

Digitaltechnik

Wintersemester 2021/2022

7. Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT





Umfrage zur letzten Woche

Respektvoller Umgang – auch online



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- **Bisher keinerlei solcher Vorkommnisse in DT, aber immer wieder in anderen Lehrveranstaltungen**
- **Die TU Darmstadt versteht sich als Ort der Vielfalt**
- **Diskriminierungen werden nicht geduldet**
- **Bitte angemessen kommunizieren – auch im Chat**
- **Bei Vorfällen Lehrende ansprechen und Mail an achtung@tu-darmstadt.de**





Datum	Anmeldungen / Anwesende
Fr, 22.10.	70 / 100
Fr, 29.10.	80 / 92
Fr, 05.11.	55 / 73
Fr, 12.11.	40 / 45
Fr, 19.11.	25 / 26
Fr, 26.11.	26 / 26

Wir freuen uns sehr, dass die DT Präsenzbearbeitung ihren Hauptzweck erfüllt hat: dass Sie sich am Anfang des Semesters treffen und Lerngruppen bilden konnten. Aufgrund der aktuellen Pandemielage und Vorgaben werden wir zum Schutz aller Beteiligten **die DT Präsenzbearbeitung bis auf Weiteres einstellen.**

Nutzen Sie gerne Ihre individuellen Lerngruppen.










Bitte passen Sie gut auf sich selbst und auf Andere auf!

1. Arithmetische Grundsaltungen

2. Sequentielle Schaltungen

3. Speicherelemente

4. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

Überblick der heutigen Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Arithmetische Grundsaltungen
- ▶ Sequentielle Logik
 - ▶ Sequentielle Schaltungen
 - ▶ Speicherelemente



Harris 2013/2016
Kap. 5.2, 3.1 - 3.2

Agenda












TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Arithmetische Grundsaltungen

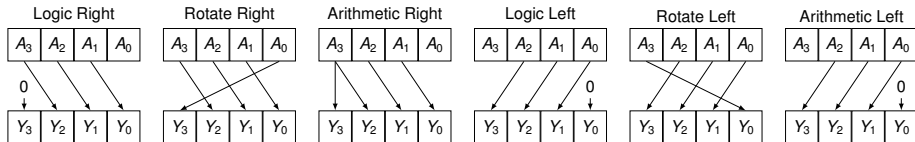
2. Sequentielle Schaltungen

3. Speicherelemente

4. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

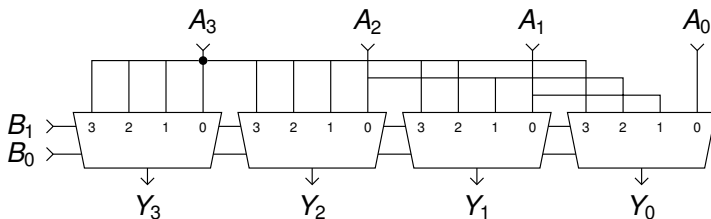
- ▶ A um B Stellen nach links/rechts verschieben
- ▶ Strategien zum Auffüllen der freien Stellen (Beispiele unten sind für $B = 1$):
 - ▶ *logischer* Rechts- oder Linksshift: Auffüllen mit Nullen
 - ▶ *umlaufender* Rechts- oder Linksshift: Auffüllen mit den aus der anderen Seite herausfallenden Bits (Rotation)
 - ▶ *arithmetischer* Rechtsshift: Auffüllen mit Vorzeichen des als Zweierkomplement interpretierten Dateneingangs (entspricht Division durch 2^B)
 - ▶ *arithmetischer* Linksshift: Auffüllen mit Nullen (entspricht Multiplikation mit 2^B)



