Digitaltechnik
Wintersemester 2017/2018
7. Vorlesung

TECHNISCHE UNIVERSITÄT DARMSTADT

Inhalt

TECHNISCHE UNIVERSITÄT DARMSTADT

Einleitung

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches

TECHNISCHE UNIVERSITÄT DARMSTADT

- `https://nabla.algo.informatik.tu-darmstadt.de/`
  (Rechenbeispiele für Zahlensysteme)

Organisatorisches

TECHNISCHE UNIVERSITÄT DARMSTADT

- `https://nabla.algo.informatik.tu-darmstadt.de/`
  (Rechenbeispiele für Zahlensysteme)

- Vorgesehene Bearbeitungszeit für Übungen

Organisatorisches

TECHNISCHE UNIVERSITÄT DARMSTADT

- `https://nabla.algo.informatik.tu-darmstadt.de/`
  (Rechenbeispiele für Zahlensysteme)
- Vorgesehene Bearbeitungszeit für Übungen
- Espresso Binär- und Beispieldateien verfügbar

Organisatorisches

TECHNISCHE UNIVERSITÄT DARMSTADT

- `https://nabla.algo.informatik.tu-darmstadt.de/`
  (Rechenbeispiele für Zahlensysteme)
- Vorgesehene Bearbeitungszeit für Übungen
- Espresso Binär- und Beispieldateien verfügbar
- Vertretung durch Raad Bahmani für V8

Rückblick auf die letzten Vorlesungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kombinatorische Logik
  - Boole'sche Gleichungen
  - Boole'sche Algebra
  - Bubble Pushing
  - Logik-Realisierung mit Basis-Gattern
  - Karnaugh Diagramme
  - Algorithmische Logikminimierung
  - Vierwertige Logik
  - Zeitverhalten

Harris 2013
Kap. 2

Wiederholung: Espresso
Minimierung eines Prioritätsencoders: $\mathbb{B}^4 \rightarrow \mathbb{B}^4$

TECHNISCHE UNIVERSITÄT DARMSTADT

- One-Hot-Encoding des höchsten gesetzten Eingangs
- Verwendet für
  - Bus-Arbitrierung
  - Interrupt-Steuerung

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Wiederholung: Espresso
Minimierung eines Prioritätsencoders: $\mathbb{B}^4 \rightarrow \mathbb{B}^4$

TECHNISCHE UNIVERSITÄT DARMSTADT

priorityencoder.esp

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

Wiederholung: Espresso
Minimierung eines Prioritätsencoders: $\mathbb{B}^4 \rightarrow \mathbb{B}^4$

TECHNISCHE UNIVERSITÄT DARMSTADT

priorityencoder.esp

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

espresso prioritycoder.esp

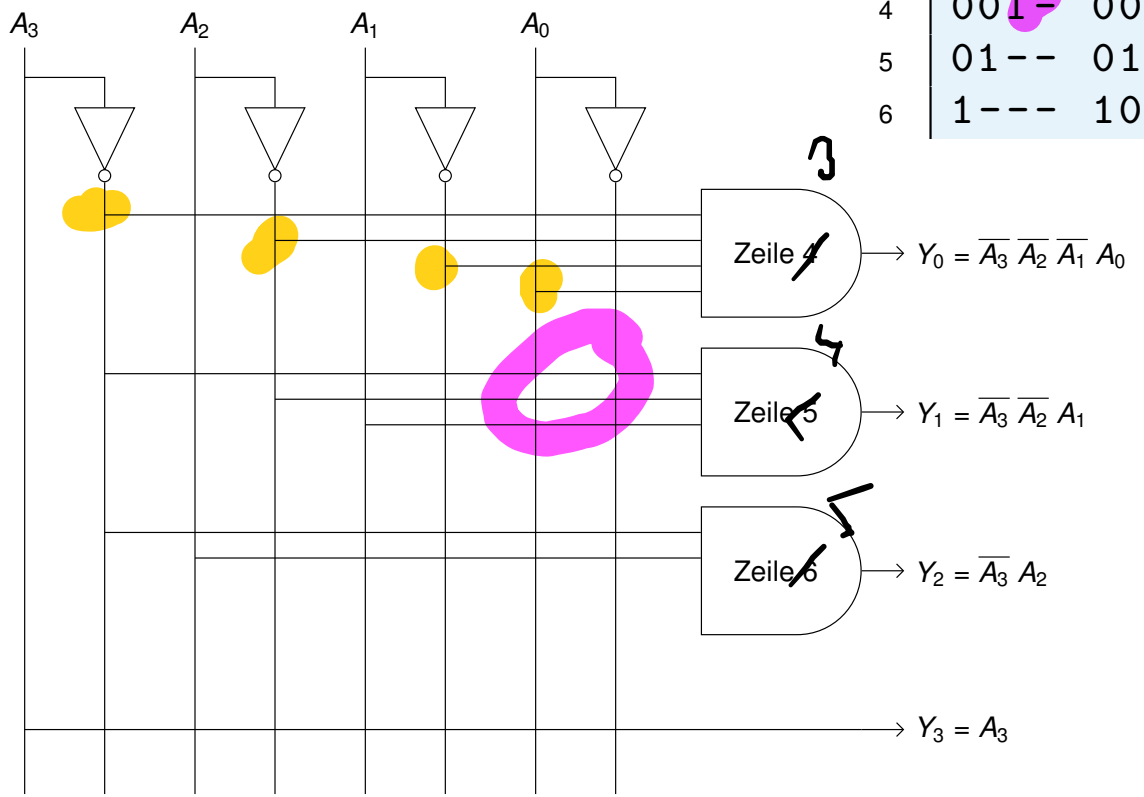
```
1  .i 4
2  .o 4
3  0001 0001
4  001- 0010
5  01-- 0100
6  1--- 1000
```

Wiederholung: Espresso
Minimierung eines Prioritätsencoders: $\mathbb{B}^4 \rightarrow \mathbb{B}^4$

TECHNISCHE UNIVERSITÄT DARMSTADT

priorityencoder.esp

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



espresso priorityencoder.esp

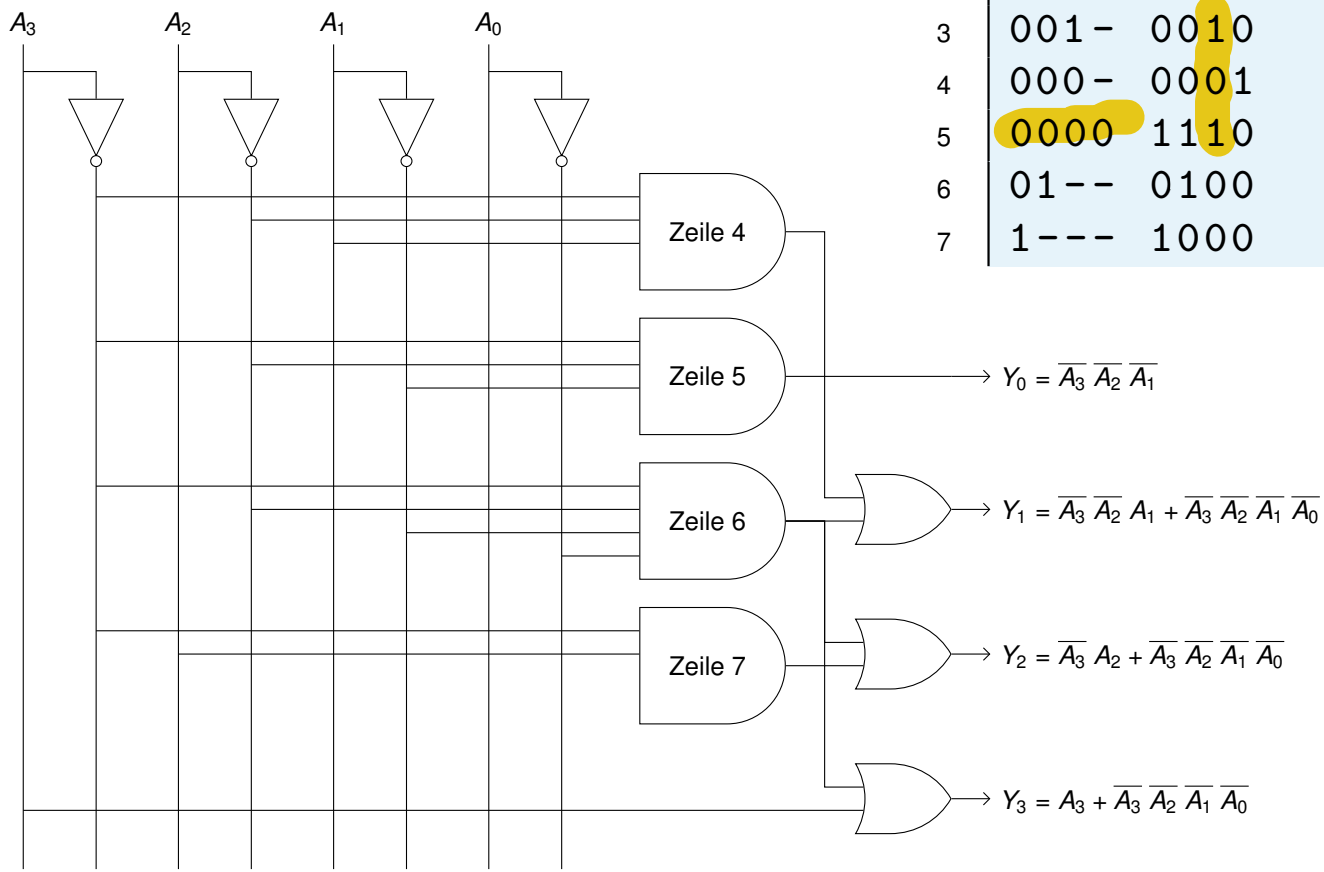
```
1  .i 4
2  .o 4
3  0001 0001
4  001- 0010
5  01-- 0100
6  1--- 1000
```

