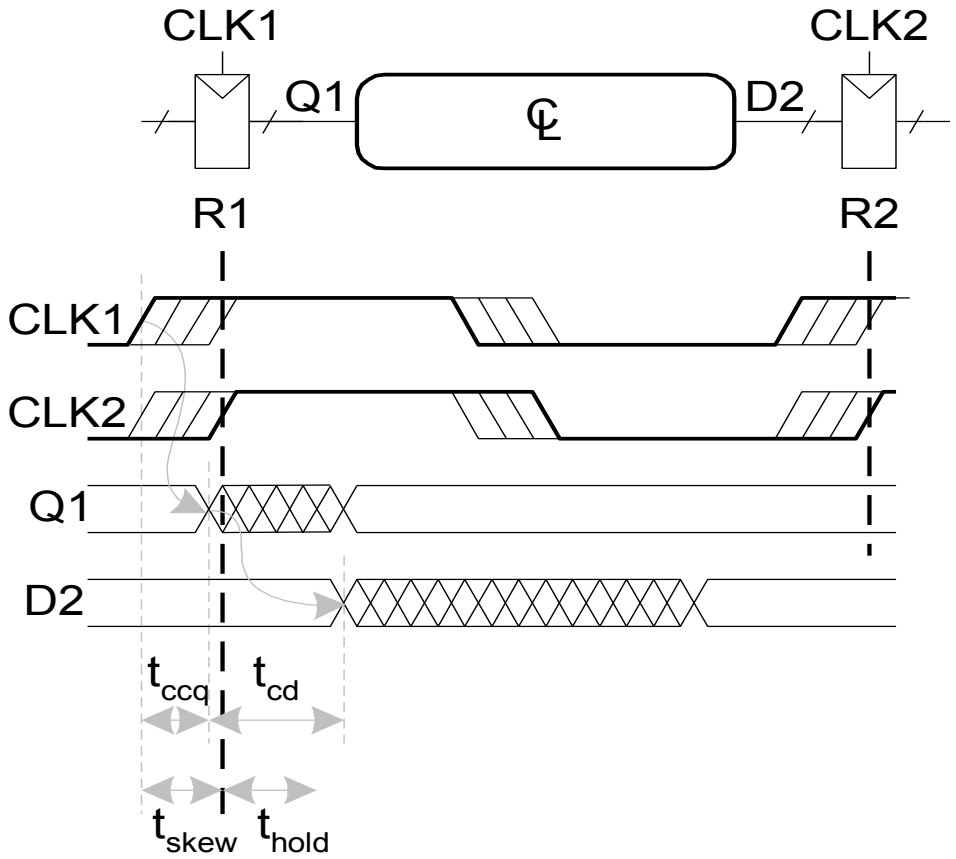
$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$

$t_{cd} >$

LEHRE WIKI TEST IN ZWEI FOLIEN

Hold-Zeitanforderungen mit Taktverschiebung

- In anderem Fall: CLK2 könnte später als CLK1 aktiviert werden

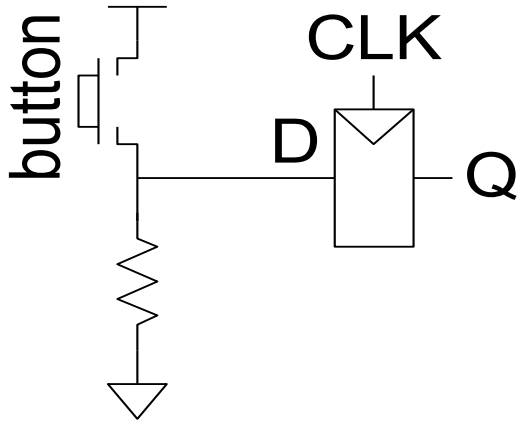


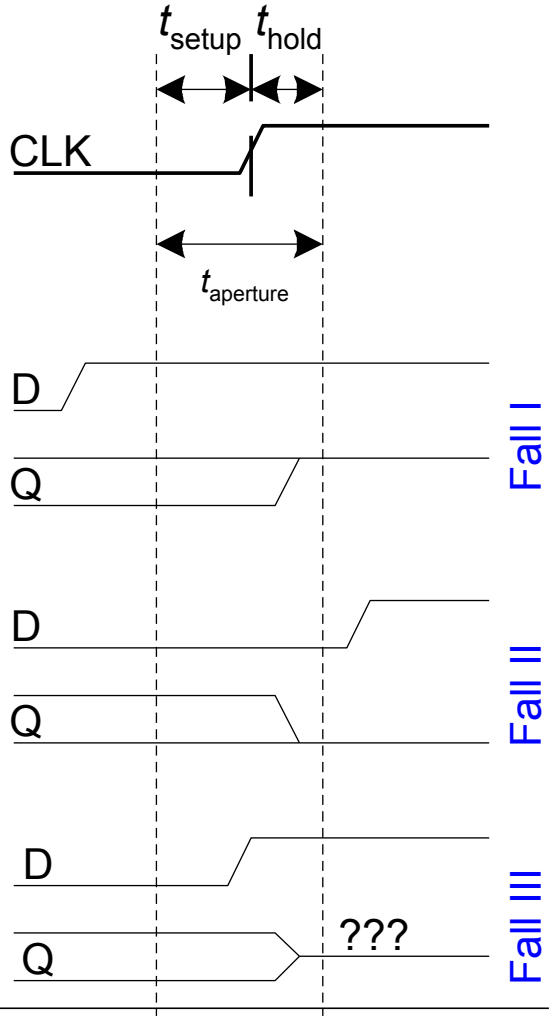
$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$

$$t_{cd} > t_{hold} + t_{skew} - t_{ccq}$$

Verletzung der dynamischen Entwurfsdisziplin

- Asynchrone Eingänge können dynamische Disziplin verletzen
- Beispiel: Benutzereingaben



