

# Kapitel 4 – Sequentielle Logik

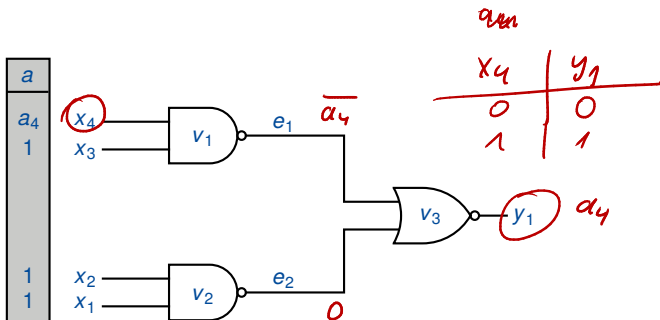
## 1. Speichernde Elemente

2. Sequentielle Schaltkreise
3. Entwurf sequentieller Schaltkreise
4. SRAM
5. Anwendung: Datenpfade von ReTI



# Schaltkreis (1/2)

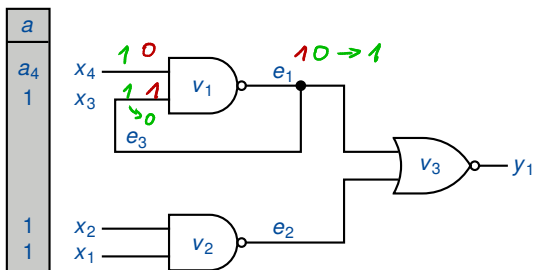
$typ(v_1) = typ(v_2) = NAND$   
 $typ(v_3) = NOR$



Schaltkreis mit Eingabebelegung  $a = (1, 1, 1, a_4)$ .



## Schaltkreis (2/2)



- Analyse von Schaltplänen  $SP = (\vec{X}_n, G, typ, IN, \vec{Y}_m)$  mit  $G$  nicht notwendigerweise azyklisch.

## Schaltpläne (2/3)

- Eine **Zellenbibliothek**  $BIB \subseteq \bigcup_{n \in \mathbb{N}} \mathbb{B}_n$  enthält Basisoperationen, die den Grundgattern entsprechen.
- Ein 5-Tupel  $SP = (\vec{X}_n, G, typ, IN, \vec{Y}_m)$  heißt **Schaltplan** mit  $n$  **Eingängen** und  $m$  **Ausgängen** über der Zellenbibliothek  $BIB$  genau dann, wenn
  - $\vec{X}_n = (x_1, \dots, x_n)$  ist eine endliche Folge von Eingängen.
  - $G = (V, E)$  ist ein gerichteter Graph mit  $\{0, 1\} \cup \{x_1, \dots, x_n\} \subseteq V$ .
  - Die Menge  $I = V \setminus (\{0, 1\} \cup \{x_1, \dots, x_n\})$  heißt **Menge der Gatter**.  
Die Abbildung  $typ : I \rightarrow BIB$  ordnet jedem Gatter  $v \in I$  einen **Zellentyp**  $typ(v) \in BIB$  zu.
- ...

- ...
- Für jedes Gatter  $v \in I$  mit  $\text{typ}(v) \in B_k$  gilt  $\text{indeg}(v) = k$ .
- $\text{indeg}(v) = 0$  für  $v \in \{0, 1\} \cup \{x_1, \dots, x_n\}$ .
- Die Abbildung  $\text{IN} : I \rightarrow E^*$  legt für jedes Gatter  $v \in I$  eine Reihenfolge der eingehenden Kanten fest, d.h. falls  $\text{indeg}(v) = k$ , dann ist  $\text{IN}(v) = (e_1, \dots, e_k)$  mit  $Z(e_i) = v \quad \forall 1 \leq i \leq k$ .
- Die Folge  $\vec{Y}_m = (y_1, \dots, y_m)$  zeichnet Knoten  $y_i \in V$  als Ausgänge aus.

# Belegungen von Schaltplänen (1/2)

---

Sei nun ein Schaltplan  $SP = (\vec{X}_n, G, typ, IN, \vec{Y}_n)$  gegeben.

- Eine Abbildung  $\phi_{SP,a} : V \rightarrow \{0, 1\}$  für  $a = (a_1, \dots, a_n) \in \mathbb{B}^n$  heißt **Belegung** für Eingabebelegung  $a$ , falls
  - $\phi_{SP,a}(\underline{x_i}) = \underline{a_i} \quad \forall 1 \leq i \leq n$ ,
  - $\phi_{SP,a}(0) = 0, \phi_{SP,a}(1) = 1$ .