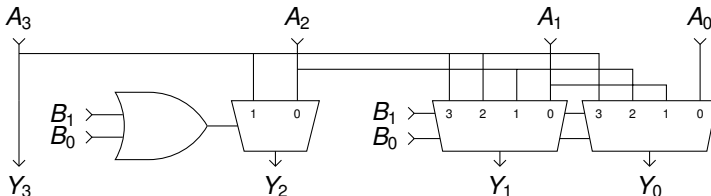
Barrel-Shifter f. Arithmetic Right



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Minimierte Schaltung



Arithmetische Shifter als Multiplizierer und Dividierer



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Arithmetischer Linksshift um n Stellen multipliziert den Zahlenwert mit 2^n

- ▶ $00001_2 \lll 3 = 01000_2 = 1 \cdot 2^3 = 8$
- ▶ $11101_2 \lll 2 = 10100_2 = -3 \cdot 2^2 = -12$

⇒ Multiplikation mit Konstanten kann aus Arithmetischen Linksshifts und Additionen zusammengesetzt werden

- ▶ $a \cdot 6 = a \cdot 110_2 = (a \lll 2) + (a \lll 1)$

- ▶ Arithmetischer Rechtsshift um n Stellen dividiert den Zahlenwert durch 2^n

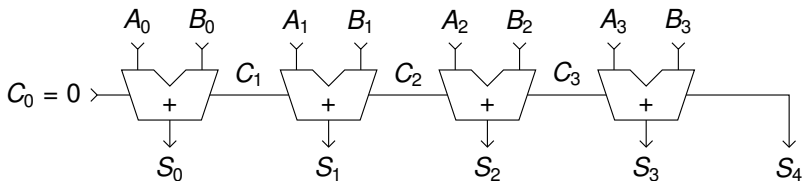
- ▶ $010000_2 \ggg 4 = 000001_2 = 16/2^4 = 1$
- ▶ $100000_2 \ggg 2 = 111000_2 = -32/2^2 = -8$

Ripple-Carry-Adder (RCA) LQ5-3 RQ5-3



TECHNISCHE
UNIVERSITÄT
DARMSTADT

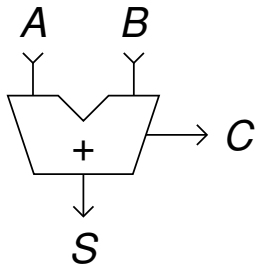
	1	0	1	1	0	Übertrag	C_4	C_3	C_2	C_1	C_0
		1	0	1	1	Summand		A_3	A_2	A_1	A_0
+		1	0	1	1	Summand		B_3	B_2	B_1	B_0
=	1	0	1	1	0	Summe	S_4	S_3	S_2	S_1	S_0



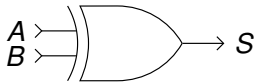
Halbaddierer



TECHNISCHE
UNIVERSITÄT
DARMSTADT



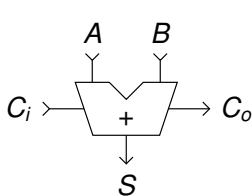
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



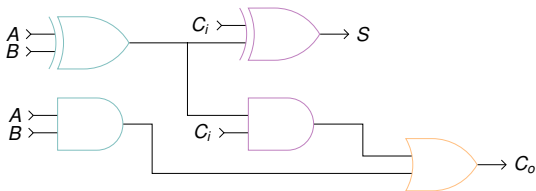
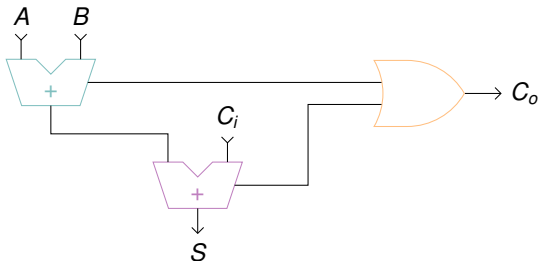
Volladdierer LQ5-2 RQ5-2