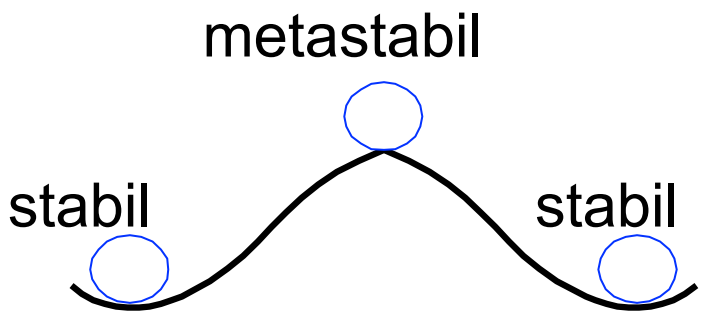


LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

Metastabilität

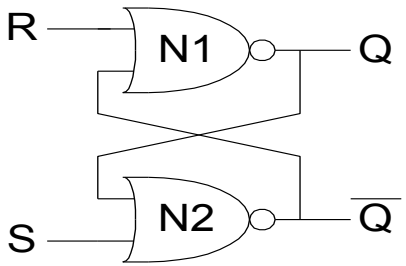
- Jedes bistabile Element hat zwei stabile Zustände und einen metastabilen dazwischen
- Ein Flip-Flop-Ausgang hat zwei stabile Zustände (0 und 1) und einen metastabilen Zustand
- Falls das Flip-Flop den metastabilen Zustand annimmt, kann es dort für unbestimmte Zeit verbleiben



Interner Aufbau eines Flip-Flops

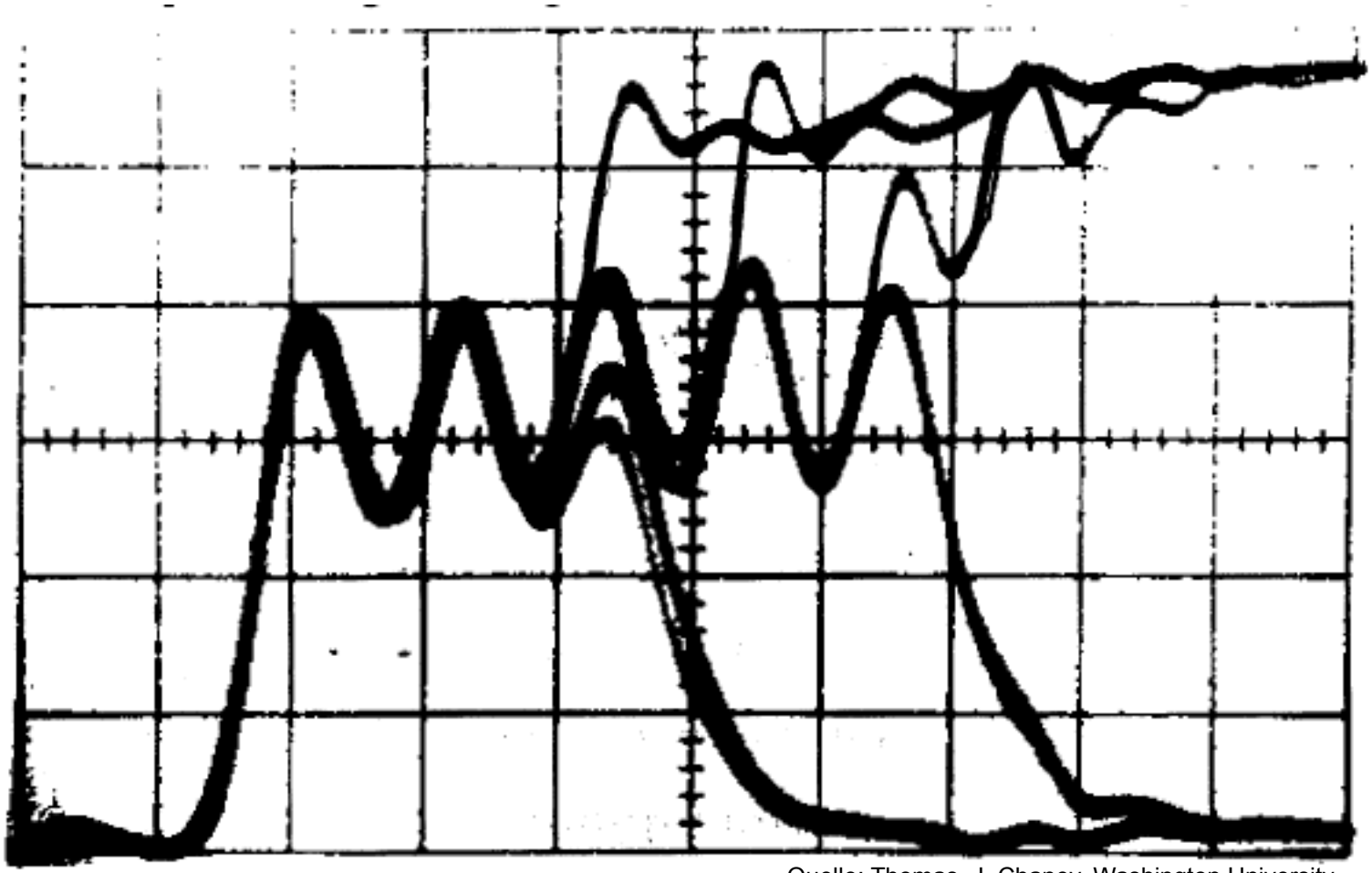
TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Flip-Flop hat intern Rückkopplungen
- Falls *Q* zwischen 1 und 0 liegt:
  - … wird es von den kreuzgekoppelten Gattern irgendwann auf 1 oder 0 getrieben
  - Je nachdem, an welchem Spannungspegel es näher lag



- Ein Signal wird als metastabil bezeichnet, wenn es noch nicht zu 1 oder 0 aufgelöst wurde

Metastabilität: Beispiel



Quelle: Thomas. J. Chaney, Washington University

Zeitdauer der Metastabilität

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wenn Flip-Flop Eingang *D* zu einem zufälligen Zeitpunkt innerhalb der Abtastzeit wechselt
- … wird Ausgang Q nach einer zufälligen Zeit t_{res} zu 0 oder 1 aufgelöst (*resolved*)

