DS 19 – Rückkopplung als Speichertechnik: Das Flipflop

Informatik 10 – MINT-Zweig | Informatiksysteme

Lernziel

Ich skizziere und erläutere eine logische Schaltung, die das Prinzip der Rückkopplung als Speichertechnik verwendet.

🕘 Stundenverlaufsplan (Doppelstunde, 90 Minuten)

Orientierungsrahmen für 11 SuS: Die ersten 80 Minuten bilden den Kern. Die letzten 10 Minuten sind bewusst flexibel und werden nicht mit neuem Pflichtstoff belegt.

Phase	Zeit	Inhalt/Aktion	Sozialform
Einstieg und Aktivierung	10'	Frage entwickeln, wie eine Schaltung einen früheren Zustand behalten kann.	Plenum; kurze Einzeldenkzeit
Fragend-entwickelnde Erarbeitung	25'	Rückkopplung und Speicherzustand anhand zweier gekoppelter Gatter erschließen.	Unterrichtsgespräch; Beiträge der SuS werden sichtbar festgehalten
Gemeinsame Umsetzung und Sicherung	20'	SR-Flipflop gemeinsam skizzieren und Zustandsänderungen vor dem Test vorhersagen.	Plenum; Live-Coding, Tafelbild oder gemeinsame Simulation
Kurzer Transfer	15'	Schaltung in CircuitVerse bauen und die Testfolge protokollieren.	Einzelarbeit mit anschließendem Tandemabgleich (4 Tandems + 1 Trio)
Auswertung	10'	Speicherwirkung mit den Begriffen Rückkopplung, Setzen und Rücksetzen erklären lassen.	Plenum; Lösung erklären und Fachsprache sichern
Reserve / Vertiefung	10'	Weitere Flipflops oder optionale Simulation nur als Vertiefung.	Flexibel; bei Bedarf für Fragen, Fehlersuche oder Abschluss nutzen

🗣️ Sprachhilfe

- „Für die Eingangswerte … liefert die Schaltung den Ausgangswert …“
- „Der Term/die Tabelle repräsentiert die Schaltung, weil …“
- „Die Rückkopplung bzw. Gruppierung bewirkt …“

📌 Merksätze

- Bisherige Schaltungen (Addierer etc.) sind **kombinatorisch**: Der Ausgang hängt nur vom **aktuellen** Eingang ab.
- Ein **Flipflop** kann sich hingegen ein Bit **merken** – der Ausgang hängt auch von der **Vorgeschichte** ab. Flipflops sind die Grundlage für schnelle Speicher im Computer, z. B. Register (der Arbeitsspeicher eines PCs speichert Bits dagegen meist als elektrische Ladung).
- Möglich wird das durch **Rückkopplung**: Der Ausgang einer Schaltung wird auf einen ihrer eigenen Eingänge zurückgeführt.
- Das einfachste Beispiel ist das **SR-Flipflop** (Set-Reset), gebaut aus zwei über Kreuz rückgekoppelten **NOR-Gattern**:
    - `S = 1` (Set) setzt den gespeicherten Wert auf `1`

- `R = 1` (Reset) setzt den gespeicherten Wert auf `0`
- `S = 0` und `R = 0`: Der zuletzt gespeicherte Wert **bleibt erhalten** – das ist der "Speicher-Effekt"!
- `S = 1` und `R = 1` gleichzeitig ist ein **verbotener Zustand** und sollte vermieden werden.

✏️ Zusatzübung – SR-Flipflop zusätzlich von Hand skizzieren

Hinweis: Der CircuitVerse-Aufbau im nächsten Schritt zusammen mit eurer schriftlichen Erklärung in Übung 2 erfüllt die Kompetenz „skizzieren und erläutern" bereits (grafische bzw. sprachliche, aufs Wesentliche reduzierte Darstellung). Diese Papier-Skizze vorab ist freiwillig, kann euch aber als Bauanleitung für den nächsten Schritt dienen.

