

Digitaltechnik

Wintersemester 2017/2018

9. Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT





1. Einleitung
2. Zeitverhalten synchroner sequentieller Logik
3. Parallelität
4. Zusammenfassung

Einleitung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1100010110001000001100111011101000110011
1010001000010011100011111011110100001111
0001001110100000110001111111000100100001
1100001000100001111110110101110111100011
1111001111101010000010101111100111111000
0101111101001110010000001100000100011010
1111001011111010101110100110101110000110
011000010010111110000111100011111011011
1010100100000101110001000010000100111010
1101001100010111100011101101011101010111
1111110111111000100100111001011011110011
0010001101010001110000101001101101000001
0000010001100001110001110100110001100110
1011010001100000010001010100011101010010
0001011101110000110011010111011100011101
11101001010001111000100111111111001001111



- ▶ Zusammenlegung von Übungsgruppen ab KW 51 geplant
 - ▶ Tutoren-Zuordnung bleibt erhalten
 - ▶ genaue Infos bis Ende KE50 im Moodle

Rückblick auf die letzten Vorlesungen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ **Sequentielle Logik**
 - ▶ Sequentielle Schaltungen
 - ▶ Speicherelemente
 - ▶ Synchrone sequentielle Logik
- ▶ **Endliche Zustandsautomaten**
 - ▶ Konzept, Notationen und Anwendungsbeispiele
 - ▶ Moore vs. Mealy
 - ▶ Zerlegen von Zustandsautomaten

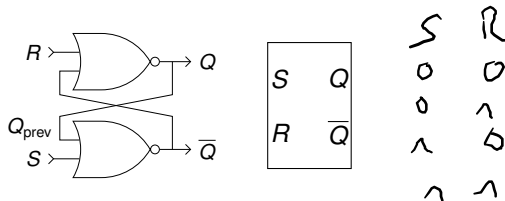


Harris 2013
Kap. 3.1 - 3.4

Wiederholung: Speicherelemente



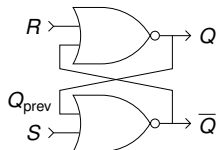
TECHNISCHE
UNIVERSITÄT
DARMSTADT



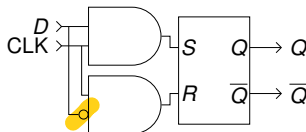
Wiederholung: Speicherelemente



TECHNISCHE
UNIVERSITÄT
DARMSTADT



S	Q
R	\overline{Q}



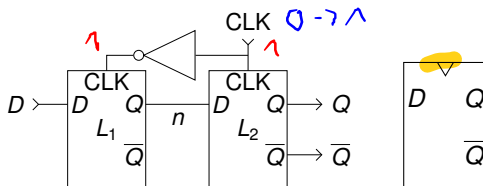
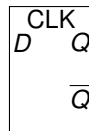
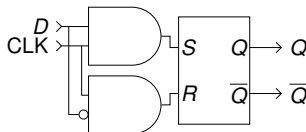
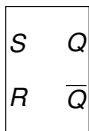
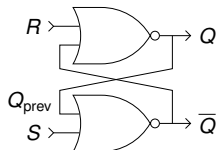
CLK	Q
D	\overline{Q}