- Wahrscheinlichkeit, dass Ausgang *Q* nach einer Wartezeit *t* noch metastabil ist:

$$P(t_{res} > t) = (T_0/T_c)\, e^{-t/\tau}$$

t_{res} : Zeit um Ausgang sicher nach 1 or 0 auzulösen

T_0, τ : Eigenschaften der Schaltung

T_C : Taktperiode

Zeitdauer der Metastabilität: Interpretation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Intuitiv
  - T_0/T_c ist Wahrscheinlichkeit, dass Eingang zu einem ungünstigen Zeitpunkt schaltet
    - Innerhalb der Abtastzeit, sinkt mit wachsender Taktperiode T_c

$$P(t_{res} > t) = (T_0/T_c)\, e^{-t/\tau}$$

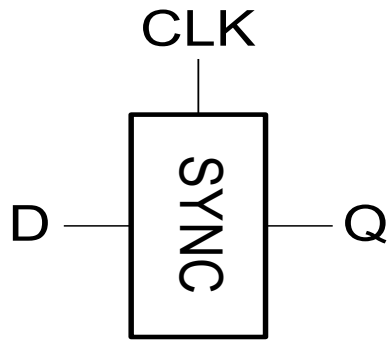
  - Die Zeitkonstante τ gibt an, wie schnell Flip-Flop sich aus dem metastabilen Zustand wegbewegen kann
    - Hängt von der Verzögerung durch die kreuzgekoppelten Gatter ab

$$P(t_{res} > t) = (T_0/T_c)\, e^{-t/\tau}$$

- Kurzfassung: Wenn man nur lange genug wartet, wird der Ausgang sicher zu 0 oder 1 aufgelöst

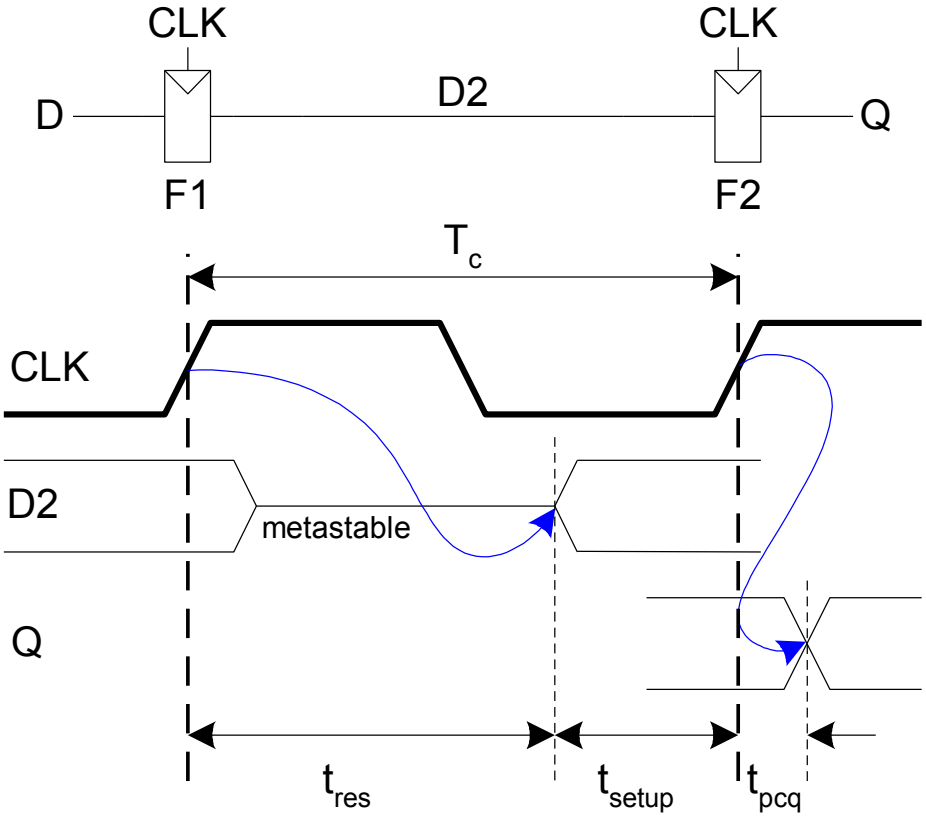
Synchronisierer (*synchronizer*)

- Asynchrone Eingänge (*D*) lassen sich praktisch nicht ganz vermeiden
  - Benutzerschnittstellen
  - Systeme mit mehreren interagierenden Taktsignalen
- Ziel eines Synchronisierers
  - Reduziere die Wahrscheinlichkeit für metastabilen Zustand
  - Liefere an Q mit hoher Wahrscheinlichkeit gültige Werte
- Metastabilität kann aber **nie** völlig ausgeschlossen werden