TECHNISCHE
UNIVERSITÄT
DARMSTADT



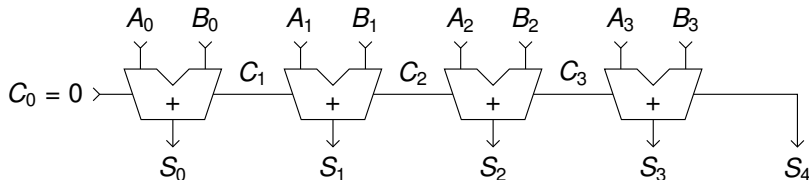
A	B	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Ripple-Carry-Adder (RCA)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

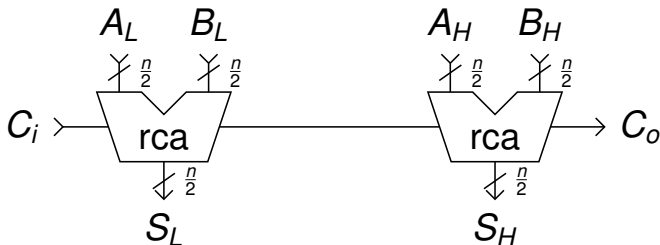


- ▶ Überträge werden über Kette von 1 bit Volladdierern vom LSB zum MSB weitergegeben
- ⇒ langer kritischer Pfad (steigt linear mit Bitbreite)
- ⇒ schnellere Addierer müssen Übertragskette aufbrechen (benötigen dafür mehr Hardware)
 - ▶ Conditional Sum Adder (CSA)
 - ▶ Carry-Lookahead Adder (CLA)
 - ▶ Prefix-Adder

Rekursiver Aufbau des Ripple-Carry-Adders

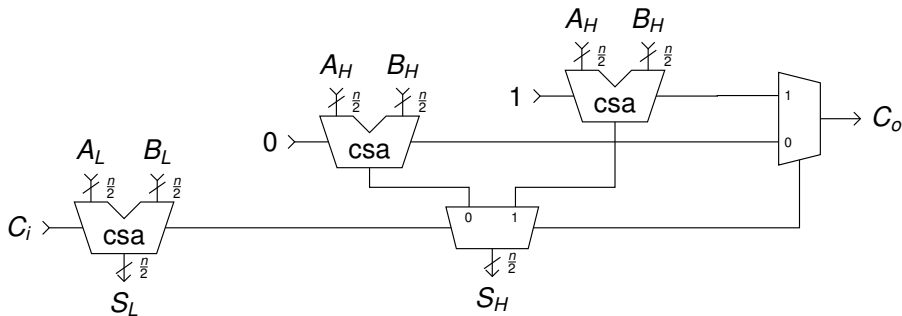


TECHNISCHE
UNIVERSITÄT
DARMSTADT



- ▶ Aufteilen in unteres (Low) und oberes (High) Halbwort (“Divide and Conquer”)
- ▶ zweiter Addierer muss auf Übertrag aus erstem Addierer “warten”
- ⇒ kritische Pfade beider Teiladdierer werden addiert
- ▶ für schnellen Addierer müssen *oberes und unteres Halbwort gleichzeitig* bearbeitet werden

Conditional Sum Adder (CSA)



- ▶ Übertrag vom unterem (L) in oberes (H) Halbwort kann nur 0 oder 1 sein
 - ▶ für beide Optionen kann oberes Halbwort schon mal vorberechnet werden
 - ▶ Auswahl des richtigen Ergebnisses, sobald tatsächlicher Übertrag bekannt
- ⇒ nach halbem CSA folgt nur noch ein MUX auf kritischem Pfad

Carry Lookahead Adder (CLA) LQ5-4 RQ5-4

Motivation



TECHNISCHE
UNIVERSITÄT
DARMSTADT

	1	0	1	1	0	Übertrag
		1	0	0	1	Summand A
+		1	0	1	1	Summand B
<hr/>						
=	1	0	1	0	0	Summe