CLK D
0 1
1 0
1 1

Wiederholung: Speicherelemente

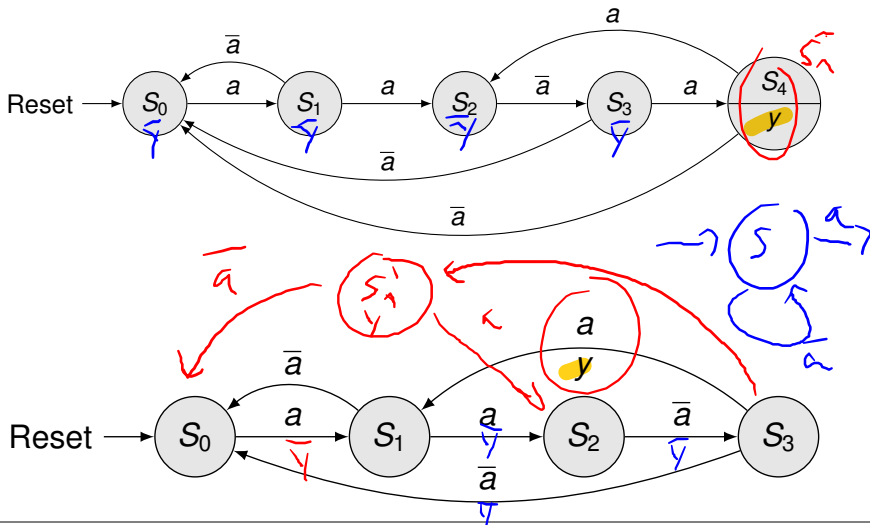


TECHNISCHE
UNIVERSITÄT
DARMSTADT



Wiederholung: Endliche Automaten

Notation, Überführung



Wiederholung: Endliche Automaten

Don't Cares



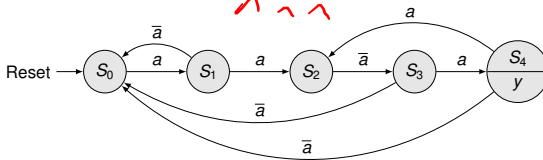
TECHNISCHE
UNIVERSITÄT
DARMSTADT

S	a	S'
S_0	0	S_0
S_0	1	S_1
S_1	0	S_0
S_1	1	S_2
S_2	0	S_3
S_2	1	S_2
S_3	0	S_0
S_3	1	S_4
S_4	0	S_0
S_4	1	S_2

S	s_2	s_1	s_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0

Handwritten red marks indicating don't care conditions in the state transition table:

- From S_2 to S_1 on input a (marked with a red checkmark)
- From S_3 to S_2 on input a (marked with a red checkmark)
- From S_4 to S_3 on input a (marked with a red checkmark)
- From S_0 to S_2 on input \bar{a} (marked with a red checkmark)
- From S_1 to S_3 on input \bar{a} (marked with a red checkmark)
- From S_2 to S_4 on input \bar{a} (marked with a red checkmark)

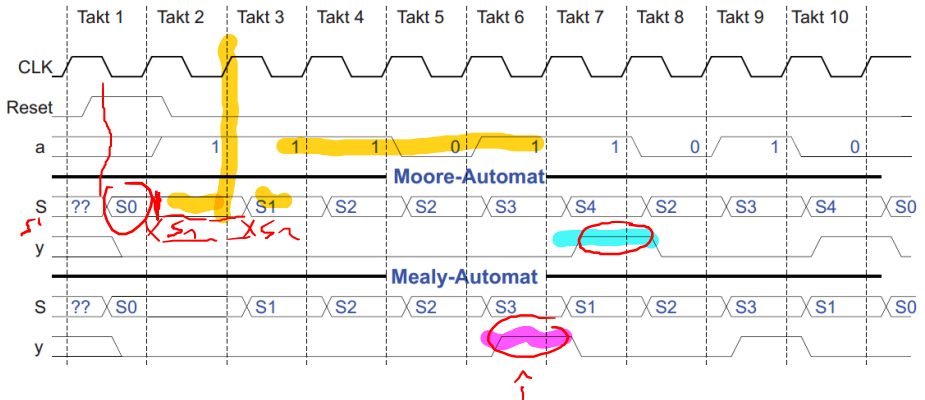


pattern/moore/state.esp

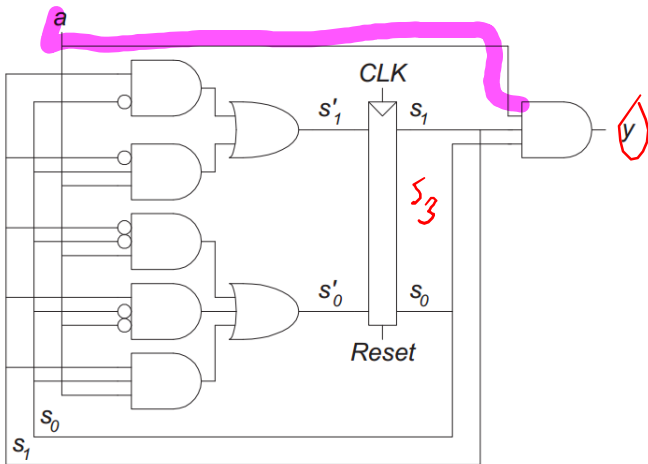
1	.i	4
2	.o	3
3	0000	000
4	0001	001
5	0010	000
6	0011	010
7	0100	011
8	0101	010
9	0110	000
10	0111	100
11	1000	000
12	1001	010
13	1010	---
14	1011	---
15	1100	---
16	1101	---
17	1110	---
18	1111	---

Wiederholung: Endliche Automaten

Mealy vs. Moore



Mealy vs. Moore

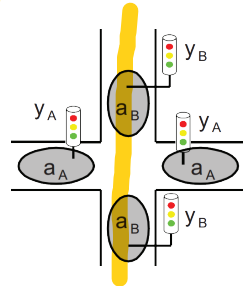
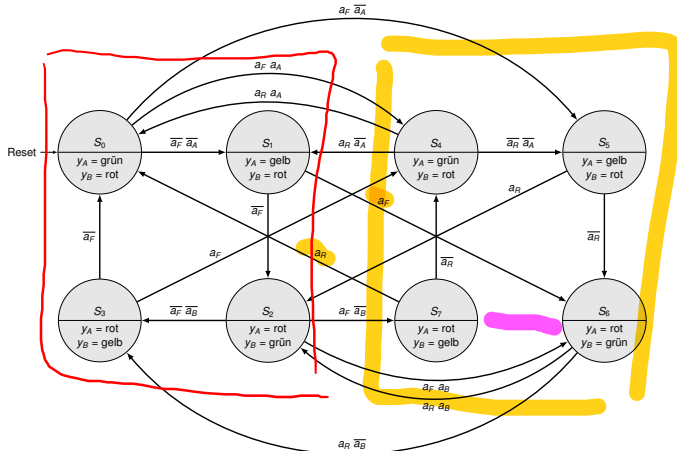