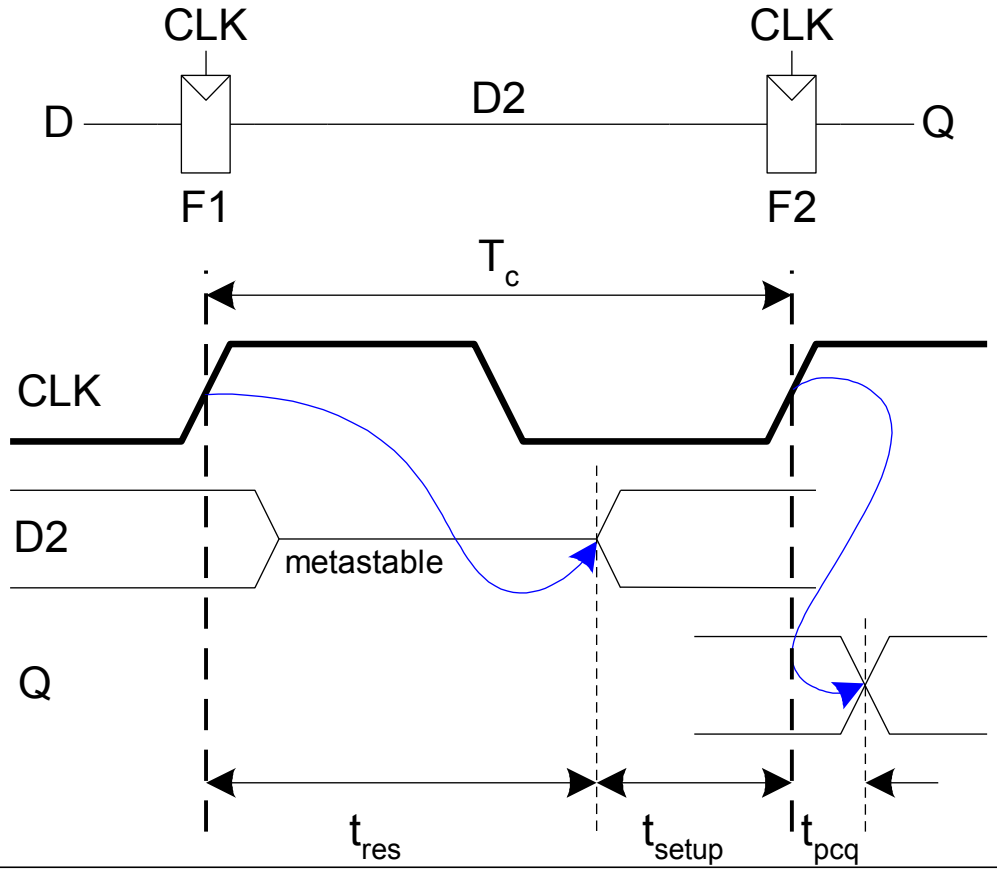
Interner Aufbau eines Synchronisierers

- Aufgebaut aus Reihenschaltung von Flip-Flops
- Annahme: Eingang D wechselt während der Abtastzeit von Flip-Flop F1
- Synchronisation gelingt, wenn Signal D2 innerhalb von (T_c - t_{setup}) zu 0 oder 1 aufgelöst wird



Wahrscheinlichkeit für Scheitern der Synchronisation

Für jede Änderung des Eingangs D: $\mathbf{P(Scheitern) = (T_0/T_c)\, e^{-(T_c - t_{setup})/\tau}}$



Mittlere Betriebsdauer zwischen Ausfällen (*mean time between failures, MTBF)*

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bei Änderung des Eingangssignals einmal pro Sekunde
  - Ausfallwahrscheinlichkeit des Synchronisierers pro Sekunde ist *P*(Scheitern)
- Bei Änderung des Eingangssignals *N*-mal pro Sekunde
  - Ausfallwahrscheinlichkeit des Synchronisierers pro Sekunde ist

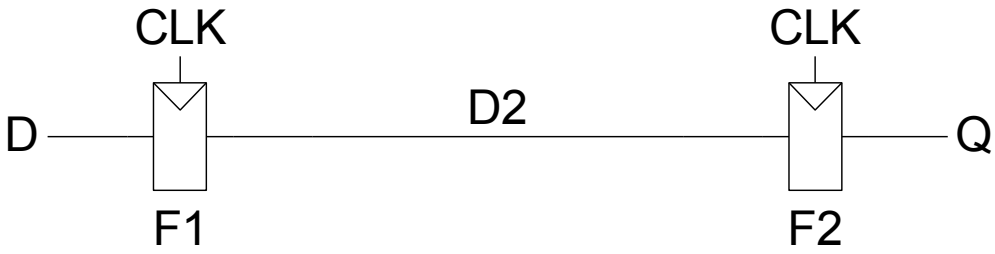
$$P(\text{Scheitern})/s = (NT_0/T_c)\, e^{-(T_c - t_{setup})/\tau}$$

- Im Durchschnitt scheitert die Synchronisation also alle 1/[P(Scheitern)/s] Sekunden

- Wird auch genannt "Mittlere Betriebsdauer zwischen Ausfällen" (MTBF)

$$\text{MTBF} = 1/[P(\text{Scheitern})/s] = (T_c/NT_0)\, e^{(T_c - t_{setup})/\tau}$$

Beispiel: Synchronisierer



- Annahmen: T_c = 1/500 MHz = 2 ns τ = 200 ps
  T_0 = 150 ps t_{setup} = 100 ps
  N = 10 Änderungen pro Sekunde

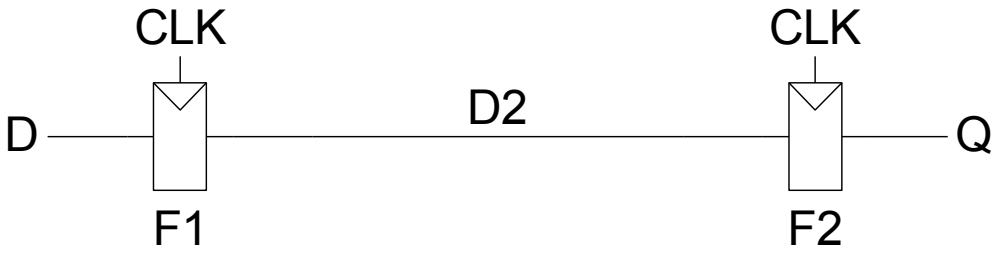
- Ausfallwahrscheinlichkeit? MTBF?