Wiederholung: Espresso
Minimierung eines Prioritätsencoders: $\mathbb{B}^4 \rightarrow \mathbb{B}^4$

TECHNISCHE UNIVERSITÄT DARMSTADT

`priorityencoder.mod.esp`

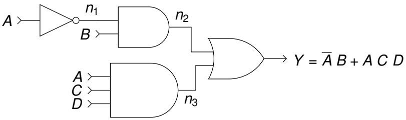
```
1  .i 4
2  .o 4
3  0000 1111
4  0001 0001
5  0010 0010
6  0011 0010
7  0100 0100
8  0101 0100
9  0110 0100
10 0111 0100
11 1000 1000
12 1001 1000
13 1010 1000
14 1011 1000
15 1100 1000
16 1101 1000
17 1110 1000
18 1111 1000
```



`espresso priorityencoder.mod.esp`

```
1  .i 4
2  .o 4
3  001- 0010
4  000- 0001
5  0000 1110
6  01-- 0100
7  1--- 1000
```

Wiederholung: Störimpulse
Abhängigkeit von Übergangsrichtung

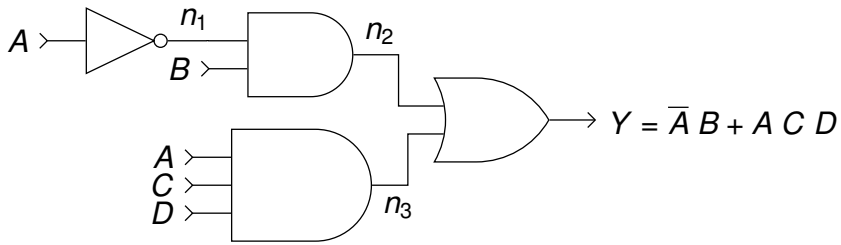


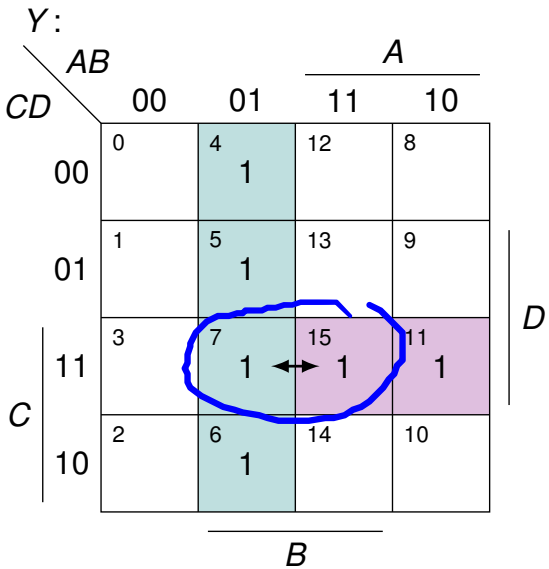
Y:

CD \ AB	00	01	11	10
00	0	4: 1	12	8
01	1	5: 1	13	9
11	3	7: 1	15: 1	11: 1
10	2	6: 1	14	10

(A: columns 11, 10; B: columns 01, 11; C: rows 11, 10; D: rows 01, 11)

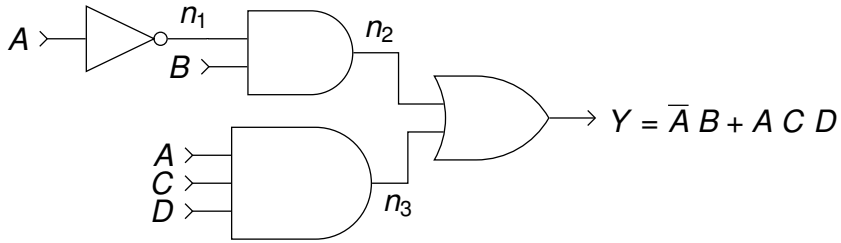
Wiederholung: Störimpulse
Abhängigkeit von Übergangsrichtung

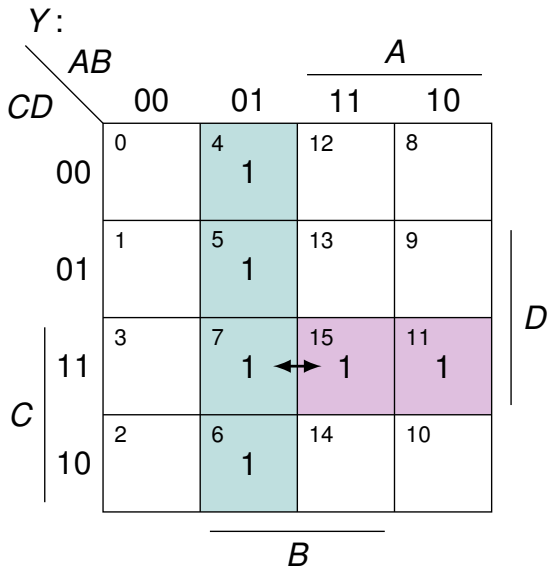




Wiederholung: Störimpulse
Abhängigkeit von Übergangsrichtung

TECHNISCHE UNIVERSITÄT DARMSTADT

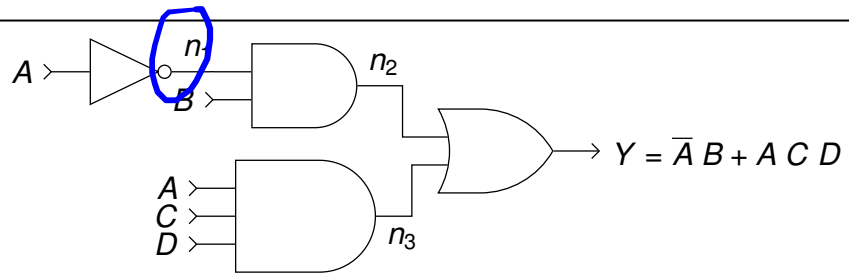


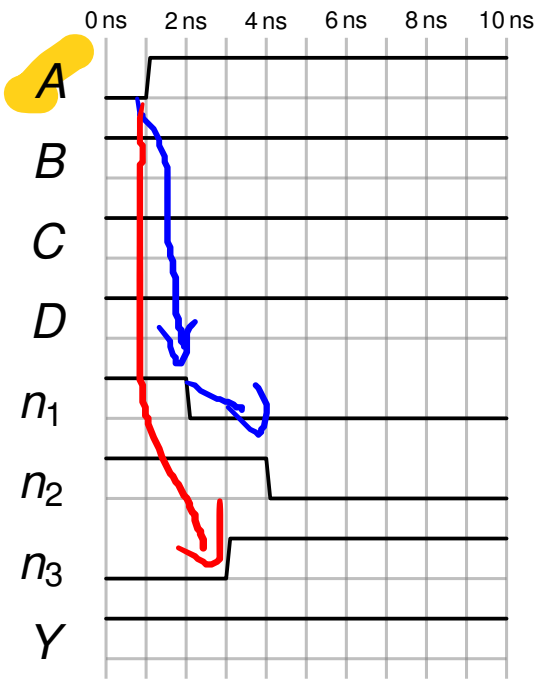


	NOT	AND	AND3	OR
t_{pd}	1 ns	2 ns	2 ns	3 ns
t_{cd}	1 ns	2 ns	2 ns	3 ns

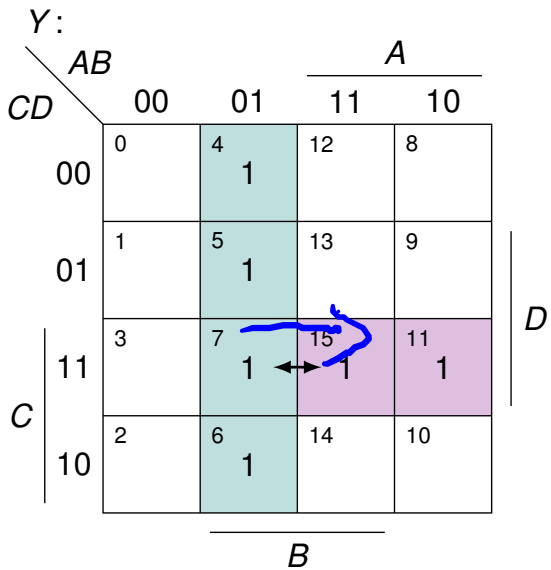
Wiederholung: Störimpulse
Abhängigkeit von Übergangsrichtung