## Belegung von Schaltplänen (2/2)

Interessant für uns sind **stabile** Belegungen.

- Eine Belegung  $\Phi_{SP,a} : V \rightarrow \{0, 1\}$  heißt stabil, wenn gilt:
  - Für alle  $v \in I$  mit  $typ(v) = g \in B_k$ ,  $IN(v) = (e_1, \dots, e_k)$  ist  $\Phi_{SP,a}(v) = g(\Phi_{SP,a}(Q(e_1)), \dots, \Phi_{SP,a}(Q(e_k)))$ .
  - $(\Phi_{SP,a}(y_1), \Phi_{SP,a}(y_2), \dots, \Phi_{SP,a}(y_n))$  heißt Ausgangsbelegung von  $SP$  bei Eingangsbelegung  $a$ .

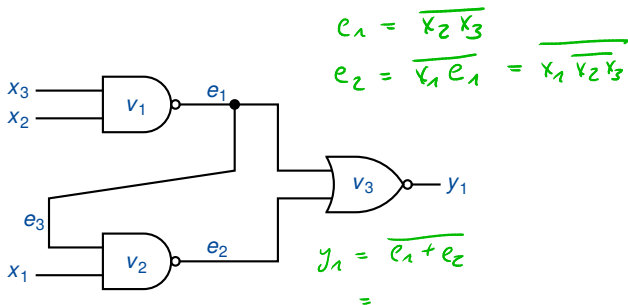
Es ist möglich, dass es zu einer Eingangsbelegung  $a$

- **keine** stabile Signalbelegung  $\Phi_{SP,a}$  gibt,
- **mehrere** stabile Signalbelegungen  $\Phi_{SP,a}$  gibt.





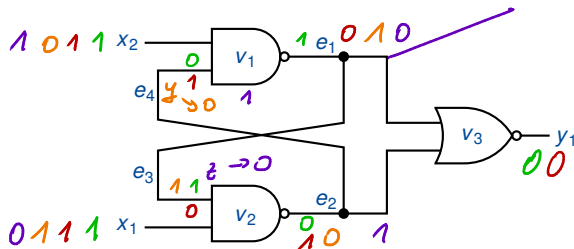
# Ein weiterer Schaltkreis (1/2)



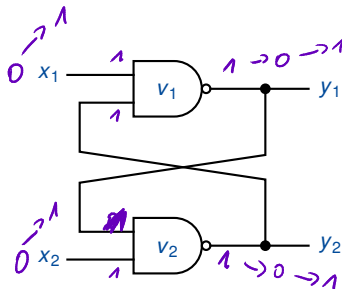
$$a = (a_1, a_2, a_3)$$



## Ein weiterer Schaltkreis (2/2)



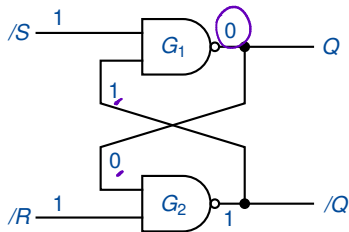
# Ausschnitt aus vorherigem Schaltplan



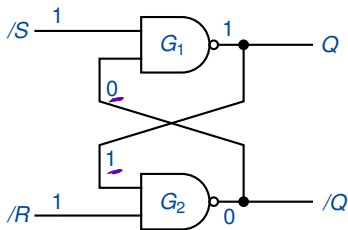
Hinweis: Gatter benötigen Zeit bis ihr “Ergebnis” am Ausgang bereit liegt. Was passiert wenn man die Eingabebelegung nach einer Zeit (mehrfach) ändert?

# RS-Flipflop (RS-FF)

- Die vorherige Schaltung heißt **RS-Flipflop**.
- Sie hat mindestens zwei stabile Zustände.



Zustand  $Q = 0$

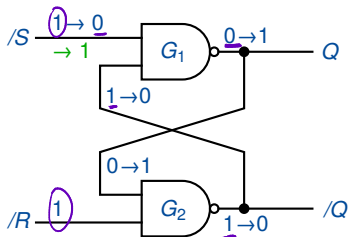


Zustand  $Q = 1$

- Für das Umschalten von einem Zustand zum anderen in einer realen Implementierung eines RS-FFs ist es von entscheidender Bedeutung, dass reale Gatter Verzögerungszeiten haben.
- D.h.: Wenn sich die Eingangsbelegung eines Gatters ändert, dann erfolgt die daraus resultierende Änderung des Ausgangswertes nicht direkt, sondern mit einer gewissen Verzögerung.
- (Detailliertere Betrachtung in Kapitel 5, Physikalische Eigenschaften von Gattern.)