P(Scheitern) =

P(Scheitern)/s =

MTBF =

Beispiel: Synchronisierer



- Annahmen: T_c = 1/500 MHz = 2 ns τ = 200 ps
  T_0 = 150 ps t_{setup} = 100 ps
  N = 10 Änderungen pro Sekunde

- Ausfallwahrscheinlichkeit? MTBF?

$$P(\text{Scheitern}) = (150\text{ ps}/2\text{ ns})\, e^{-(1.9\text{ ns})/200\text{ ps}} = 5.6 \times 10^{-6}$$

$$P(\text{Scheitern})/\text{s} = 10 \times (5.6 \times 10^{-6}) = 5.6 \times 10^{-5} / \text{s}$$

$$\text{MTBF} = 1/[P(\text{Scheitern})/\text{s}] \approx 5 \text{ Stunden}$$

Parallelität

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Zwei Arten von Parallelität**
  - **Räumliche Parallelität**
    - Vervielfachte Hardware bearbeitet mehrere Aufgaben gleichzeitig
  - **Zeitliche Parallelität**
    - Aufgabe wird in mehrere Unteraufgaben aufgeteilt
    - Unteraufgaben werden parallel ausgeführt
    - Beispiel: Fließbandprinzip bei Autofertigung
      - Nur eine Station für einen Arbeitsschritt
      - Aber alle unterschiedlichen Arbeitsschritte für mehrere Autos werden parallel ausgeführt
    - Auch genannt: Pipelining

Parallelität: Grundlegende Begriffe

TECHNISCHE UNIVERSITÄT DARMSTADT

- Einige Definitionen:
  - **Datensatz:** Vektor aus Eingabewerten, die zu einem Vektor aus Ausgabewerten bearbeitet werden
  - **Latenz:** Zeit von der Eingabe eines Datensatzes bis zur Ausgabe der zugehörigen Ergebnisse
  - **Durchsatz:** Die Anzahl von Datensätzen die pro Zeiteinheit bearbeitet werden können
- Parallelität erhöht Durchsatz

Beispiel Parallelität: Plätzchen backen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Weihnachtszeit steht vor der Tür, also rechtzeitig anfangen!

- Annahmen
  - Genug Teig ist fertig
  - 5 Minuten um ein Blech mit Teig zu bestücken
  - 15 Minuten Backzeit

- Vorgehensweise
  - Ein Blech nach dem anderen vorbereiten und backen

Latenz =

Durchsatz =

Beispiel Parallelität: Plätzchen backen (seriell)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Weihnachtszeit steht vor der Tür, also rechtzeitig anfangen!

- Annahmen
  - Genug Teig ist fertig
  - 5 Minuten um ein Blech mit Teig zu bestücken
  - 15 Minuten Backzeit

- Vorgehensweise
  - Ein Blech nach dem anderen vorbereiten und backen

Latenz = 5 + 15 = 20 Minuten = 1/3 h

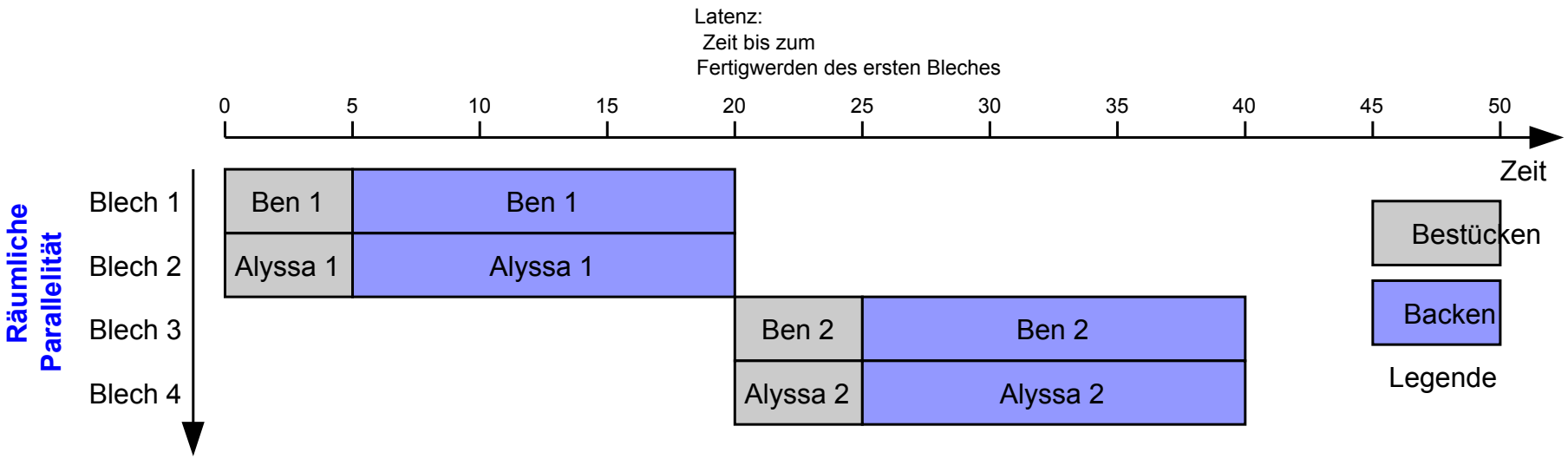
Durchsatz = 1 Blech alle 20 Minuten = 3 Bleche/h

Beispiel Parallelität: Plätzchen backen (parallel)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gleiche Annahmen wie eben
  - 5 Minuten Blech bestücken, 15 Minuten Backen
- Alternative Vorgehensweisen
  - **Räumliche Parallelität:** Zwei Bäcker (Ben & Alyssa), jeder mit einem eigenen Ofen
  - **Zeitliche Parallelität:** Aufteilen der Keksherstellung in Unteraufgaben
    - Blech bestücken
    - Backen
    - Nächstes Blech bestücken, während erstes noch im Ofen gebacken wird
- Latenz und Durchsatz?