TECHNISCHE UNIVERSITÄT DARMSTADT



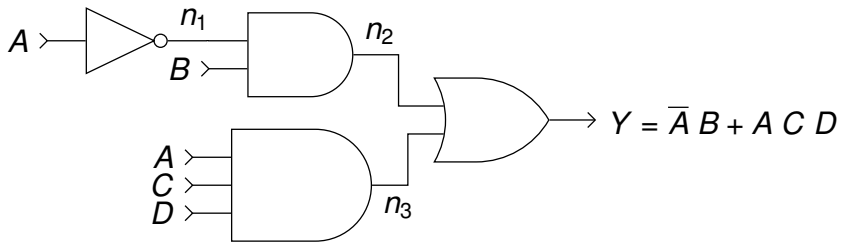


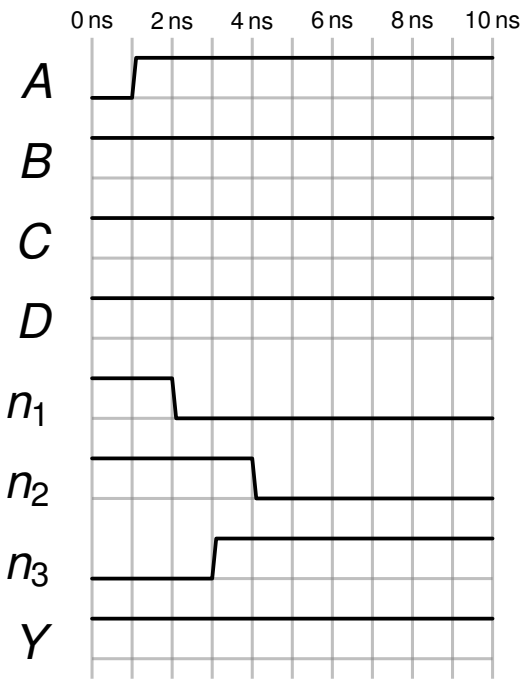
$(0111) \rightarrow (1111)$



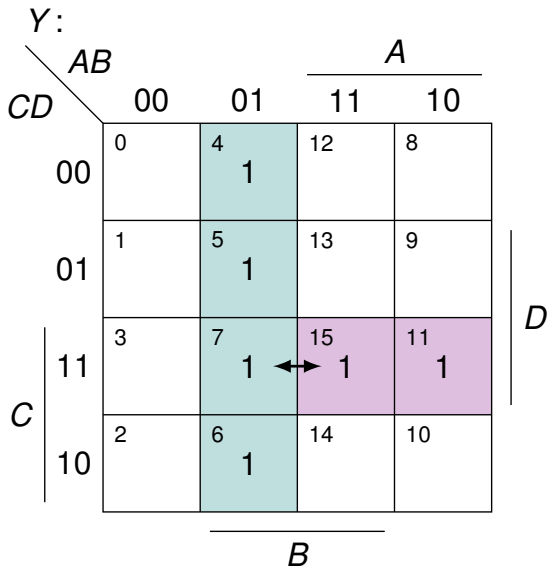
	NOT	AND	AND3	OR
t_{pd}	1 ns	2 ns	2 ns	3 ns
t_{cd}	1 ns	2 ns	2 ns	3 ns

Wiederholung: Störimpulse
Abhängigkeit von Übergangsrichtung

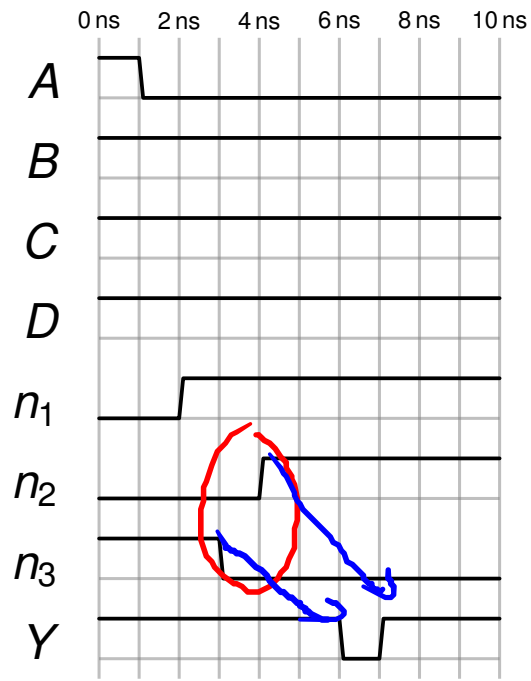




(0111) → (1111)



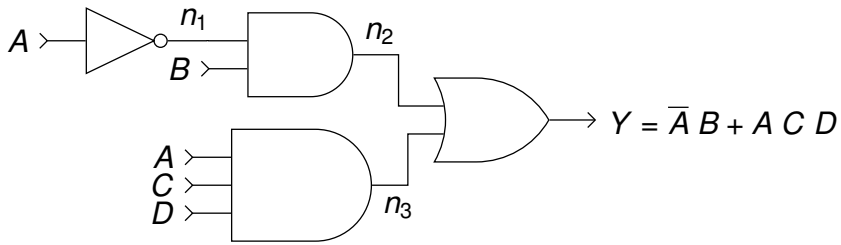
	NOT	AND	AND3	OR
t_{pd}	1 ns	2 ns	2 ns	3 ns
t_{cd}	1 ns	2 ns	2 ns	3 ns

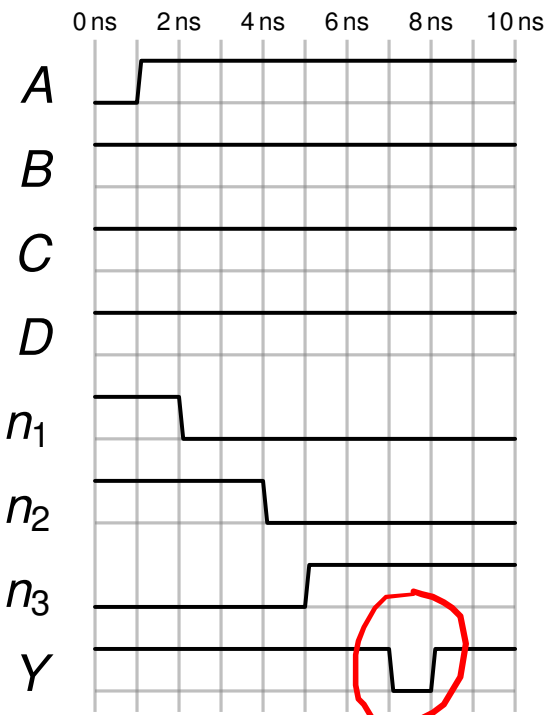


(1111) → (0111)

Wiederholung: Störimpulse
Abhängigkeit von Gatter-Verzögerung

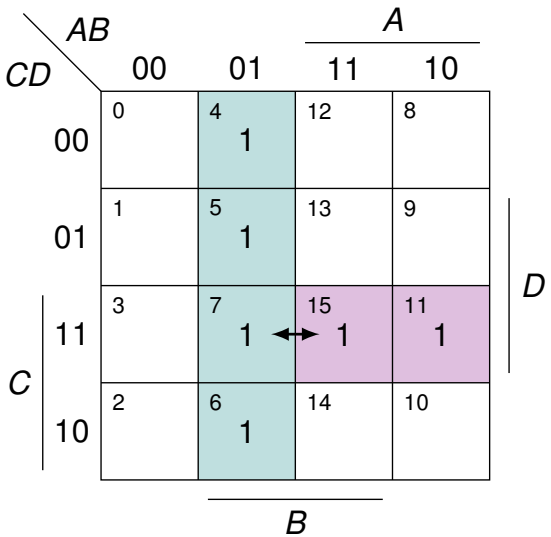
TECHNISCHE UNIVERSITÄT DARMSTADT



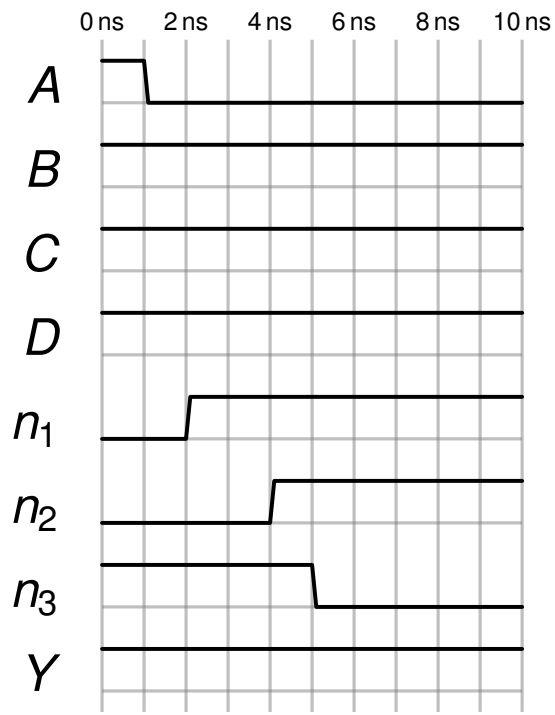


(0111) → (1111)

Y:



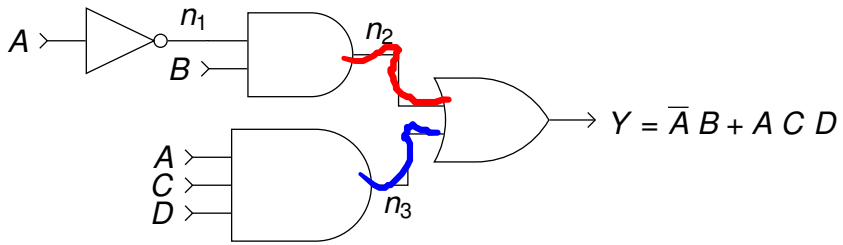
	NOT	AND	AND3	OR
t_{pd}	1 ns	2 ns	4 ns	3 ns
t_{cd}	1 ns	2 ns	4 ns	3 ns