## Übergang (2/2)

- Zustand  $Q = 0 \rightarrow$  Zustand  $Q = 1$ :



- Senke  $/S$  zur Zeit  $t_0$  ab und hebe zu  $t_0 + x$  wieder an (einen solchen Signalverlauf nennt man **Puls**).
- Nach Zeit  $t_{P/SQ}$  ist  $Q = 1$ .
- Nach Zeit  $t_{P/S/Q}$  ist  $/Q = 0$ .

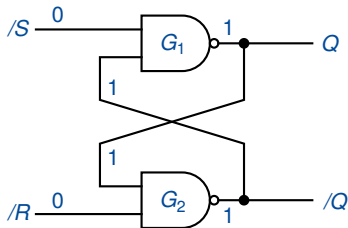
# Weitere Bezeichnungen

---

- Umschalten des FF in Zustand  $Q = 1$  heißt **Setzen (set)**.
- Umschalten des FF in Zustand  $Q = 0$  heißt **Zurücksetzen (reset)**.
- $/S$  heißt **Set-Signal**.
- $/R = /C$  heißt **Reset-** oder **Clear-Signal**.
- Weil  $/R, /S$  durch Absenken aktiviert werden, nennt man sie **active low**.
- Signalnamen von active-low-Signalen beginnen in der Regel mit /.

“Zustand”  $Q = 1, /Q = 1$

---



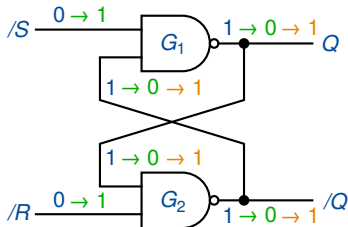
- Stabile Signalbelegung bei Eingangsbelegung  $/S = 0, /Q = 0$
- Aber warum ist es trotzdem problematisch,  $/S$  und  $/R$  gleichzeitig zu aktivieren ( $/S = 0, /R = 0$ )?



# Flackern

Annahme:

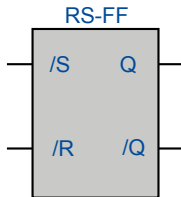
- $/S$  und  $/R$  werden nach ihrer Aktivierung beide gleichzeitig inaktiv ( $00 \rightarrow 11$ )
- $G_1$  und  $G_2$  schalten exakt gleich schnell, d.h. haben exakt die gleiche Verzögerungszeit



- $\Rightarrow$  Es kommt es zum Flackern ("metastabiler" Zustand).
- In der Praxis wird in der Regel nach einer gewissen Zeit einer der beiden stabilen Zustände angenommen (weil Gatterverzögerungen leicht variieren).

# Schaltsymbol eines RS-FF

---



# Nachteil von RS-FF

---

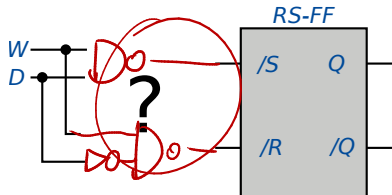
Beim Speichern eines Wertes 0 oder 1 muss man den Wert kennen:

- 0 → Aktiviere  $/R$
- 1 → Aktiviere  $/S$

## Ziel:

- Speichern unbekannter Werte.

# D-Latch (1/2)

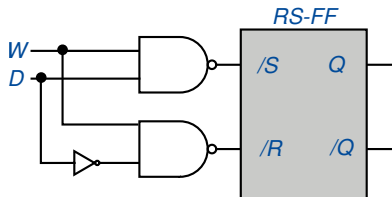


<i>W</i>	<i>D</i>	<i>/S</i>	<i>/R</i>
0	0		
0	1		
1	0		
1	1		

- *W* ist active high.
  - $W = 0 \Rightarrow /S, /R$  inaktiv
  - $W = 1 \Rightarrow \begin{cases} /S \text{ aktiv, falls } D = 1 \\ /R \text{ aktiv, falls } D = 0 \end{cases}$