- ▶ für $A_i B_i = 1$ ist $C_i = 1$ unabhängig von C_{i-1}
⇒ Spalte i *generiert* einen Übertrag (“generate”)
- ▶ für $A_i + B_i = 1$ ist $C_i = 1$ wenn $C_{i-1} = 1$
⇒ Spalte i *leitet* Übertrag *weiter* (“propagate”)
- ▶ für $A_i + B_i = 0$ ist $C_i = 0$ unabhängig von C_{i-1}
⇒ Spalte i *leitet* Übertrag *nicht weiter*

Carry Lookahead Adder (CLA)

Generate und Propagate pro Spalte



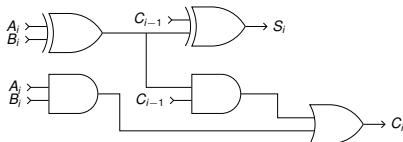
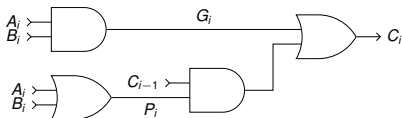
TECHNISCHE
UNIVERSITÄT
DARMSTADT

► Generate-Flag für Spalte i : $G_i = A_i B_i$

► Propagate-Flag für Spalte i : $P_i = A_i + B_i$

⇒ Übertrag aus Spalte i : $C_i = G_i + P_i C_{i-1}$

► Bei naiver Verwendung davon (links) kein Vorteil ggü. Volladdierer (rechts):
In beiden Fällen AND und OR auf kritischem Pfad zwischen C_{i-1} und C_i



Carry Lookahead Adder (CLA)

Generate und Propagate über k Spalten



- ▶ Generate- und Propagate-Flags können über mehrere Spalten kombiniert werden (hier gezeigt für $k = 4$ Spalten)
- ▶ k -Spalten Block *propagiert* Übertrag, wenn jede einzelne Spalte propagiert
 $\Rightarrow P_{3:0} = P_3 P_2 P_1 P_0$
- ▶ k -Spalten Block *generiert* Übertrag, wenn eine der Spalten generiert, und alle anderen Spalten darüber propagieren
 $\Rightarrow G_{3:0} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$
- ▶ Übertrag überspringt k Spalten auf einmal:

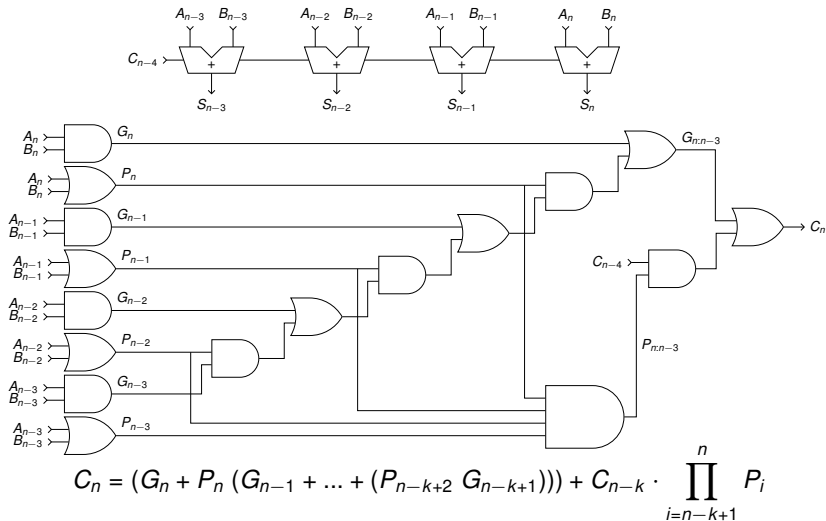
$$\begin{aligned} C_n &= G_{n:n-k+1} + C_{n-k} \cdot P_{n:n-k+1} \\ &= (G_n + P_n (G_{n-1} + \dots + (P_{n-k+2} G_{n-k+1}))) + C_{n-k} \cdot \prod_{i=n-k+1}^n P_i \end{aligned}$$