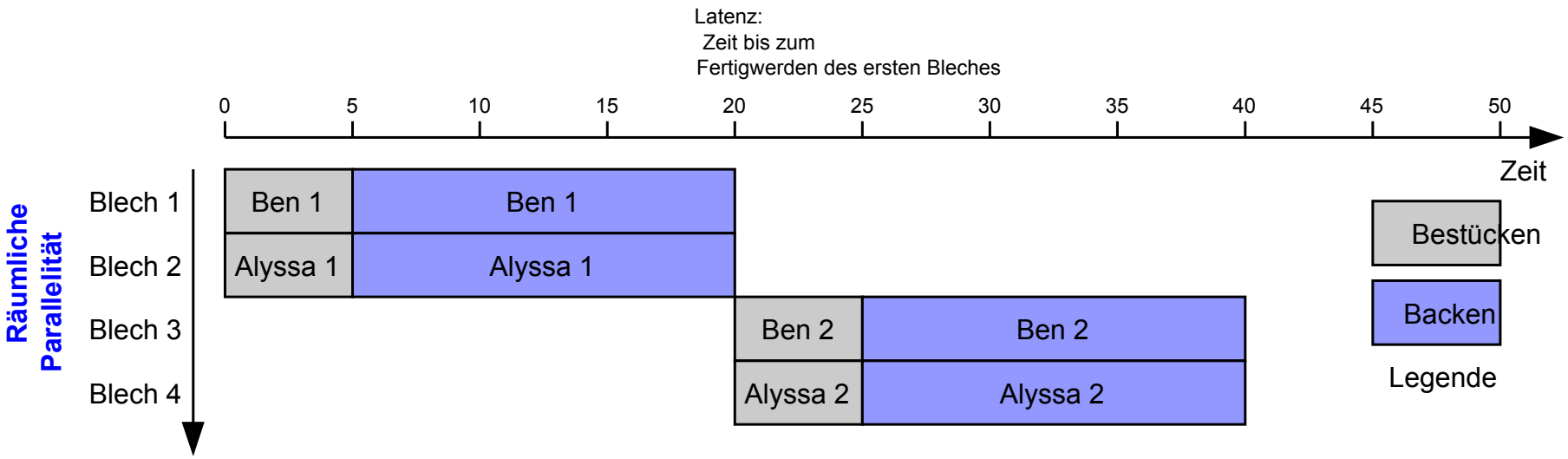
Räumliche Parallelität



Latenz =

Durchsatz =

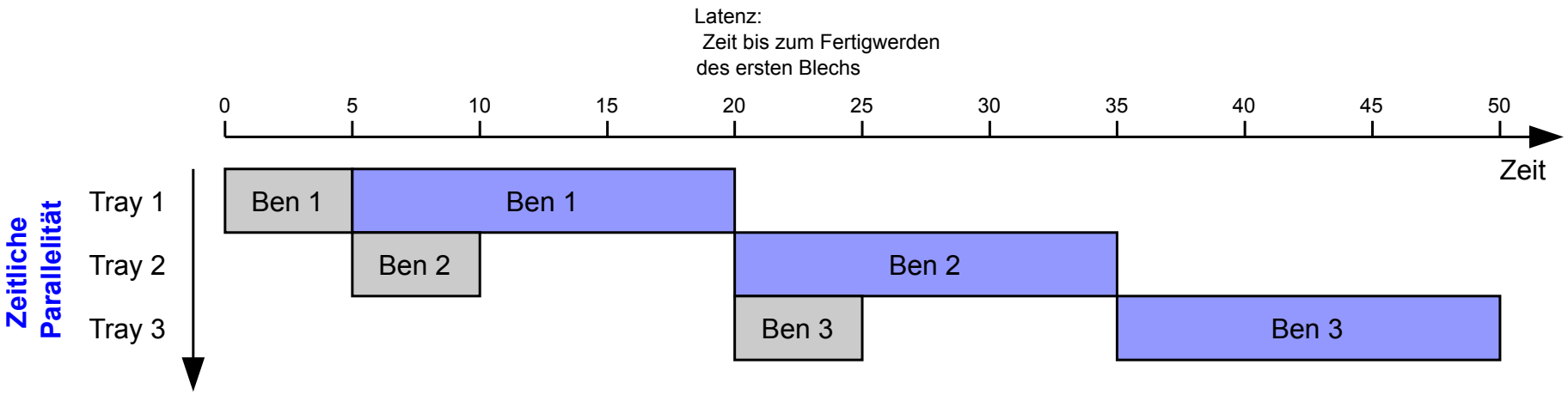
Räumliche Parallelität



Latenz = 5 + 15 = 20 Minuten = 1/3 h

Durchsatz = 2 Bleche alle 20 Minuten = 6 Bleche/h

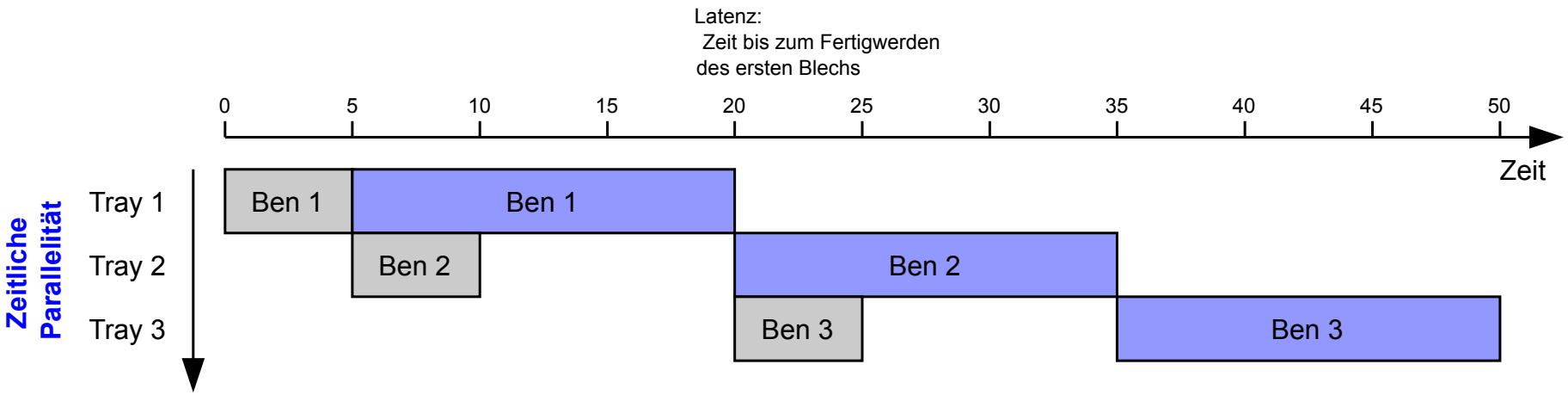
Zeitliche Parallelität



Latenz =

Durchsatz =