Wiederholung: Endliche Automaten

Zerlegung

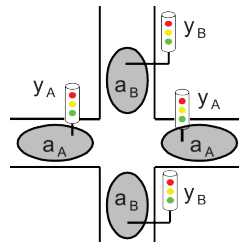
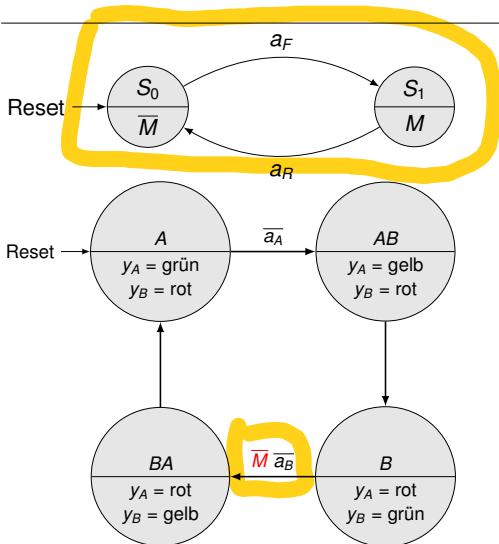


TECHNISCHE
UNIVERSITÄT
DARMSTADT



Wiederholung: Endliche Automaten

Zerlegung



Überblick der heutigen Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Zeitverhalten synchroner sequentieller Schaltungen
- ▶ Parallelität



Harris 2013
Kap. 3.5-3.6
Seite 133 - 153

Zeitverhalten synchroner sequentieller Logik

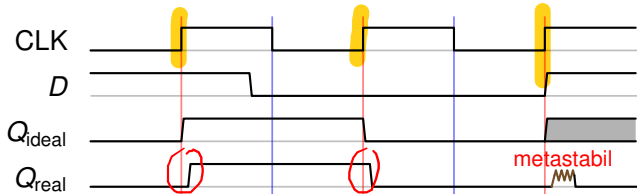
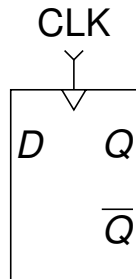


TECHNISCHE
UNIVERSITÄT
DARMSTADT

0010100110100011011100100101000010100100
1001000011111100111100111011111010010110
1011000100011000110001101010110001100110
0011001110111110010011100001000000110011
1110001101110111100001101001101101100111
1111110011111111011101001111010110010011
0010111001011000100010011101111101000001
01010010101101001100100010000011010111
1001011010010001110010111011010000000100
1111100110111111011100110010001011110011
1000001110101001011101001001100110111110
1100100101101001010101110101100001001100
0110010010001000111110010001010101111000
1011011100001000001100101001101000110010
0001000000011111000000100001010011010000
10101101111111110010000011011110111111000

Zeitverhalten synchroner sequentieller Logik

- ▶ Flip-Flop übernimmt D zur **steigenden Taktflanke**
- ▶ Was passiert bei **zeitgleicher** Änderung von D und CLK?
- ▶ bisher vereinfachte Annahme:
 - ▶ Wert **unmittelbar vor der Taktflanke** wird übernommen
- ▶ Aber:
 - ▶ Was heißt „**unmittelbar**“?
 - ▶ Wie **schnell wird neuer Zustand** am Ausgang sichtbar?
 - ▶ Was muss daher bei synchronen sequentiellen Schaltungen beachtet werden?

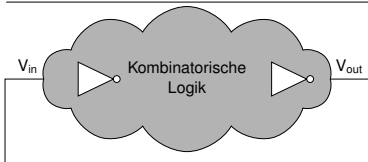


Analogie zur Fotografie: Bewegungsunschärfe (Bewegung während Belichtung)

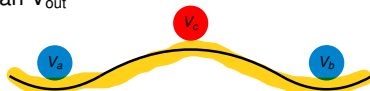
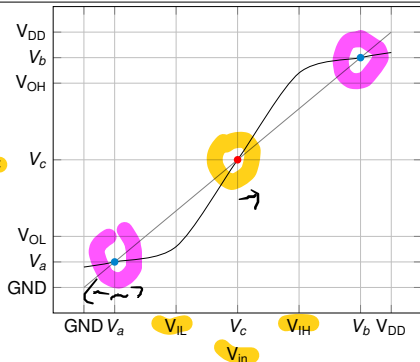