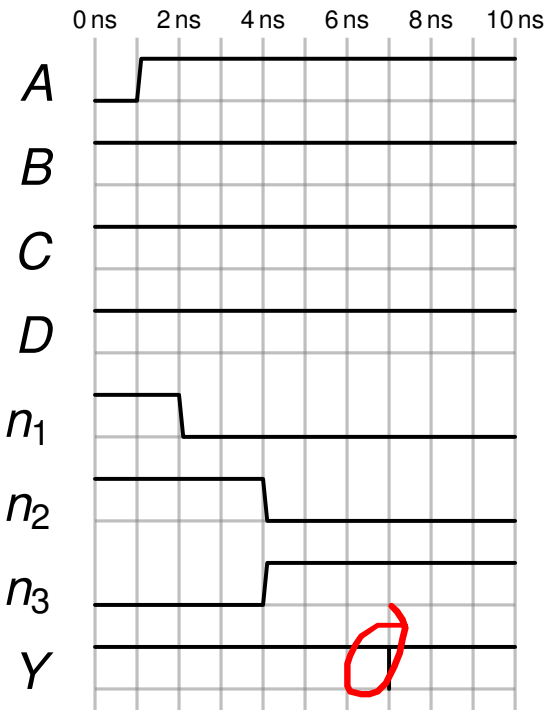


(1111) → (0111)

Wiederholung: Störimpulse
Abhängigkeit von Leitungsverzögerung

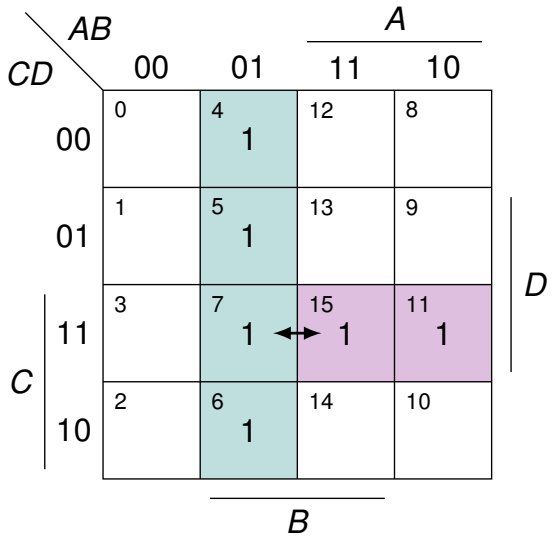
TECHNISCHE UNIVERSITÄT DARMSTADT



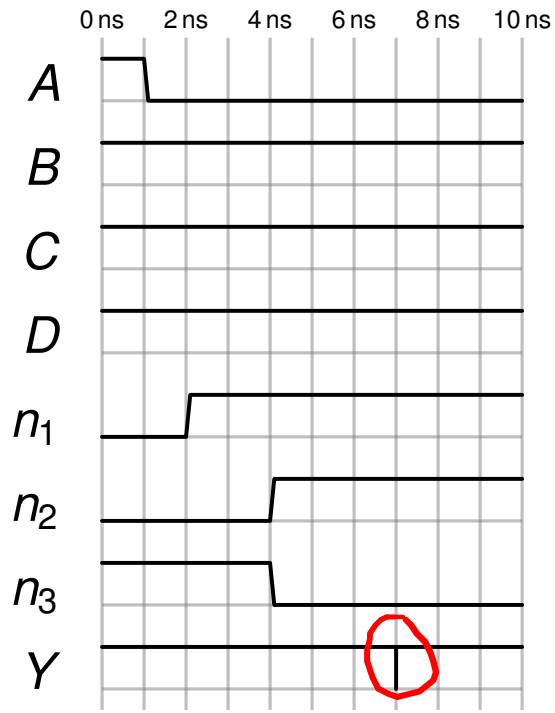


(0111) → (1111)

Y:



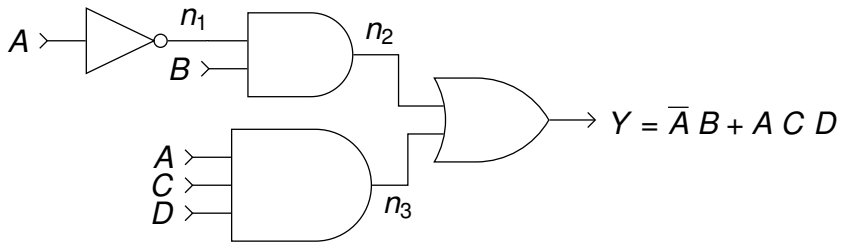
	NOT	AND	AND3	OR
t_{pd}	1 ns	2 ns	3 ns	3 ns
t_{cd}	1 ns	2 ns	3 ns	3 ns

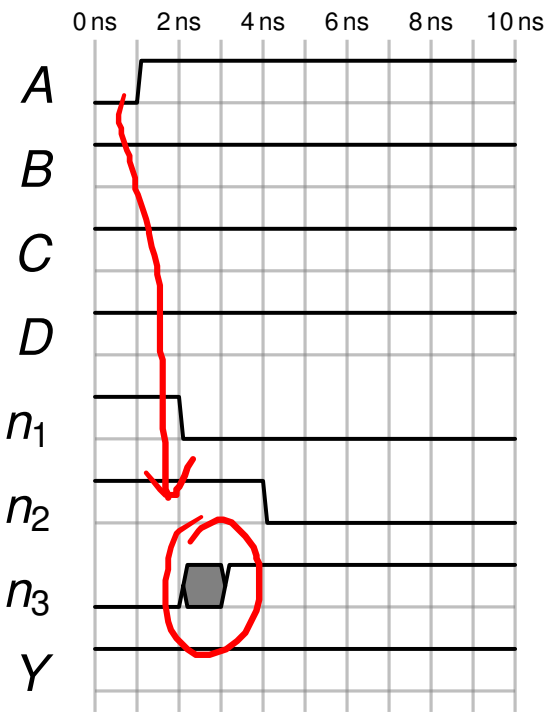


(1111) → (0111)

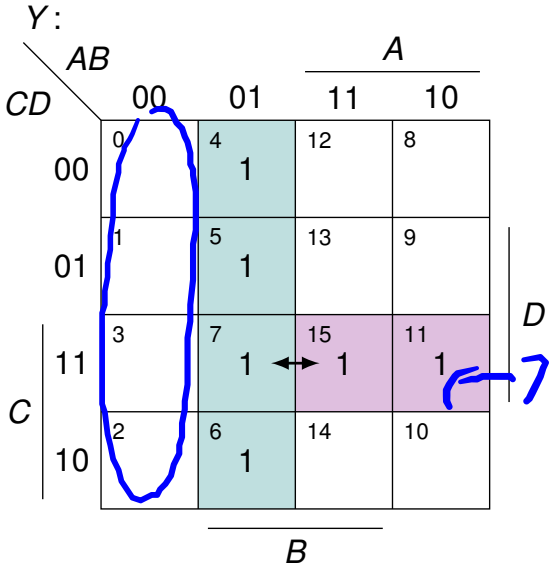
Wiederholung: Störimpulse
Ausbreitung von Unsicherheit

TECHNISCHE UNIVERSITÄT DARMSTADT

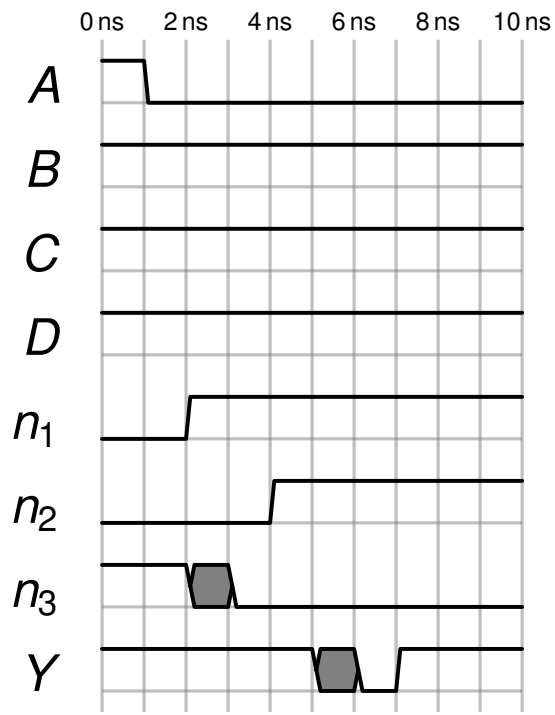




(0111) → (1111)



	NOT	AND	AND3	OR
t_{pd}	1 ns	2 ns	2 ns	3 ns
t_{cd}	1 ns	2 ns	1 ns	3 ns



(1111) → (0111)

Überblick der heutigen Vorlesung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sequentielle Logik
  - Sequentielle Schaltungen
  - Speicherelemente
  - Synchrone sequentielle Logik

Harris 2013
Kap. 3.1-3.3
Seite 103 - 117

Sequentielle Schaltungen

TECHNISCHE UNIVERSITÄT DARMSTADT

Sequentielle Schaltungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ausgänge hängen ab von
  - aktuellen Eingabewerten
  - vorherigen Eingabewerten

$\Rightarrow$ sequentielle Schaltung speichern *internen Zustand*

  - (Kurzzeit-)Gedächtnis repräsentiert Eingabesequenzen
  - realisiert durch Rückkoppplungen von Ausgängen zu Eingängen

  $\Rightarrow$ nicht kombinatorisch

- Warum reichen kombinatorische Schaltungen nicht aus?

