2.5 Rückkopplung: Das Flipflop speichert ein Bit

Leitidee

Stell dir die Alarmanlage aus dem Abschnitt über Schaltungsentwurf vor: Ein Einbrecher öffnet die Tür, und der Alarm geht los. Dann schließt er die Tür wieder – und der Alarm hört sofort auf. Das ist keine gute Alarmanlage! Die Schaltung hat kein **Gedächtnis**: Sie reagiert nur auf das, was gerade jetzt an den Eingängen anliegt. Ein Computer muss sich aber ständig etwas merken: Zahlen, Buchstaben, Zwischenergebnisse. In diesem Abschnitt lernst du eine Schaltung kennen, die sich ein Bit merken kann: das **Flipflop**. Der Trick dahinter heißt **Rückkopplung**.

Lernziele

Nach diesem Abschnitt kannst du

- erklären, worin sich eine Schaltung ohne Gedächtnis von einer speichernden Schaltung unterscheidet,
- das Prinzip der Rückkopplung erläutern,
- ein SR-Flipflop aus zwei NOR-Gattern skizzieren,
- die Zustände Setzen, Rücksetzen, Speichern und den verbotenen Zustand beschreiben,
- Schritt für Schritt nachvollziehen, wie die Rückkopplung einen Wert festhält,
- ein SR-Flipflop in CircuitVerse bauen und testen.

2.5.1 Schaltungen ohne Gedächtnis

Alle Schaltungen, die du bisher gebaut hast, funktionieren nach demselben Prinzip: Die Eingangswerte laufen von links nach rechts durch die Gatter, und am Ende kommt ein Ausgangswert heraus. Die Alarmanlage zum Beispiel berechnet

$$\text{Alarm} = (\text{Tür} \wedge \text{Scharf}) \vee \text{Rauch}.$$

Welcher Wert am Ausgang anliegt, hängt **nur** davon ab, welche Werte **jetzt gerade** an den Eingängen anliegen. Was vor einer Sekunde war, spielt keine Rolle. Genauso ist es beim Halbaddierer und beim Volladdierer: Sie rechnen mit den Bits, die gerade anliegen. Ändert sich ein Eingang, ändert sich sofort auch das Ergebnis.

Definition: Kombinatorische Schaltung

Eine **kombinatorische Schaltung** ist eine Schaltung, deren Ausgangswerte nur von den **aktuellen** Eingangswerten abhängen. Zu jeder Kombination von Eingangswerten gehört immer derselbe Ausgangswert.

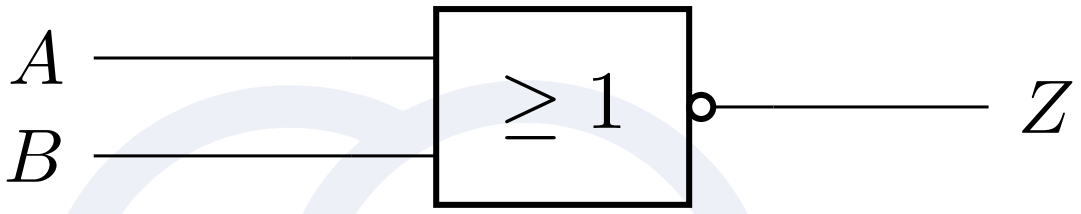
Für die Alarmanlage heißt das: Sobald die Tür wieder zu ist (Tür = 0) und kein Rauch da ist, ist auch der Alarm aus. Wir wollen aber eine Anlage, die **weiter heult**, bis jemand sie mit einer Taste ausschaltet. Der Ausgang soll also auch davon abhängen, was **vorher** passiert ist – von der **Vorgeschichte**. Dafür brauchen wir eine Schaltung, die sich etwas merken kann.

Merke

- Bisherige Schaltungen (Addierer usw.) sind **kombinatorisch**: Der Ausgang hängt nur vom **aktuellen** Eingang ab.
- Ein **Flipflop** kann sich dagegen ein Bit **merken**: Der Ausgang hängt auch von der **Vorgeschichte** ab. Flipflops sind die Grundlage für Speicher im Computer, zum Beispiel für die Register im Prozessor.