Quelle: <http://lightwatching.de>

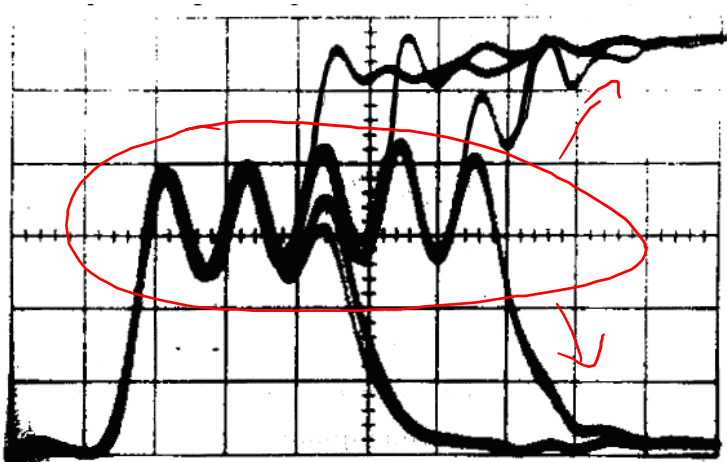
Metastabilität durch Rückkopplung kombinatorischer Logik



- ▶ Rückkopplung stabil für $V_{out} = V_{in}$
 - ▶ Transferfunktion schneidet Hauptdiagonale in
 - V_a repräsentiert 0
 - V_b repräsentiert 1
 - V_c im „verbotenen“ Spannungsbereich
 - ▶ V_c ist „metastabil“, da
 - ▶ kleine Änderung an $V_{in} \rightarrow$ große Änderung an V_{out}
- ⇒ geht nach zufälliger Verzögerung in einen stabilen Zustand über



Metastabilität an Flip-Flops



Quelle: Thomas. J. Chaney, Washington University

Zeitanforderungen an DFF Eingangssignale



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Dateneingang D muss in Abtast-Zeitfenster um Taktflanke stabil sein, um Metastabilität zu vermeiden

t_{setup}

Zeitintervall vor Taktflanke, in dem D stabil sein muss

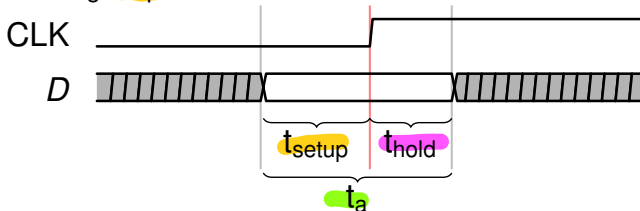
t_{hold}

Zeitintervall nach Taktflanke, in dem D stabil sein muss

t_a

Abtastzeitfenster: $t_a = t_{\text{setup}} + t_{\text{hold}}$

- ▶ Größenordnung: 10 ps

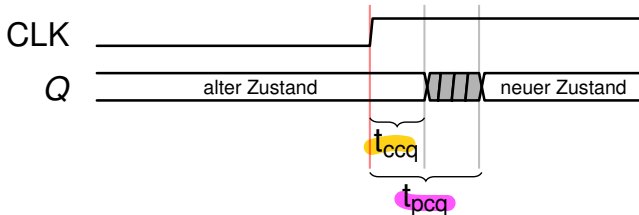


Zeitcharakteristik der DFF Ausgangssignale



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Verzögerung des Registerausgangs relativ zur steigenden Taktflanke
 - t_{ccq} Kontaminationsverzögerung, bis Q (frühestens) umschaltet
 - t_{pcq} Laufzeitverzögerung, bis Q (spätestens) stabil
- ▶ Größenordnung: 10 ps

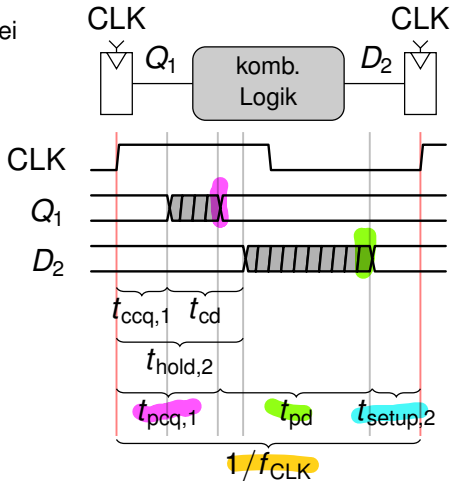


- ▶ kombinatorische Logik zwischen zwei Registern hat max/min Verzögerung
 - ▶ abhängig von Verzögerungen der Gatter und des *ersten* Registers
- ⇒ Timing-Bedingungen des *zweiten* Registers müssen erfüllt werden

- ▶ $t_{ccq,1} + t_{cd} > t_{hold,2}$
- ▶ $t_{pcq,1} + t_{pd} + t_{setup,2} \leq \frac{1}{f_{CLK}}$