Sequentielle Schaltungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ausgänge hängen ab von
  - aktuellen Eingabewerten
  - vorherigen Eingabewerten

$\Rightarrow$ sequentielle Schaltung speichern *internen Zustand*

- (Kurzzeit-)Gedächtnis repräsentiert Eingabesequenzen
- realisiert durch Rückkoppplungen von Ausgängen zu Eingängen

$\Rightarrow$ nicht kombinatorisch

- Warum reichen kombinatorische Schaltungen nicht aus?
  - (Zwischen-)Ergebnisse können nicht gespeichert / wiederverwendet werden
  - kritische Pfade können nicht beliebig lang / divers werden

$\Rightarrow$ Zeitverhalten bei sequentiellen Schaltungen besser kontrollierbar

Speicherelemente für sequentielle Schaltungen

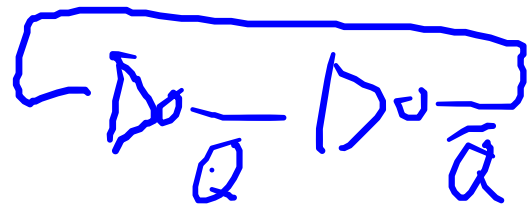
TECHNISCHE UNIVERSITÄT DARMSTADT

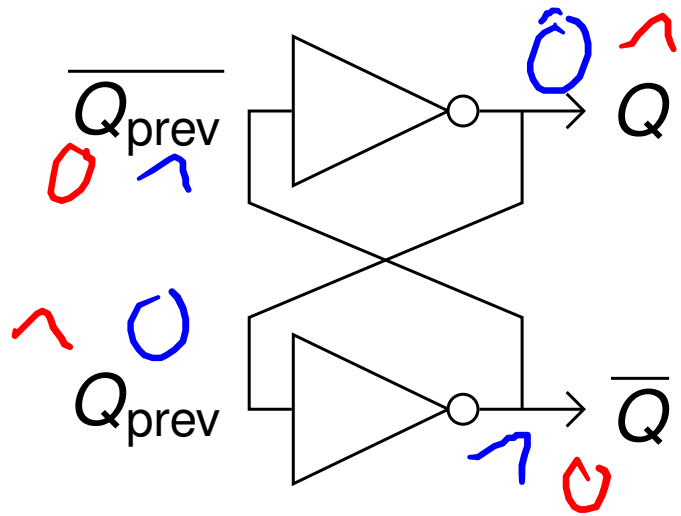
Bistabile Grundschaltung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Grundlage des Zustandsspeichers
- zwei Inverter mit Rückkopplung:
  Q_{prev} = previous Q
- zwei Ausgänge: Q, $\overline{Q}$
- speichert 1 bit durch zwei stabile Zustände
  - $Q = 0 \Rightarrow \overline{Q} = 1 \Rightarrow Q = 0$
  - $Q = 1 \Rightarrow \overline{Q} = 0 \Rightarrow Q = 1$
- *keine* Eingänge

  $\Rightarrow$ gespeicherter Zustand kann nicht beeinflusst werden

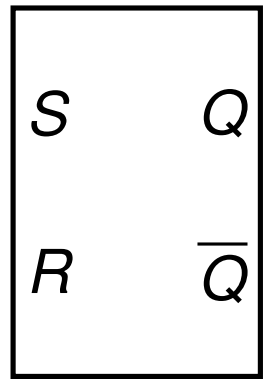




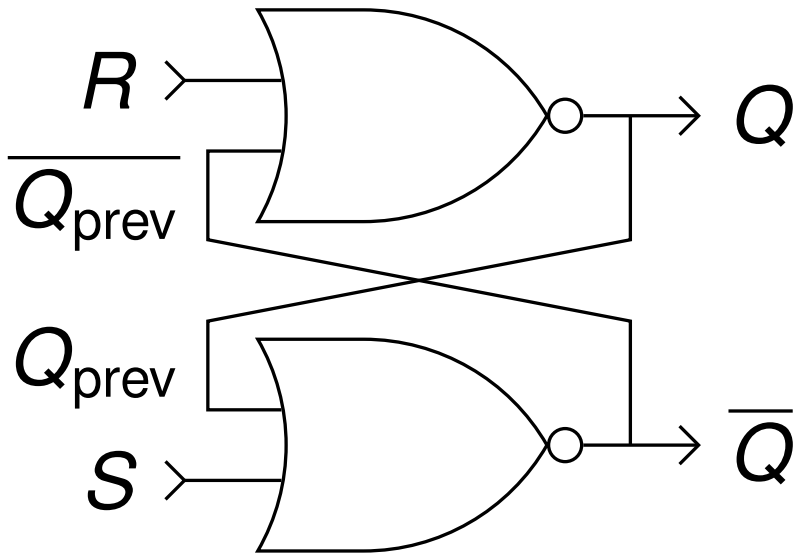
SR-Latch

TECHNISCHE UNIVERSITÄT DARMSTADT

- bistabile Grundschaltung mit NOR statt NOT
- Interpretation der freien Eingänge S und R
  - $\overline{S}\,\overline{R} \rightarrow$ Zustand halten („latch“ = verriegeln)
  - $\overline{S}\,R \rightarrow$ Zustand auf 0 rücksetzen („reset“)
  - $S\,\overline{R} \rightarrow$ Zustand auf 1 setzen („set“)
  - $S\,R \rightarrow$ ungültiger Zustand ($Q = \overline{Q} = 0$)



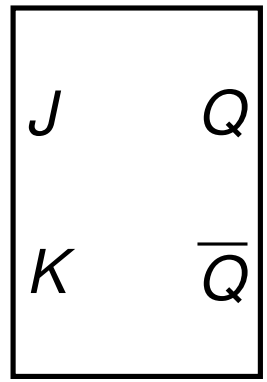
S	R	Q_{prev}	$\overline{Q_{prev}}$	Q	$\overline{Q}$
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0



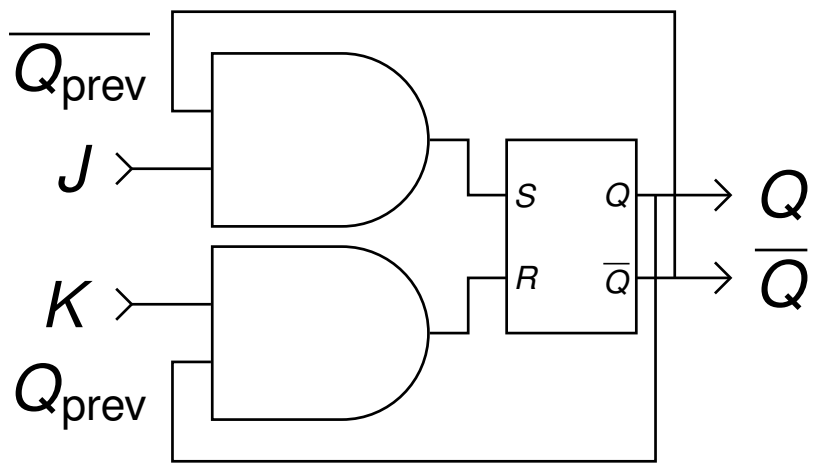
JK-Latch

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ungültigen Zustand am SR-Latch verhindern
- Interpretation der freien Eingänge J und K
  - $\overline{J}\,\overline{K} \rightarrow$ Zustand halten
  - $\overline{J}\,K \rightarrow$ Zustand auf 0 rücksetzen, falls nötig
  - $J\,\overline{K} \rightarrow$ Zustand auf 1 setzen, falls nötig
  - $J\,K \rightarrow$ Zustand invertieren („toggle“)



J	K	Q_{prev}	$\overline{Q_{prev}}$	S	R	Q	$\overline{Q}$
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	1	0	0	0	1
0	1	1	0	0	1	0	1
1	0	0	1	1	0	1	0
1	0	1	0	0	0	1	0
1	1	0	1	1	0	1	0
1	1	1	0	0	1	0	1



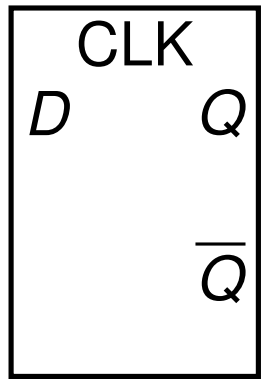
D-Latch

TECHNISCHE UNIVERSITÄT DARMSTADT

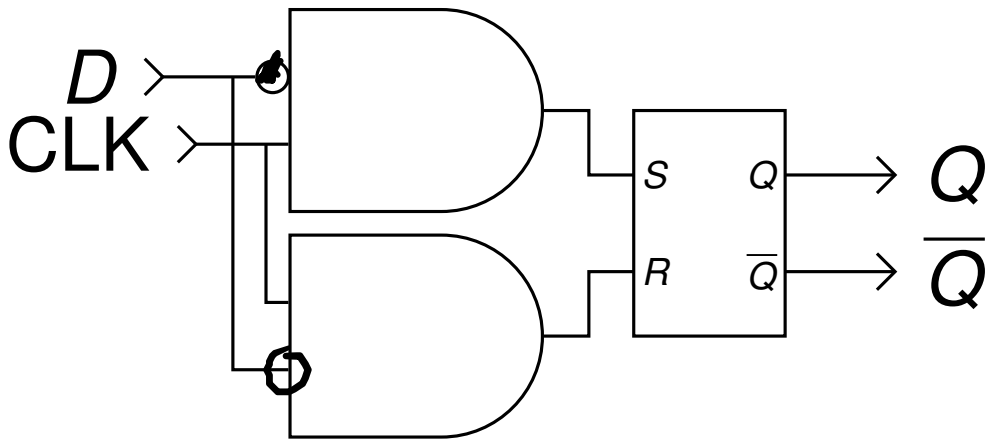
- Daten-Latch mit Taktsignal (CLK) und Dateneingang (D)
  - CLK = 1 $\rightarrow$ Zustand auf D setzen (Latch transparent)
  - CLK = 0 $\rightarrow$ Zustand halten (Latch nicht transparent)