Carry Lookahead Adder (CLA)

Block für $k = 4$ Spalten



TECHNISCHE
UNIVERSITÄT
DARMSTADT



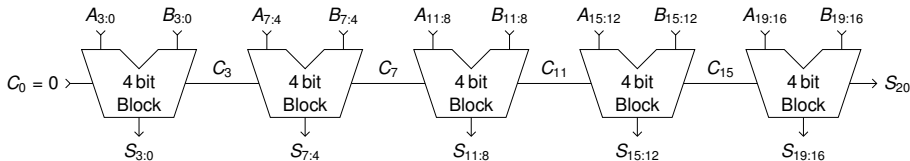
Carry Lookahead Adder (CLA)

kritischer Pfad



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Propagate und Generate Signale können in allen Blöcken gleichzeitig berechnet werden
 - ▶ für große Bitbreiten N dominiert $\frac{N}{k} \cdot (t_{pd,AND} + t_{pd,OR})$. Unten: $N = 20, k = 4$
- ⇒ Blöcke möglichst groß wählen (kostet aber mehr Ressourcen)
- ▶ Bereits ab $N = 8$ bit ist CLA mit $k = 4$ schneller als RCA:
 - ▶ RCA: $8 \cdot (t_{pd,AND} + t_{pd,OR})$
 - ▶ CLA: $(4 \cdot t_{pd,AND} + 4 \cdot t_{pd,OR}) + (t_{pd,AND} + t_{pd,OR}) = 5 \cdot (t_{pd,AND} + t_{pd,OR})$



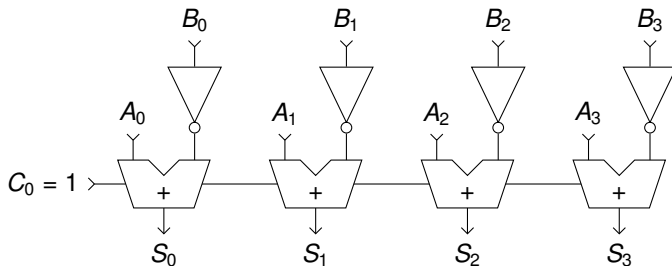


- ▶ Parallel Prefix Adder
 - ▶ *alle* C_i per Generate und Propagate möglichst schnell bestimmen
 - ▶ Kritischer Pfad logarithmisch in Bitbreite N
- ▶ Carry-Save Adder
 - ▶ Verwendet parallele Volladdierer, um 3 Werte in Vektor aus Carries C_i und Summen S_i zu addieren



Harris 2013/2016
Kap. 5.2.1

- ▶ kann mit Addition und Negation realisiert werden: $A - B = A + (-B)$
 - ▶ Negation im Zweierkomplement: Komplement und Inkrement
- ⇒ Addierer mit NOT-Gatter an B -Eingängen und $C_0 = 1$