- ⇒ maximale Taktrate wird durch **kritischen Pfad** bestimmt

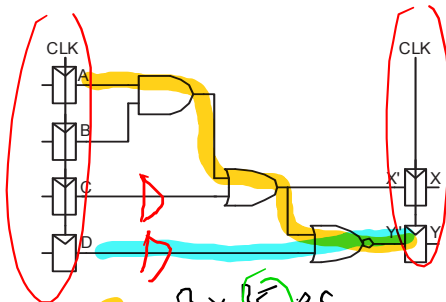
- ▶ $f_{CLK} \leq \frac{1}{t_{pcq} + t_{pd} + t_{setup}}$



Analyse des Zeitverhaltens



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Verzögerungsangaben

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 50 \text{ ps}$$

$$t_{setup} = 60 \text{ ps}$$

$$t_{hold} = 70 \text{ ps}$$

Pro Gatter

$$t_{pd} = 35 \text{ ps}$$

$$t_{cd} = 25 \text{ ps}$$

$$t_{pd} = 3 \times 35 \text{ ps}$$

$$t_{cd} = 1 \times 25 \text{ ps}$$

Einhalten von Setup-Zeit Anforderung: Einhalten von Hold-Zeit Anforderung:

$$T_c \geq 5 \text{ ps} + 35 \text{ ps} + 60 \text{ ps}$$

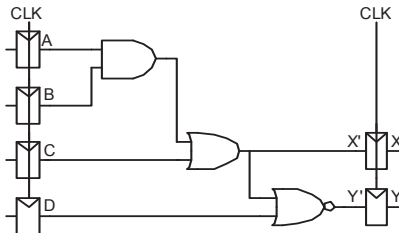
$$f_c = 1/T_c =$$

$$145$$

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

$$30 \text{ ps} + 25 \text{ ps} = 55 \text{ ps}$$

Analyse des Zeitverhaltens



$$t_{pd} = 3 \times 35 \text{ ps} = 105 \text{ ps}$$

$$t_{cd} = 25 \text{ ps}$$

Einhalten der Setup-Zeit-anforderung: Einhalten der Hold-Zeit-anforderung:

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

$$f_c = 1/T_c = 4,65 \text{ GHz}$$

Verzögerungsangaben

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 50 \text{ ps}$$

$$t_{setup} = 60 \text{ ps}$$

$$t_{hold} = 70 \text{ ps}$$

Pro Gatter

$$\left[\begin{array}{l} t_{pd} = 35 \text{ ps} \\ t_{cd} = 25 \text{ ps} \end{array} \right.$$

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

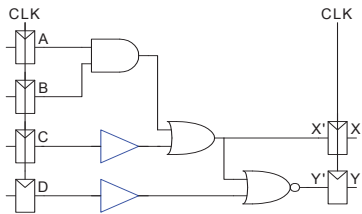
$$(30 + 25) \text{ ps} > 70 \text{ ps} ? \text{ Nein, verletzt!}$$

Beheben der verletzten Hold-Zeit Anforderung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Füge Puffer in zu kurze Pfade ein!



$$t_{pd} =$$

$$t_{cd} =$$

Einhalten der Setup-Zeit Anforderung: Einhalten der Hold-Zeit Anforderung:

$$T_c \geq$$

$$f_c =$$

Verzögerungsangaben

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 50 \text{ ps}$$

$$t_{\text{setup}} = 60 \text{ ps}$$

$$t_{\text{hold}} = 70 \text{ ps}$$

Pro Gatter

$$\left[\begin{array}{l} t_{pd} = 35 \text{ ps} \\ t_{cd} = 25 \text{ ps} \end{array} \right.$$

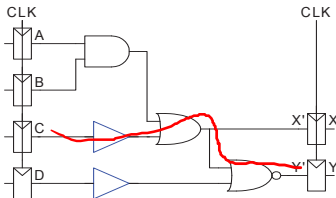
$$t_{ccq} + t_{cd} > t_{\text{hold}} ?$$

Beheben der verletzten Hold-Zeit Anforderung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Füge Puffer in zu kurze Pfade ein!



$$t_{pd} = 3 \times 35 \text{ ps} = 105 \text{ ps}$$

$$t_{cd} = 2 \times 25 \text{ ps} = 50 \text{ ps}$$

Einhalten der Setup-Zeit Anforderung: **Einhalten der Hold-Zeit Anforderung:**

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

$$f_c = 1/T_c = 4.65 \text{ GHz}$$

Verzögerungsangaben

$$t_{ccq} = 30 \text{ ps}$$

$$t_{pcq} = 50 \text{ ps}$$

$$t_{setup} = 60 \text{ ps}$$

$$t_{hold} = 70 \text{ ps}$$

Pro Gatter

$$\left[\begin{array}{l} t_{pd} = 35 \text{ ps} \\ t_{cd} = 25 \text{ ps} \end{array} \right.$$