$\Rightarrow$ ungültiger Zustand am SR-Latch wird vermieden

- Rückkopplung nur noch im SR-Latch



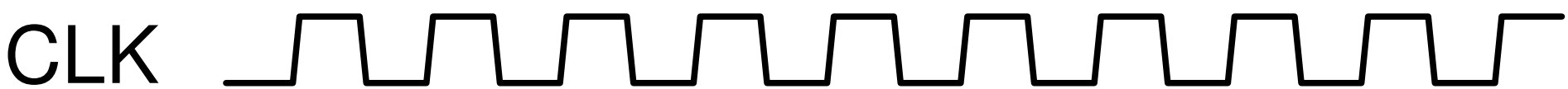
CLK	D	S	R	Q
0	0	0	0	Q_{prev}
0	1	0	0	Q_{prev}
1	0	0	1	0
1	1	1	0	1



Problem des D-Latch

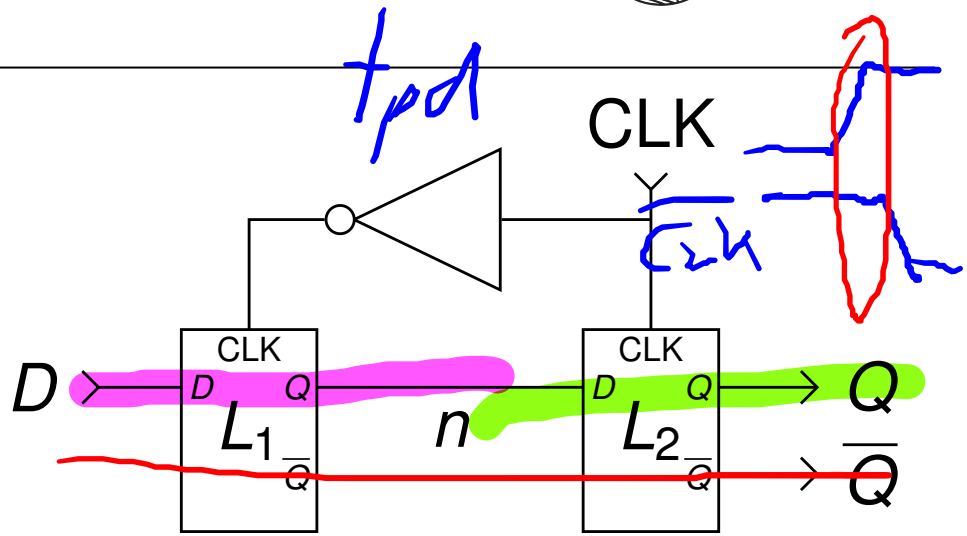
TECHNISCHE UNIVERSITÄT DARMSTADT

- periodische Taktsignale üblicherweise symmetrisch
  - 0-Phase und 1-Phase gleich lang
- D-Latch ist Taktphasen-gesteuert
  - für Hälfte der gesamten Zeit transparent
  - sequentielle Schaltungen mit D-Latches für Hälfte der Zeit kombinatorisch
- breites „Abtastfenster“ sorgt für Unschärfe
  - bspw. unklar, ob Störmimpulse übernommen

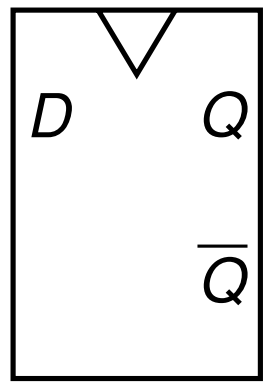


D-Flip-Flop

TECHNISCHE UNIVERSITÄT DARMSTADT



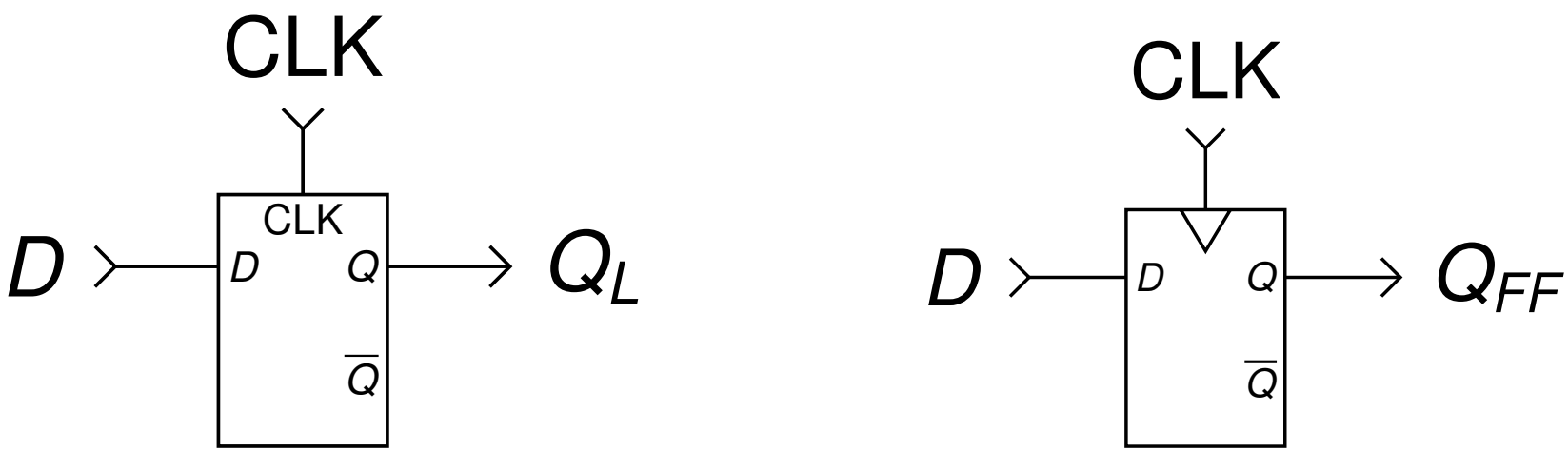
- Zwei D-Latches in Serie
  - L_1 = Master
  - L_2 = Slave
  - komplementäre Taktsignale
- CLK= 0
  - Master transparent $\rightarrow$ $n = D$
  - Slave nicht transparent $\rightarrow$ Q bleibt unverändert
- CLK= 1
  - Master nicht transparent $\rightarrow$ n bleibt unverändert
  - Slave transparent $\rightarrow$ $Q = n$

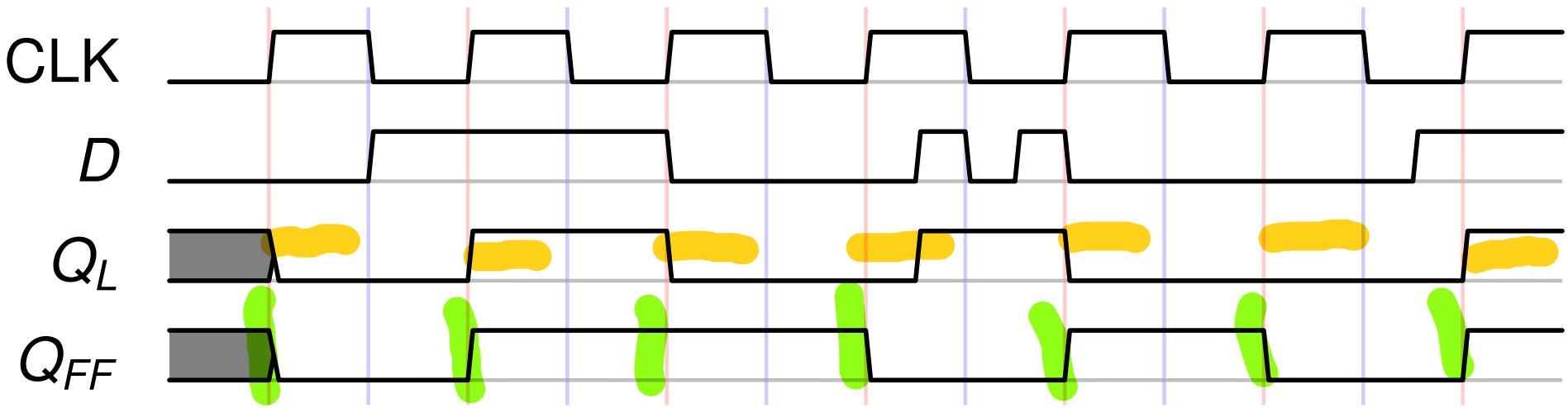


$\Rightarrow$ Taktflanken-gesteuert
- genau bei steigender CLK Flanke wird $Q = D$
- es wird der Wert von D übernommen, der *unmittelbar vor* der Taktflanke anliegt