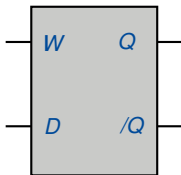


## D-Latch (2/2)



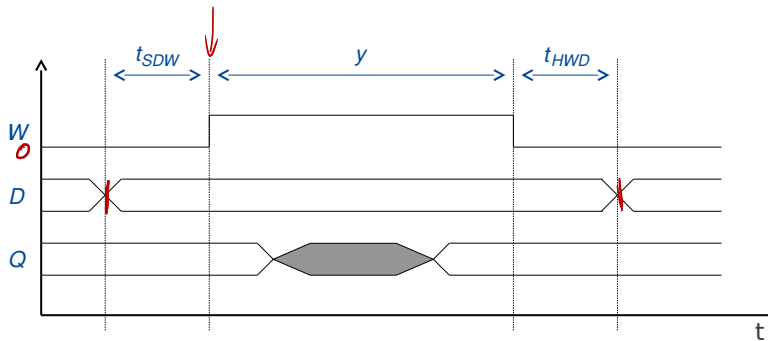
$W$	$D$	$/S$	$/R$
0	0	1	1
0	1	1	1
1	0	1	0
1	1	0	1

Symbol:



- Die Daten müssen für eine gewisse Zeit  $t_{SDW}$ , genannt **Setup-Zeit**, an  $D$  stabil anliegen.
- Dann geht  $W$  von 0 auf 1, bleibt für eine Zeit  $y$ , genannt **Pulsweite**, auf 1 und geht auf 0 zurück.
- Anschließend müssen die Daten für eine Zeit  $t_{HDW}$ , genannt **Hold-Zeit**, an  $D$  stabil gehalten werden.

# Timing-Diagramm



Wie lange müssen die einzelnen Signale aktiv sein, damit der Schreibvorgang reibungslos abläuft?

⇒ Siehe nächstes Kapitel ([Timing](#)).

# Weitere Eigenschaften eines D-Latches



---

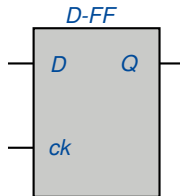
- Bisher: Keine Datenänderungen während des Schreibpulses auf  $W$ .
- Man kann das D-Latch aber auch im transparenten Modus betreiben:
  - Das D-Latch heißt transparent, wenn das Schreibsignal aktiv ist.
  - Hält man  $W$  lange aktiv und ändert  $D$  zur Zeit  $t$ , dann ändert sich  $Q$  zur Zeit  $t + t_{PDQ}$ .
  - Auch hier sind zeitliche Bedingungen zu beachten: Keine Datenänderungen kurz nach Beginn des Transparenzmodus bzw. kurz vor Ende des Transparenzmodus.



# Taktflankengesteuertes D-Flipflop (1/2)

- **Taktflankengesteuerte Flipflops** wie das D-Flipflop übernehmen Daten zu einem bestimmten **Zeitpunkt** (kein transparenter Modus!), nämlich bei der steigenden Flanke des Clocksignals.

<i>D</i>	<i>ck</i>	<i>Q</i>	<i>/Q</i>
0		0	1
1		1	0
X	0	Q	<i>/Q</i>
X	1	Q	<i>/Q</i>



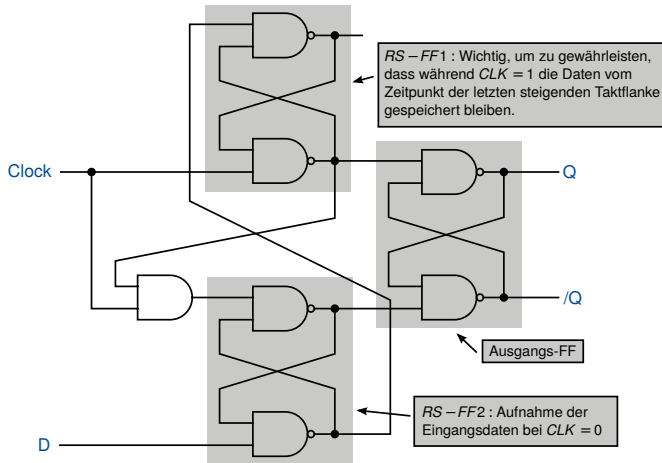
- **Vorteil:** Daten müssen lediglich bei der steigenden Taktflanke stabil sein (zzgl. Setup- und Holdzeit).

## Taktflankengesteuertes D-Flipflop (2/2)

---

- Realisierung: Wesentlich komplexer als bei taktzustandsgesteuerten D-Latches
- Analyse des Schaltplanes (und entsprechende Timing-Analyse) wesentlich komplizierter.