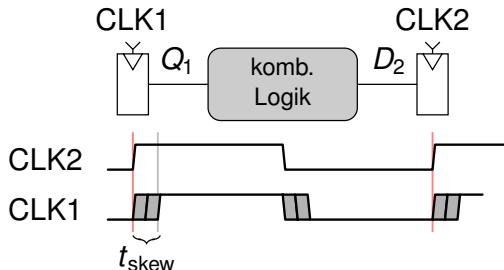
$$t_{ccq} + t_{cd} > t_{hold} ?$$

$$(30 + 50) \text{ ps} > 70 \text{ ps} ? \text{ Ja, eingehalten!}$$

Taktverschiebung (clock skew)



- ▶ Takt kommt nicht bei allen Registern gleichzeitig an
 - ▶ unterschiedliche Verdrahtungswege auf dem Chip (clock tree),
 - ▶ Logik in Taktsignal (bspw. gated clock)
- ▶ t_{skew} ist max. Differenz der Taktankunftszeit zwischen zwei Registern



Timing-Bedingungen mit Taktverschiebung



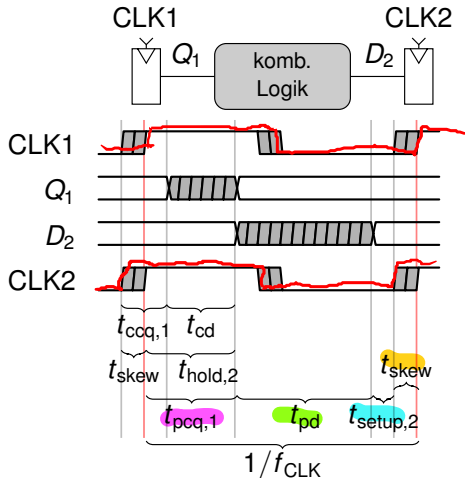
TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Timing-Bedingungen müssen auch im worst-case eingehalten werden:

$$\begin{aligned} & \bullet t_{ccq,1} + t_{cd} > t_{skew} + t_{hold,2} \\ & \bullet t_{pcq,1} + t_{pd} + t_{setup,2} + t_{skew} \leq \frac{1}{f_{CLK}} \end{aligned}$$

⇒ idR. wird Timing durch t_{skew} enger

- ▶ Taktfrequenz kann durch t_{skew} auch steigen, wenn CLK2 sicher nach CLK1 schaltet



Verletzung der dynamischen Entwurfsdisziplin



TECHNISCHE
UNIVERSITÄT
DARMSTADT

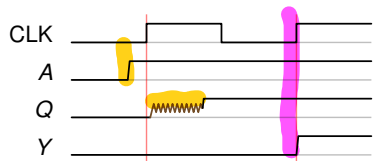
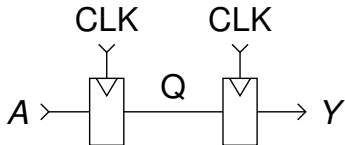
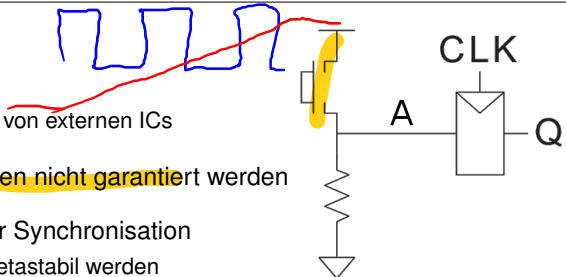
▶ asynchronen Eingänge:

- ▶ Benutzereingaben
- ▶ Kommunikationssignale von externen ICs

⇒ Timing-Bedingungen können nicht garantiert werden

▶ zweifaches Shiftregister für Synchronisation

- ▶ erstes Flip-Flop kann metastabil werden
- ▶ kippt idR vor nächster Taktflanke in stabilen Zustand
- ▶ zweites Flip-Flop wird nicht metastabil



Parallelität



TECHNISCHE
UNIVERSITÄT
DARMSTADT

0000010111100100101101110110011100100011
1010100010101110001101110111010011100101
1100010101000011011010010110100111001000
0101001110100010010101100100111001010100
1001010011111000011100001110001110000010
0110110110100011010110000001001100011110
0001001100000010101001110111111000100011
1110010111000010010011101111011110101010
0101101010100001011101100011111101111001
0000110010010001100011011001001011011000
0101101110100010001101000010101101011001
0110001010100000000000100011110111110000
1100111001011001010000010001000110010000
1111110111110010011010100100011101011110
0101111101111101011010011001010011011010
1011111101100110110001110001001101101111



- ▶ **räumliche** Parallelität
 - ▶ mehrere Aufgaben durch vervielfachte Hardware gleichzeitig bearbeiten
 - ▶ **zeitliche** Parallelität
 - ▶ Aufgabe in mehrere Unteraufgaben aufteilen
 - ▶ Unteraufgaben parallel ausführen
 - ▶ Beispiel: Fließbandprinzip bei Autofertigung („**Pipelining**“)
 - ▶ nur eine Station für **pro Arbeitsschritt**
 - ▶ alle **unterschiedlichen Arbeitsschritte** für mehrere Autos werden parallel ausgeführt
- ⇒ **zeitliche Parallelität**