Skizziert **freiwillig**, bevor ihr baut (auf Papier oder als Beschreibung in dieser Zelle): zwei NOR-Gatter, deren Ausgang jeweils auf einen Eingang des **anderen** Gatters zurückgeführt wird ("über Kreuz"). Beschriftet die freien Eingänge mit `S` und `R` sowie die beiden Ausgänge mit `Q` und `Q'`.

🖥️ Praxisteil: SR-Flipflop in CircuitVerse bauen

1. Legt zwei Input-Schalter `S` und `R` sowie zwei Output-LEDs (`Q` und `Q'`, das negierte Q) an.
2. Fügt **zwei NOR-Gatter** hinzu (falls kein NOR verfügbar ist: OR-Gatter + nachgeschaltetem NOT-Gatter).
3. Verschaltet **über Kreuz**:
   - Eingang des ersten NOR-Gatters: `R` und der Ausgang des **zweiten** NOR-Gatters
   - Eingang des zweiten NOR-Gatters: `S` und der Ausgang des **ersten** NOR-Gatters
4. Der Ausgang des ersten Gatters ist `Q`, der des zweiten ist `Q'`.
5. **Testet die Speicherwirkung:** Setzt kurz `S=1, R=0` → `Q` wird `1`. Schaltet danach `S` wieder auf `0` (also `S=0, R=0`) → beobachtet, dass `Q` weiterhin `1` bleibt, obwohl kein Eingang mehr aktiv ist!

📝 Übung 1 – Zustände protokollieren

Führt die Testreihenfolge unten in eurer CircuitVerse-Schaltung durch und tragt jeweils den beobachteten Wert von `Q` ein (Doppelklick zum Bearbeiten):

Schritt	S	R	Q (beobachtet)	Erklärung
1	1	0		Setzen
2	0	0		Speichern (Q bleibt wie zuvor)
3	0	1		Rücksetzen
4	0	0		Speichern (Q bleibt wie zuvor)

📝 Übung 2 – Erklärung in eigenen Worten

Beantwortet in dieser Zelle in 2–3 Sätzen: Warum kann eine **rein kombinatorische** Schaltung (ohne Rückkopplung) kein Bit speichern? Was genau ermöglicht die Rückkopplung, das eine gewöhnliche Gatterkette nicht kann?

(Eure Antwort hier)

🚀 Zusatzaufgabe (Vertiefung)

Recherchiert (z. B. über eine Websuche) den Unterschied zwischen dem hier gebauten **SR-Flipflop** und einem **D-Flipflop** oder **JK-Flipflop**. Was ist deren Vorteil gegenüber dem einfachen SR-Flipflop?

🐍 Optionale Simulation in Python

Rein kombinatorisch (ohne Speicher) lässt sich ein Flipflop nicht direkt als einfache Funktion simulieren – genau das ist ja der Punkt aus Übung 2! Mit einer **Zustandsvariablen**, die zwischen Aufrufen erhalten bleibt, lässt sich der Speicher-Effekt aber nachbilden. Optional zur Vertiefung:

```python
# Optionale Vertiefung: SR-Flipflop-Verhalten mit einer Zustandsvariable simulieren
q = 0  # Anfangszustand (nicht Teil einer "reinen" Schaltung, sondern unser Gedächtnis)

def sr_flipflop(s, r, q):
    if s == 1 and r == 0:
        return 1
    elif s == 0 and r == 1:
        return 0
    elif s == 0 and r == 0:
        return q  # Speichern: alter Zustand bleibt erhalten
    else:
        raise ValueError("S=1 und R=1 gleichzeitig ist der verbotene Zustand!")

# Testreihenfolge wie in Übung 1
for s, r in [(1, 0), (0, 0), (0, 1), (0, 0)]:
    q = sr_flipflop(s, r, q)
    print(f"S={s} R={r} -> Q={q}")
```

```
S=1 R=0 -> Q=1
S=0 R=0 -> Q=1
S=0 R=1 -> Q=0
S=0 R=0 -> Q=0
```