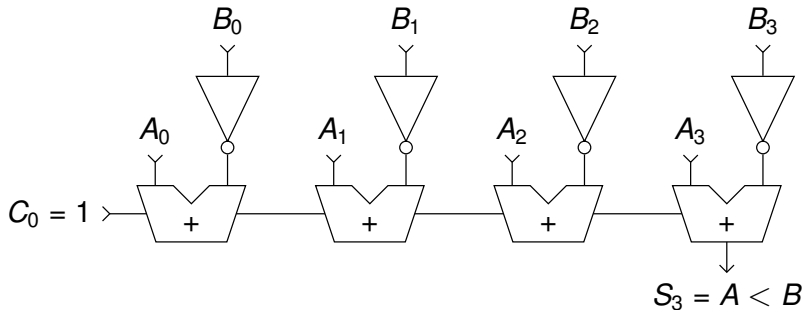


Vergleich: Kleiner als



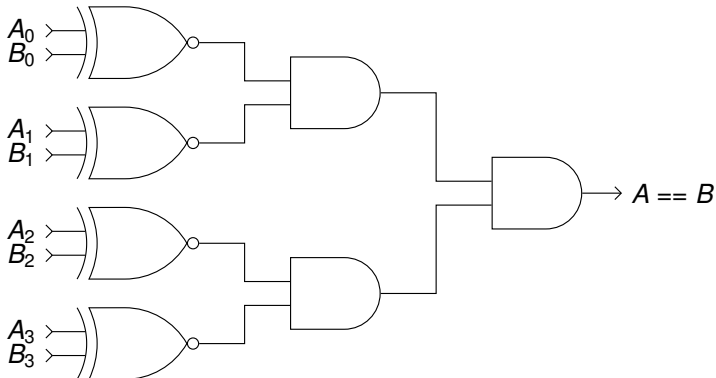
TECHNISCHE
UNIVERSITÄT
DARMSTADT

- kann mit Subtraktion realisiert werden: $A < B \Leftrightarrow A - B < 0$





► Bitweise XNOR und AND-Baum





- ▶ Produkt von n und m bit breiten Faktoren ist $n + m$ bit breit
- ▶ Teilprodukte aus einzelnen Ziffern des Multiplikators mit dem Multiplikanden
- ▶ verschobene Teilprodukte danach addieren

Decimal

$$\begin{array}{r} 230 \\ \times 42 \\ \hline 460 \\ + 920 \\ \hline 9660 \end{array}$$

Multiplikand
Multiplikator

Teilprodukte

Ergebnis

$$230 \times 42 = 9660$$

Binary

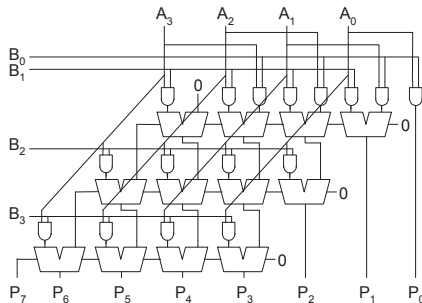
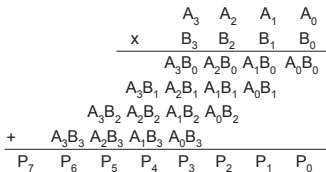
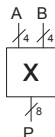
$$\begin{array}{r} 0101 \\ \times 0111 \\ \hline 0101 \\ 0101 \\ 0101 \\ + 0000 \\ \hline 0100011 \end{array}$$

$$5 \times 7 = 35$$

Kombinatorische 4×4 Multiplikation



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Weitere wichtige arithmetische Algorithmen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Division (Kap. 5.2.7)
- ▶ Festkomma/Gleitkomma Arithmetik (Kap. 5.3)



Harris 2013/2016



Pause & Umfrage bis hier

Agenda



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Arithmetische Grundsaltungen

2. Sequentielle Schaltungen

3. Speicherelemente

4. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



- ▶ kombinatorische Logik (“Schaltnetz”)
 - ▶ Ausgänge hängen nur von *aktuellen* Eingangswerten ab
- ▶ Warum reichen kombinatorische Schaltungen nicht aus?
 - ▶ Nicht alle Funktionalitäten lassen sich als kombinatorische Schaltungen realisieren
 - ▶ (Zwischen-)Ergebnisse können nicht gespeichert/wiederverwendet werden
 - ▶ kritische Pfade können nicht beliebig lang werden
 - ▶ Zeitverhalten bei kombinatorischen Schaltungen schwer kontrollierbar (siehe Timing-Analyse in vorheriger Vorlesung)



- ▶ Ausgänge hängen ab von
 - ▶ aktuellen Eingabewerten und
 - ▶ *vorherigen* Eingabewerten
- ⇒ sequentielle Schaltung speichern *internen Zustand*
 - ▶ (Kurzzeit-)Gedächtnis repräsentiert bisherige Eingabesequenzen
 - ▶ realisiert durch Rückkopplungen von Ausgängen
- ⇒ nicht kombinatorisch