- ▶ **Datensatz:**
 - ▶ Vektor aus Eingabewerten, die zu Vektor aus Ausgabewerten bearbeitet werden
- ▶ **Latenz:**
 - ▶ Zeit von der Eingabe eines Datensatzes bis Ausgabe der zugehörigen Ergebnisse
- ▶ **Durchsatz**
 - ▶ 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 = 2 min
 - ▶ Durchsatz = $3/h$

Beispiel Parallelität

Plätzchen backen

- ▶ 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)

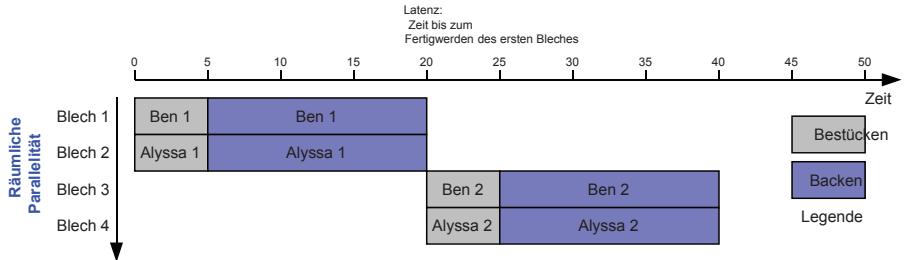
- ▶ gleiche Annahmen wie eben
 - ▶ 5 Minuten Blech bestücken, 15 Minuten Backen
- ▶ Alternative Vorgehensweisen
 - ▶ **Räumliche Parallelität**: Zwei Bäcker (Ben & Alyssa), jeder mit einem eigenen Ofen
 - ▶ **Zeitliche Parallelität**: Aufteilen der Keksherstellung in Unteraufgaben
 - ▶ Blech bestücken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

⇒ Latenz und Durchsatz?

Räumliche Parallelität



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Latenz =

20 min

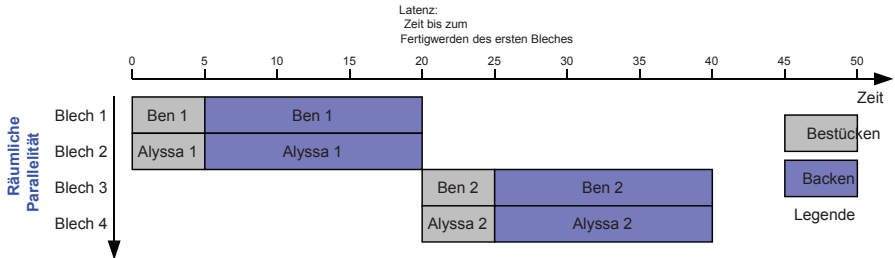
Durchsatz =

6/h

Räumliche Parallelität



TECHNISCHE
UNIVERSITÄT
DARMSTADT



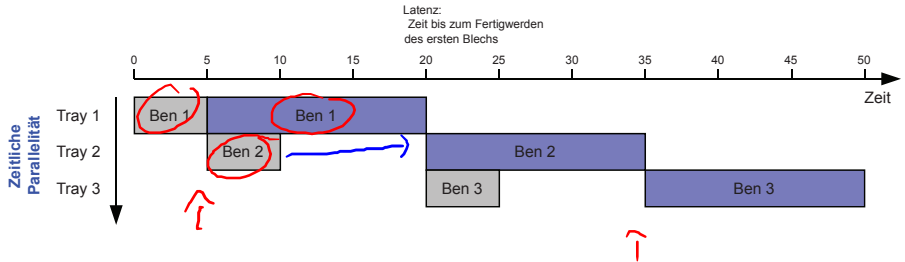
$$\text{Latenz} = 5 + 15 = 20 \text{ Minuten} = 1/3 \text{ h}$$

$$\text{Durchsatz} = 2 \text{ Bleche alle } 20 \text{ Minuten} = 6 \text{ Bleche/h}$$