Zeitliche Parallelität



Latenz = 5 + 15 = 20 Minuten = 1/3 h

Durchsatz = 1 Blech alle 15 Minuten = 4 Bleche/h

Kombinieren

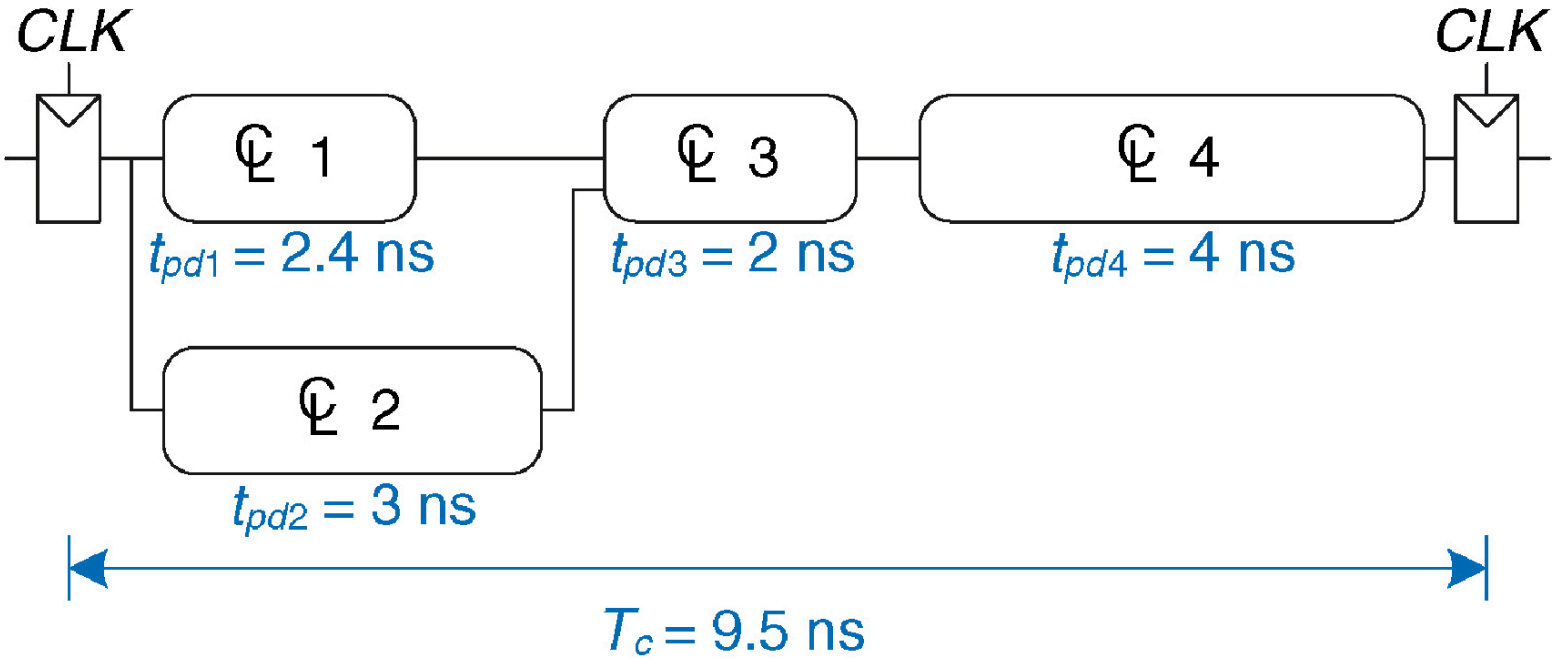
- Zeitliche und räumliche Parallelität können miteinander kombiniert werden

- Hier:
  - Zwei Bäcker und Öfen
  - Nächstes Blech bestücken während altes gebacken wird