Agenda












TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Arithmetische Grundsaltungen

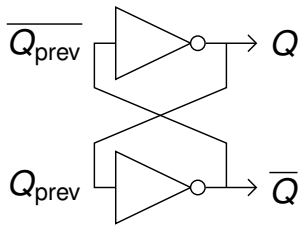
2. Sequentielle Schaltungen

3. Speicherelemente

4. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

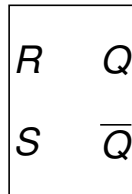
- ▶ Grundlage des Zustandsspeichers
- ▶ zwei Inverter mit Rückkopplung:
 Q_{prev} (previous Q)
- ▶ zwei Ausgänge: Q, \overline{Q}
- ▶ speichert 1 bit durch zwei stabile Zustände
 - ▶ $Q = 0 \Rightarrow \overline{Q} = 1 \Rightarrow Q = 0$
 - ▶ $Q = 1 \Rightarrow \overline{Q} = 0 \Rightarrow Q = 1$
- ▶ *keine* Eingänge
 - ⇒ gespeicherter Zustand kann nicht beeinflusst werden



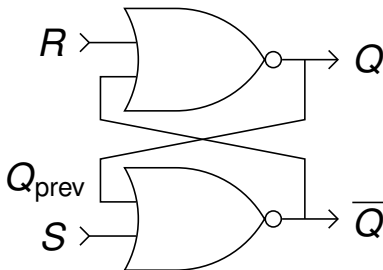
SR-Latch



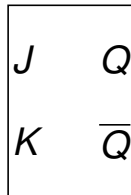
- ▶ bistabile Grundschialtung mit NOR statt NOT
- ▶ NOR: Ausgang 0 wenn einer der Inputs 1 ist
- ▶ Interpretation der freien Eingänge S und R
 - ▶ $\bar{S} \bar{R} \rightarrow$ Zustand halten ("latch" = verriegeln)
 - ▶ $\bar{S} R \rightarrow$ Zustand auf 0 rücksetzen ("reset" R)
 - ▶ $S \bar{R} \rightarrow$ Zustand auf 1 setzen ("set" S)
 - ▶ $S R \rightarrow$ ungültiger Zustand ($Q = \bar{Q} = 0$)



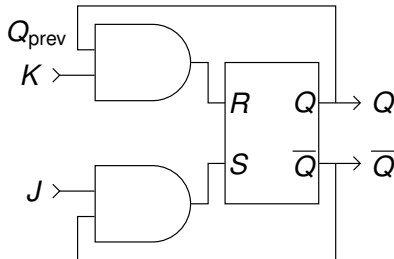
S	R	Q_{prev}	Q	\bar{Q}
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	0



- ▶ Ungültigen Zustand SR am SR-Latch verhindern
- ▶ Historisch unklar, woher die Bezeichnung "JK" kommt
- ▶ Interpretation der freien Eingänge J und K
 - ▶ $\bar{J} \bar{K} \rightarrow$ Zustand halten
 - ▶ $\bar{J} K \rightarrow$ Zustand auf 0 rücksetzen, falls nötig
 - ▶ $J \bar{K} \rightarrow$ Zustand auf 1 setzen, falls nötig
 - ▶ $J K \rightarrow$ Zustand invertieren ("toggle")



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

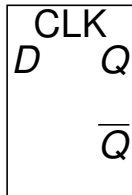


- ▶ Daten-Latch mit Taktsignal (CLK) und Dateneingang (D)