# D-FF: Realisierung mit RS-Flipflops



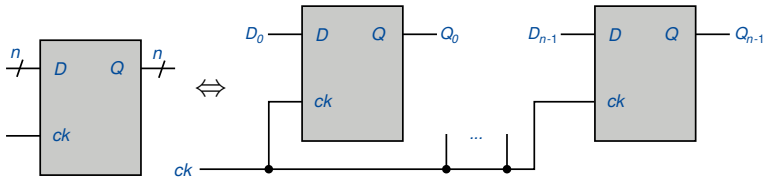
# Einfache Bausteine mit Flipflops

---

- Register
- Schieberegister
- Zähler

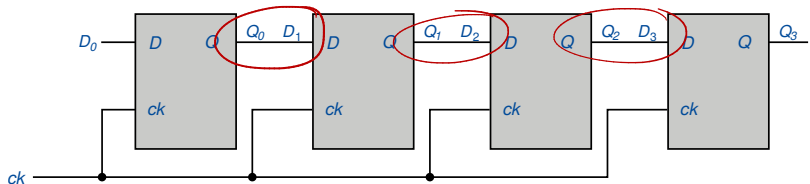
# $n$ -Bit Register

- $n$  D-Flipflops mit **gemeinsamen Clocksignal**.



- Entsprechend:  $n$ -Bit Latch =  $n$  D-Latches mit gemeinsamem Schreibsignal  $W$ .

# Schieberegister

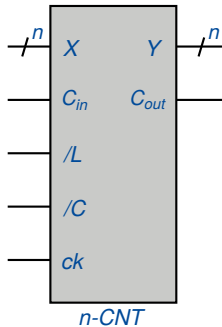


- In jedem Takt (bei jeder steigenden Flanke von  $ck$ ) werden die **Werte** im Register um **eine Position nach rechts** verschoben.

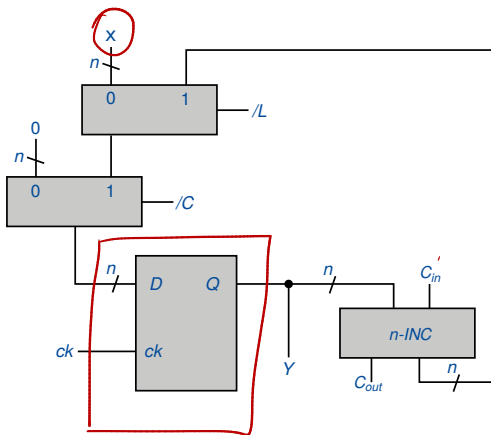
# Zähler

Ein  $n$ -Bit-Zähler ist eine Schaltung mit folgenden Ein- und Ausgängen:

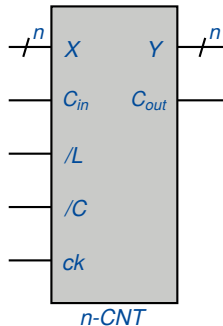
- Dateneingänge  $X = (X_{n-1}, \dots, X_0)$
- Datenausgänge  $Y = (Y_{n-1}, \dots, Y_0)$
- Dateneingang  $C_{in}$  für Eingangsübertrag
- Datenausgang  $C_{out}$  für Ausgangsübertrag
- Eingänge für Kontrollsignale:
  - $/C$  (Clear)
  - $/L$  (Load)
  - $ck$  (Clock)



# Aufbau eines Zählers

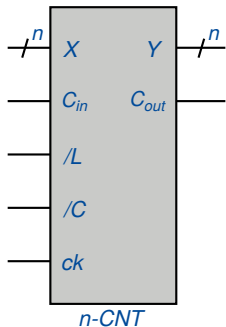


Register mit  
 $n$  Bits





# $n$ -Bit Zähler: Funktionalität



- Ein Zähler speichert ein  $n$ -Bit-Wort, das an den Ausgängen  $Y$  erscheint (**Zählerstand**).
- Bei jeder steigenden Flanke von  $ck$  wird ein neuer Zählerstand  $Y_{neu}$  gespeichert. Für  $Y_{neu}$  gilt :

$$Y_{neu} = \begin{cases} 0 \dots 0, & \text{falls } /C = 0 \\ X, & \text{falls } /C = 1, /L = 0 \\ \text{bin}_n((\langle Y \rangle + C_{in}) \bmod 2^n), & \text{falls } /C = 1, /L = 1 \end{cases}$$

# $n$ -Bit Zähler kaskadieren

- Ausgangsübertrag  $C_{out}$  ermöglicht es, den Zähler zu **kaskadieren**, zum Beispiel aus  $s$   $n$ -Bit-Zählern einen  $(s \cdot n)$ -Bit-Zähler zu bauen.