- Latenz = 20 Minuten

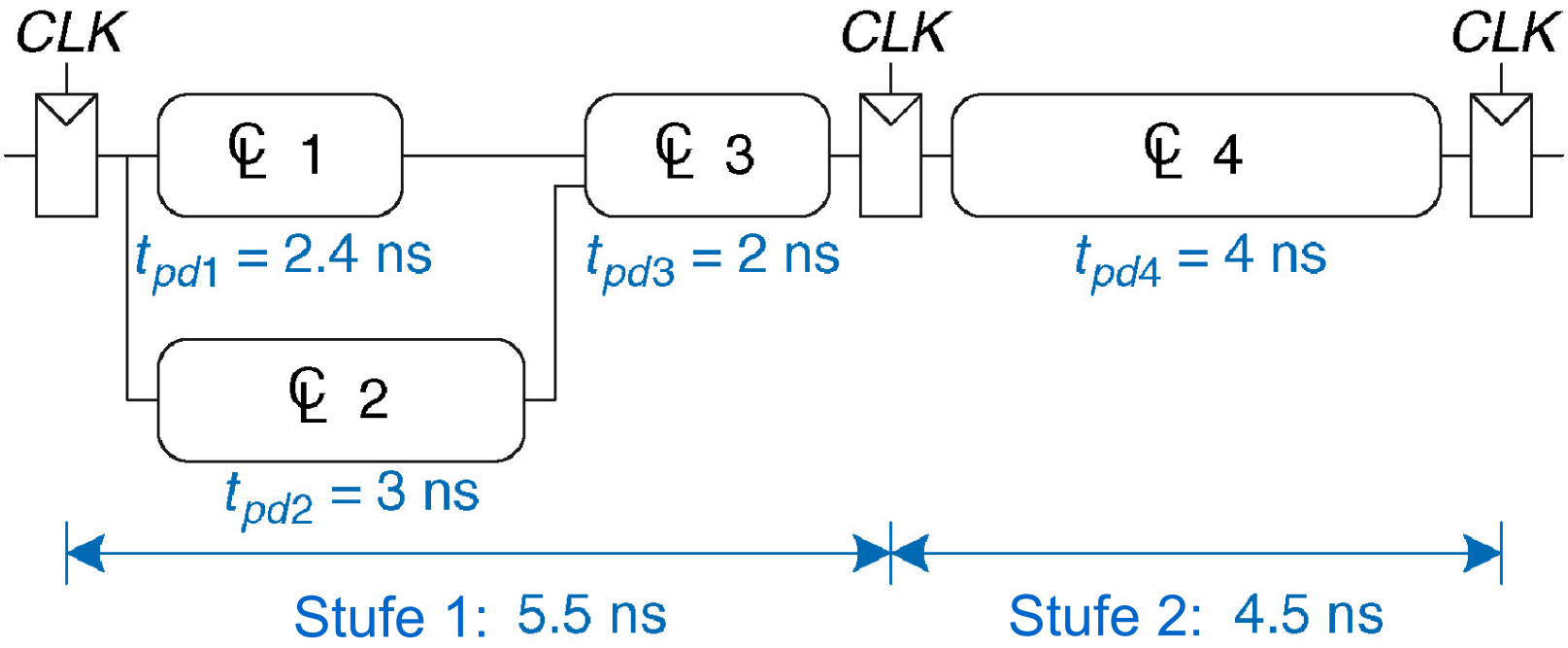
- Durchsatz = 8 Bleche/h

Schaltung ohne Pipelining



- Kritischer Pfad durch Elemente 2, 3, 4: 9 ns
- t_{setup} = 0,2 ns und t_{pcq}=0,3 ns ➔ T_c = 9+0,2+0,3 = 9,5 ns
- Latenz = 9,5ns ; Durchsatz = 1 / 9,5ns = 105 MHz

Schaltung mit zweistufiger Pipeline

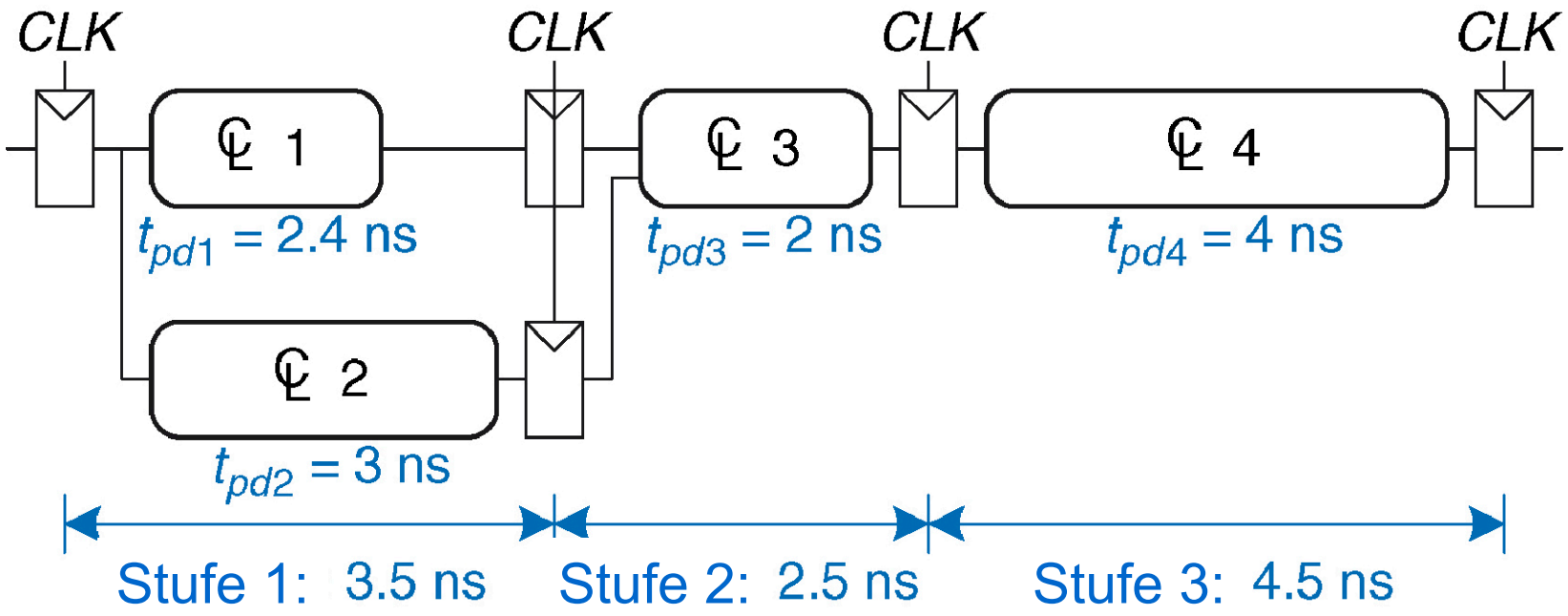


- Stufe 1: 3+2+0,2+0,3 = 5,5 ns
- ➔ T_c = 5,5 ns
- Latenz = 2 Takte = 11 ns
- Durchsatz = 1 / 5,5 ns = 182 MHz

Stufe 2: 4+0,2+0,3 = 4,5 ns

LEHRE WIKI TEST IN ZWEI FOLIEN

Schaltung mit dreistufiger Pipeline



- T_c = 4,5ns
- Latenz = 3 Takte = 13,5 ns
- Durchsatz = 1 / 4,5 ns = 222 MHz

Diskussion Pipelining

TECHNISCHE UNIVERSITÄT DARMSTADT

- Mehr Pipelinestufen
  - Höherer Durchsatz (mehr Ergebnisse pro Zeiteinheit)
  - aber auch höhere Latenz (länger warten auf das erste Ergebnis)
  - ➔ Lohnt sich nur, wenn viele Datensätze bearbeitet werden müssen

- Klappt aber nicht immer
- Problem: Abhängigkeiten

- Beispiel Kekse: Erstmal schauen wie ein Blech geworden ist, bevor das nächste bestückt wird

- Wird noch intensiv im 7. Kapitel behandelt (Prozessorarchitektur)

LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!