Vergleich D-Latch mit D-Flip-Flop

TECHNISCHE UNIVERSITÄT DARMSTADT

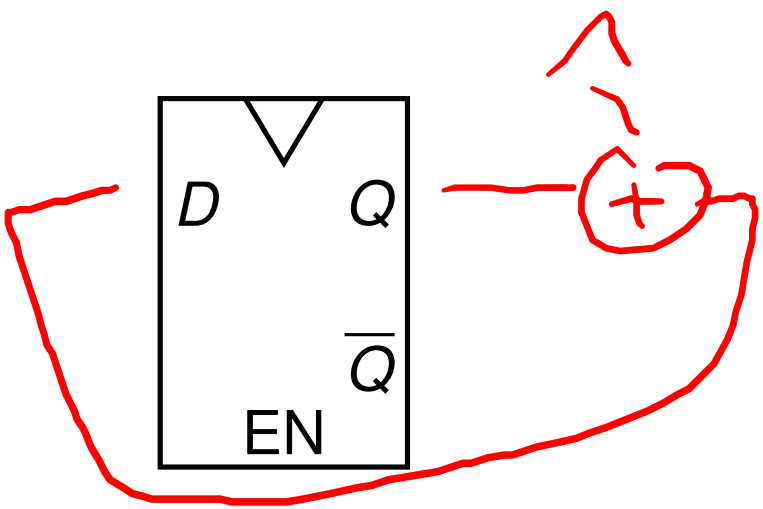


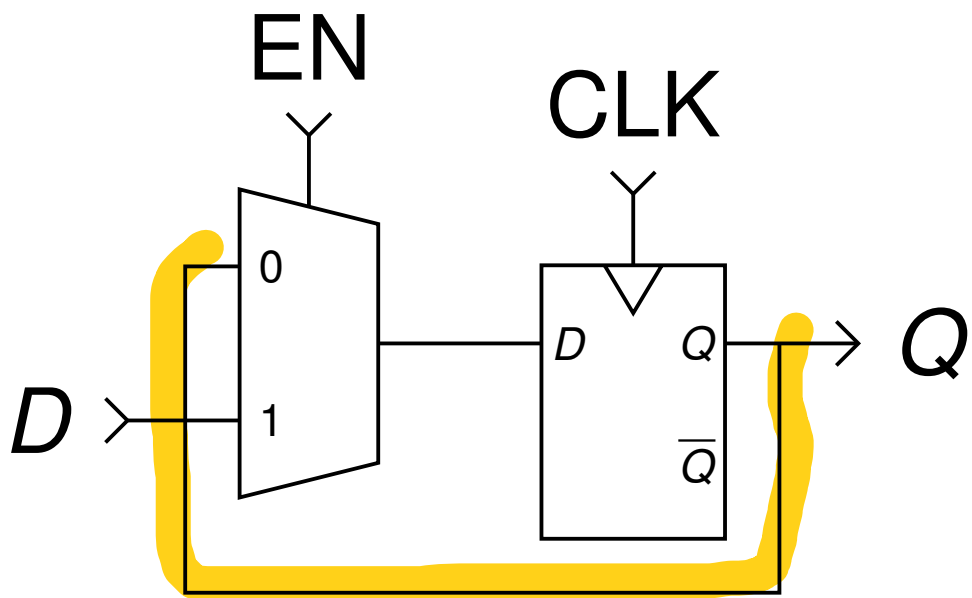


Flip-Flops mit Taktfreigabe

TECHNISCHE UNIVERSITÄT DARMSTADT

- Freigabeeingang (enable) steuert, wann Daten gespeichert werden
  - $EN = 1 \rightarrow D$ wird bei steigender CLK-Flanke gespeichert
  - $EN = 0 \rightarrow Q$ bleibt auch bei steigender CLK-Flanke unverändert
- Anwendungsbeispiele
  - Zähler
  - Speicher mit Adressdecoder

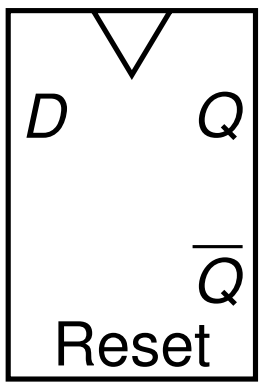


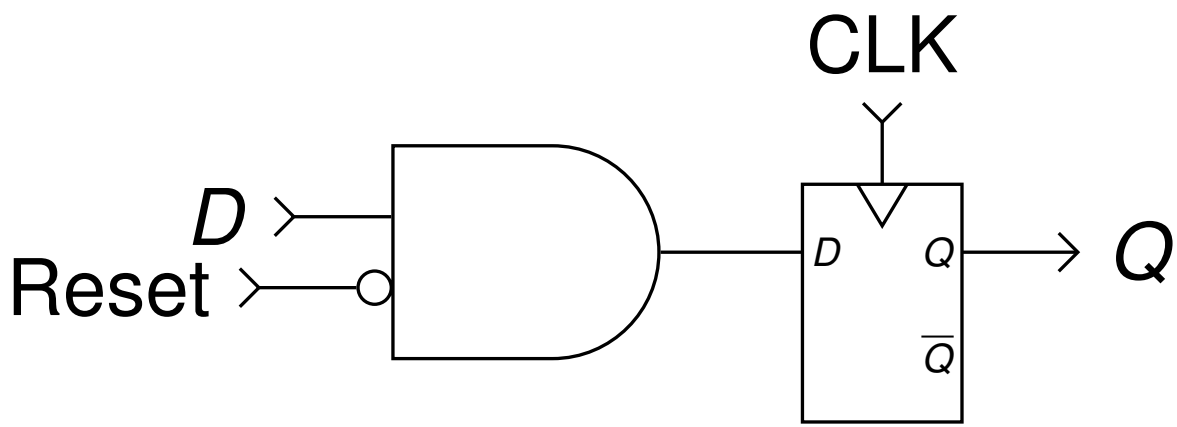


Zurücksetzbare Flip-Flops

TECHNISCHE UNIVERSITÄT DARMSTADT

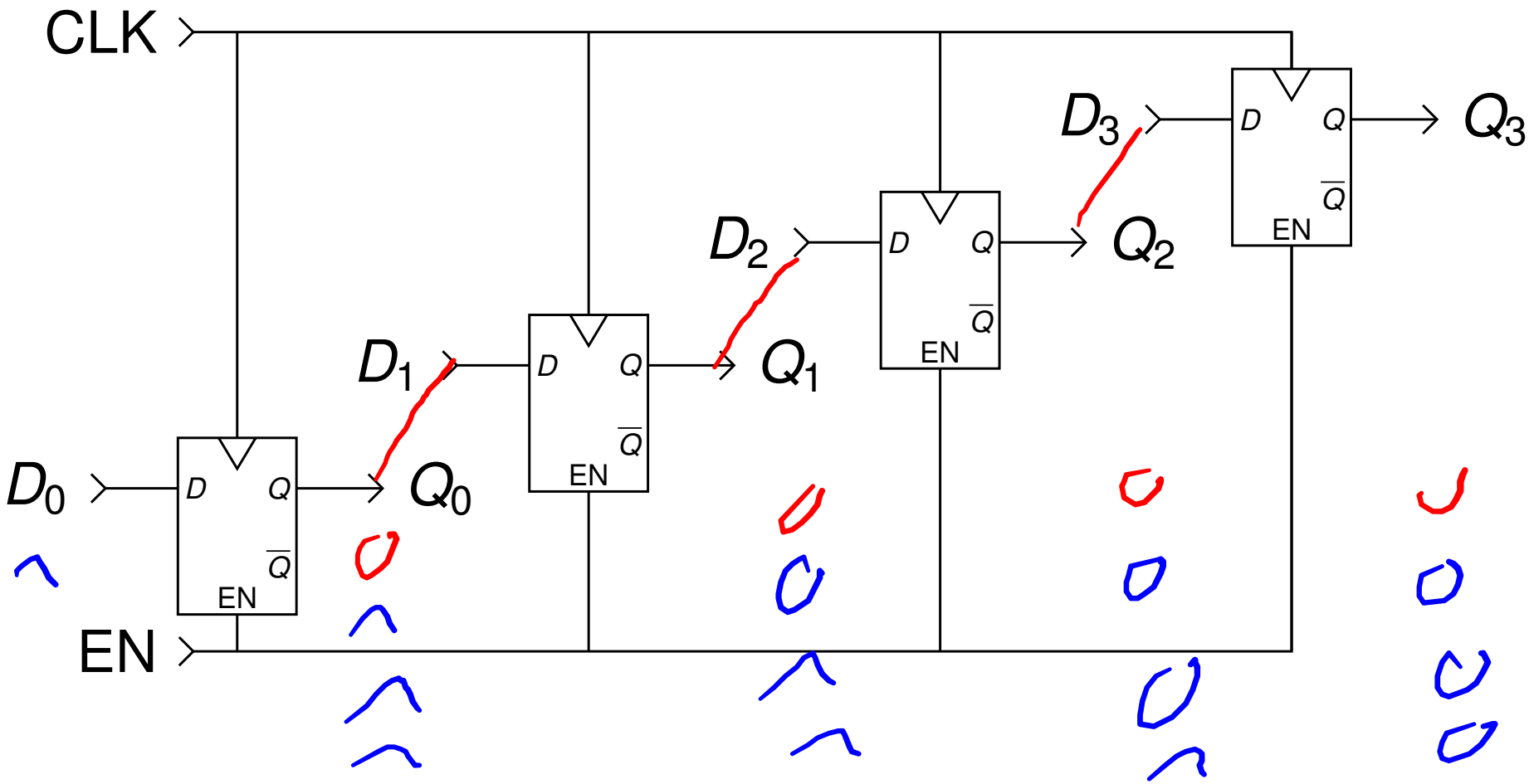
- Reset setzt internen Zustand unabhängig von *D* auf 0
  - synchron: nur zur steigenden Taktflanke wirksam
  - asynchron: jederzeit (unabhänig von CLK)
- Anwendungsbeispiele
  - sequentielle Schaltung in definierten Ausgangszustand versetzen
- setzbare Flip-Flops analog