Zeitliche Parallelität



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Latenz =

30min

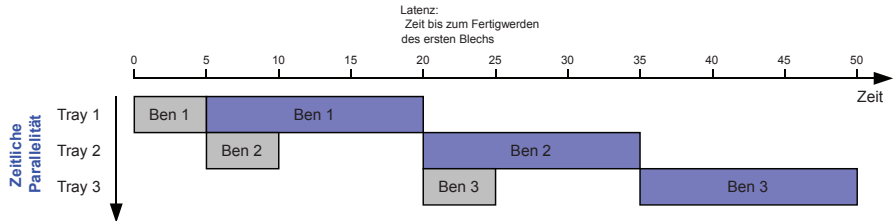
Durchsatz =

4h

Zeitliche Parallelität



TECHNISCHE
UNIVERSITÄT
DARMSTADT



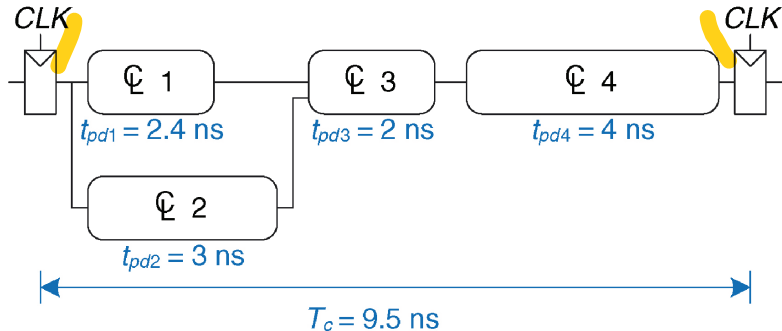
Latenz = 15 + 15 = 30 Minuten = 1/2 h

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



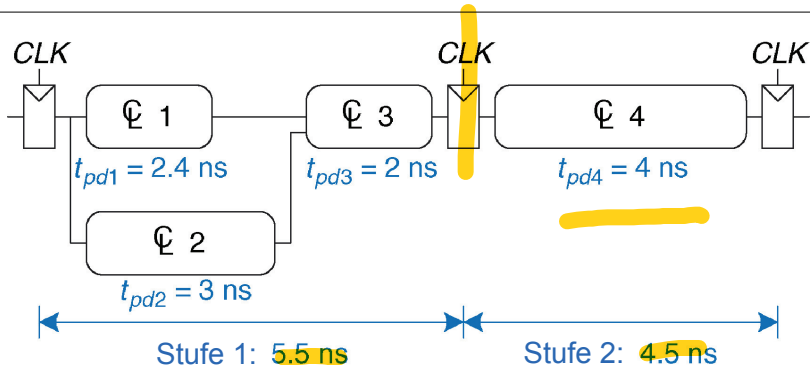
- ▶ Zeitliche und räumliche Parallelität können **miteinander** kombiniert werden
- ▶ Hier:
 - ▶ Zwei Bäcker und Öfen
 - ▶ Nächstes Blech bestücken während altes gebacken wird
- ▶ Latenz = 30 Minuten
- ▶ Durchsatz = 8 Bleche/h

Schaltung ohne Pipelining



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

Schaltung mit (mehrstufiger) Pipeline



▪ Stufe 1: $3 + 2 + 0,2 + 0,3 = 5,5 \text{ ns}$

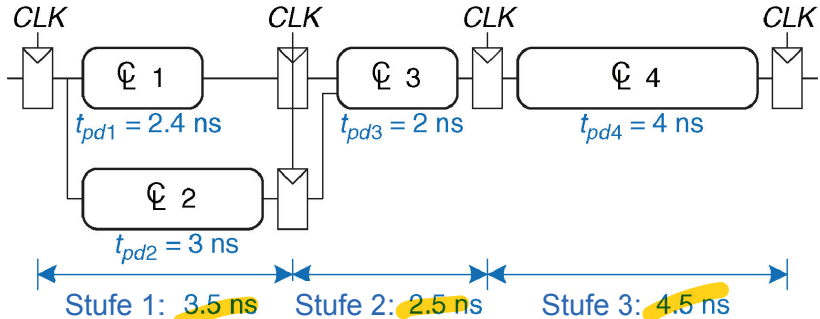
Stufe 2: $4 + 0,2 + 0,3 = 4,5 \text{ ns}$

▪ $\rightarrow T_c = 5,5 \text{ ns}$

▪ Latenz = 2 Takte = 11 ns

▪ Durchsatz = $1 / 5,5 \text{ ns} = 182 \text{ MHz}$

Schaltung mit (mehrstufiger) Pipeline



- $T_c = 4.5 \text{ ns}$
- Latenz = 3 Takte = 13.5 ns
- Durchsatz = $1 / 4.5 \text{ ns} = 222 \text{ MHz}$

- ▶ mehr Pipelinestufen
 - ▶ höherer Durchsatz (mehr Ergebnisse pro Zeiteinheit)
 - ▶ aber auch höhere Latenz (länger warten auf das 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 beim Thema Befehlsverarbeitung von Prozessoren behandelt

Zusammenfassung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

0100001000100100100000001111001010100111
0011011101011001101001101011111011111100
1001000000011101011100011000101011101100
0010101101001001010001011101001101100000
0111110001000010110110110111100001101110
1011010000011100101101110010111000101111
1111011001011101011001000000000011010010
101001100001010101010100110100111011100110
00100000110111000000001111111111100110110
0011010110110001001101000010101111000101
1001111111010101000001111000000111100111
1111010001011100111011101110000100100111
0001100101011000111011110001011010110101
0101110000010110100111101110110111111101
0000111010101000001101010100101010111101
0100000101110101011110100100110110100011



- ▶ Zeitverhalten synchroner sequentieller Schaltungen
- ▶ Parallelität

- ▶ Nächste Vorlesung behandelt
 - ▶ Hardwarebeschreib mit SystemVerilog