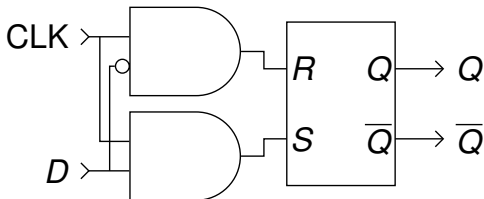
- ▶ CLK = 1 \rightarrow Zustand auf D setzen (Latch transparent)
- ▶ CLK = 0 \rightarrow Zustand halten (Latch nicht transparent)

\Rightarrow ungültiger Zustand am SR-Latch wird vermieden

- ▶ Rückkopplung nur noch im SR-Latch



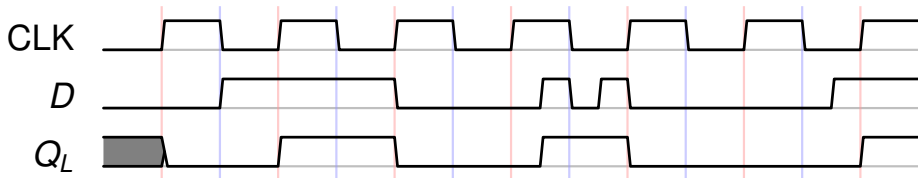
CLK	D	S	R	Q
0	0	0	0	Q_{prev}
0	1	0	0	Q_{prev}
1	0	0	1	0
1	1	1	0	1



Problem des D-Latch



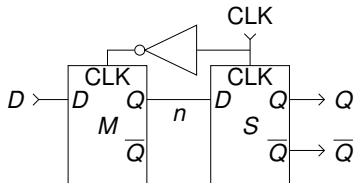
- ▶ periodische Taktsignale üblicherweise symmetrisch
 - ▶ 0-Phase und 1-Phase gleich lang
- ▶ D-Latch ist Taktphasen-gesteuert
 - ▶ für Hälfte der gesamten Zeit transparent
 - ▶ sequentielle Schaltungen mit D-Latches für Hälfte der Zeit kombinatorisch
- ▶ breites “Abtastfenster” sorgt für Unschärfe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





► Zwei D-Latches in Serie

- Master (M)
- Slave (S)
- komplementäre Taktsignale

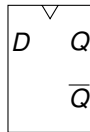


► CLK = 0

- Master transparent $\rightarrow n = D$
- Slave nicht transparent $\rightarrow Q$ bleibt unverändert

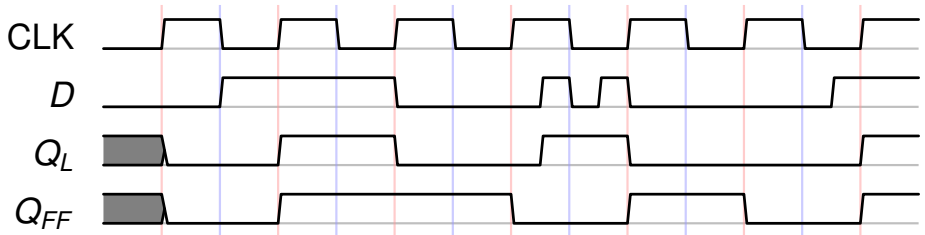
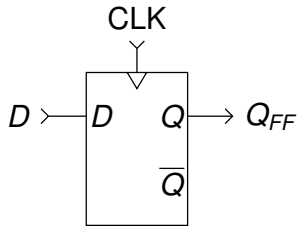
► CLK = 1

- Master nicht transparent $\rightarrow n$ bleibt unverändert
- Slave transparent $\rightarrow Q = n$



⇒ Taktflanken-gesteuert

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



Agenda



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. Arithmetische Grundsaltungen

2. Sequentielle Schaltungen

3. Speicherelemente

4. Zusammenfassung

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



- ▶ Arithmetische Grundsaltungen
- ▶ Sequentielle Logik
 - ▶ Sequentielle Schaltungen
 - ▶ Speicherelemente

- ▶ Nächste Vorlesung behandelt
 - ▶ Synchrone Schaltungen
 - ▶ Zeitverhalten Sequentieller Logik