Anwendungsbeispiel: (Shift-)Register

TECHNISCHE UNIVERSITÄT DARMSTADT



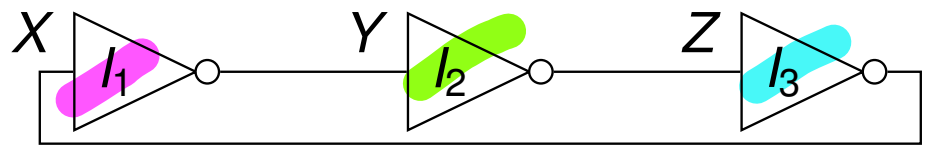
Synchrone sequentielle Logik

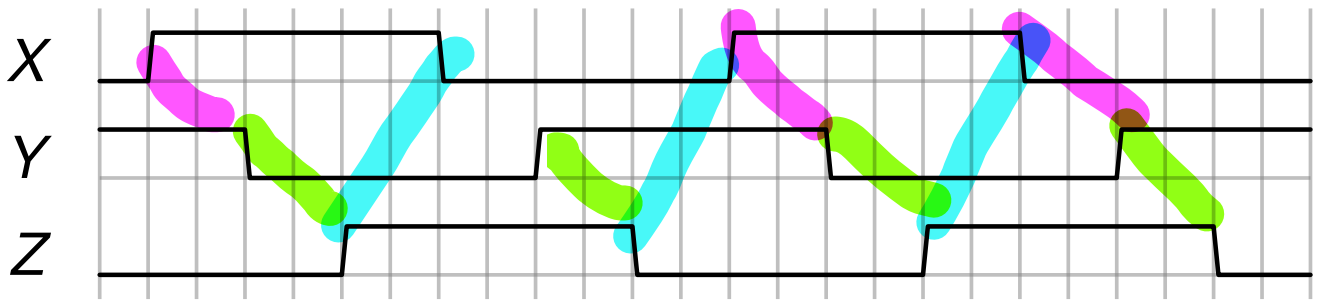
TECHNISCHE UNIVERSITÄT DARMSTADT

Sequentielle Logik

TECHNISCHE UNIVERSITÄT DARMSTADT

- alle nicht-kombinatorischen Schaltungen
- erlaubt Rückkopplungen, bspw:





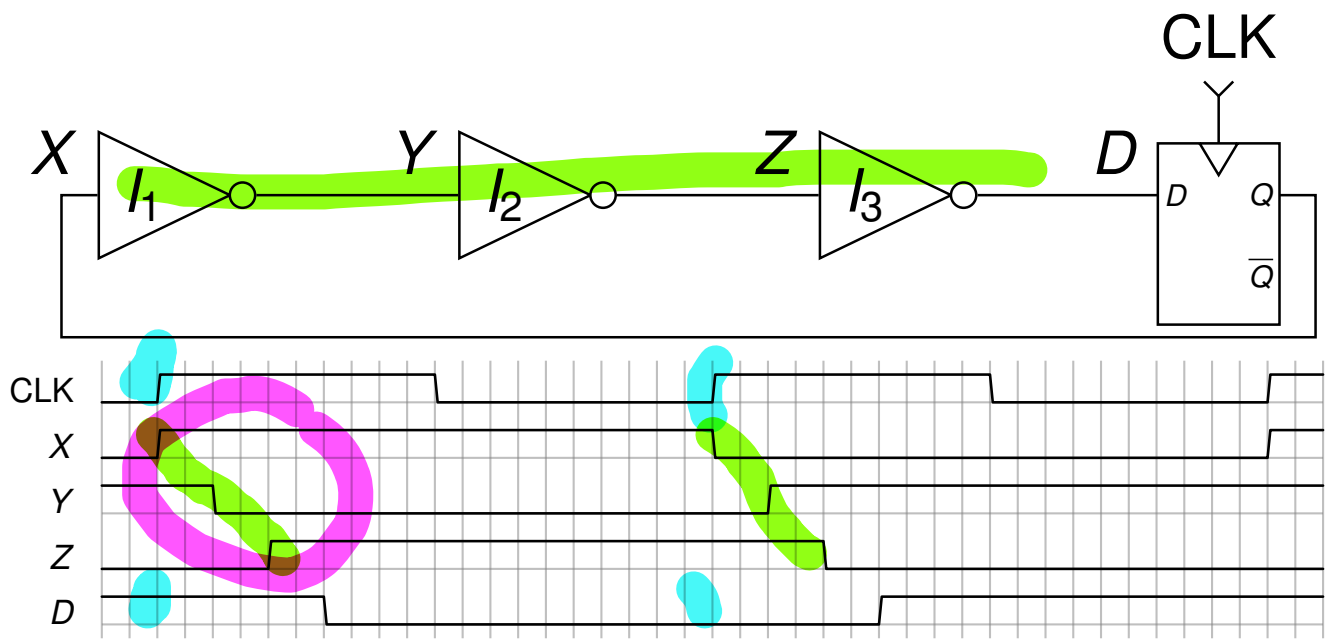
$\Rightarrow$ instabile (oszillierende Schaltung)
- Verhalten abhängig von Herstellungsprozess, Spannung, Temperatur
- nich vorhersagbar

Entwurf synchroner sequentieller Logik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Rückkopplungen durch Registern aufbrechen
  - halten den Zustand der Schaltung
  - ändern Zustand nur zur Taktflanke

  $\Rightarrow$ gesamte Schaltung *synchronisiert* mit Taktflanke



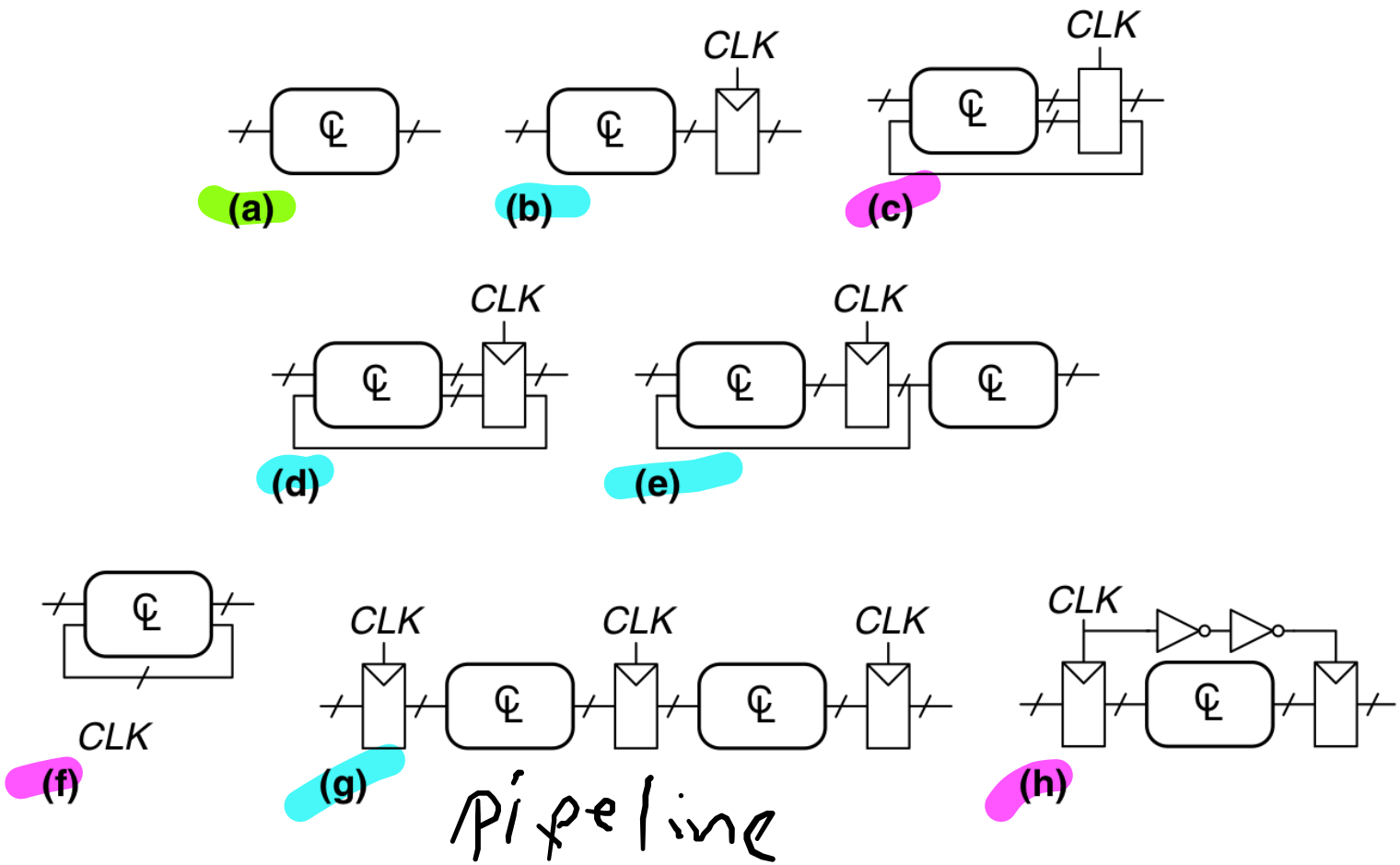
Synchrone sequentielle Schaltungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Regeln für Aufbau
  - jedes Schaltungselement ist entweder Register oder kombinatorische Schaltung
  - mindestens ein Schaltungselement ist Register
  - alle Register werden durch gleiches Taktsignal gesteuert
  - jeder Taktzyklus enthält mindestens ein Register
- Anwendungsbeispiele
  - Endliche Zustandsautomaten
  - Pipelines

Kombinatorisch? Sequentiell? Synchron?

TECHNISCHE UNIVERSITÄT DARMSTADT



Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

Zusammenfassung und Ausblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sequentielle Logik
  - Sequentielle Schaltungen
  - Speicherelemente
  - Synchrone Schaltungen

- Nächste Vorlesung behandelt
  - Endliche Zustandsautomaten