2.5.2 Ein neues Gatter: das NOR-Gatter

Für das Flipflop brauchst du ein Gatter, das du noch nicht kennst: das **NOR-Gatter**. Der Name ist eine Abkürzung für „NOT OR“, also NICHT-ODER: Das Gatter rechnet zuerst ein ODER und dreht das Ergebnis dann um. Im IEC-Symbol steht deshalb ≥ 1 wie beim ODER-Gatter, und am Ausgang sitzt ein kleiner Kreis. Dieser Kreis bedeutet „umdrehen“. In manchen Büchern steht an seiner Stelle ein kleiner Schrägstrich. Das bedeutet dasselbe.



Das NOR-Gatter (IEC-Symbol): ein ODER-Gatter mit dem Zeichen für „umdrehen“ am Ausgang

A	B	$A \vee B$	**NOR:** $\neg(A \vee B)$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Für das Flipflop ist eine Eigenschaft besonders wichtig. Lies sie direkt aus der Tabelle ab:

- Liegt an **einem** Eingang eine 1, ist der Ausgang **sicher 0** – egal, was am anderen Eingang anliegt.
- Nur wenn **beide** Eingänge 0 sind, ist der Ausgang 1. Dann gibt das Gatter sozusagen die Umkehrung des anderen Eingangs weiter.

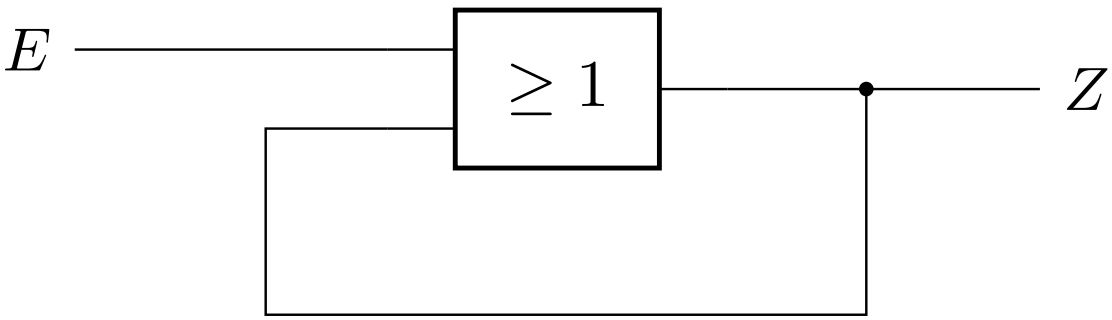
Tipp

Merke dir für die folgenden Abschnitte den kurzen Satz: **„Eine 1 am Eingang macht den NOR-Ausgang zu 0.“** Damit kannst du jedes Flipflop-Rätsel lösen.

2.5.3 Die Idee: Rückkopplung

Bisher liefen alle Leitungen nur in eine Richtung: von den Eingängen zum Ausgang. Jetzt machen wir etwas Neues: Wir führen den Ausgang eines Gatters **zurück** auf einen seiner eigenen Eingänge. Das nennt man **Rückkopplung**.

Ein ganz einfaches Beispiel ist ein ODER-Gatter, dessen Ausgang Z auf den zweiten Eingang zurückgeführt wird:



Ein ODER-Gatter mit Rückkopplung: Der Ausgang Z wird auf den unteren Eingang zurückgeführt

Wir nehmen an, dass am Anfang $E = 0$ und $Z = 0$ ist. Dann passiert Folgendes:

1. Solange $E = 0$ ist, rechnet das Gatter $0 \vee 0 = 0$. Der Ausgang bleibt 0.
2. Jetzt wird E kurz auf 1 gestellt. Das Gatter rechnet $1 \vee 0 = 1$, also wird $Z = 1$.
3. Diese 1 läuft über die Rückkopplung zurück an den unteren Eingang.
4. Nun wird E wieder auf 0 gestellt. Das Gatter rechnet $0 \vee 1 = 1$. Der Ausgang **bleibt 1**!

Die Schaltung hat sich gemerkt, dass E einmal 1 war. Der Ausgang hält sich über die Rückkopplung selbst fest. Leider hat diese Schaltung einen Haken: Die 1 lässt sich nie wieder löschen. Für einen brauchbaren Speicher brauchen wir also noch einen zweiten Eingang, mit dem man den Wert wieder auf 0 setzen kann. Genau das leistet das Flipflop, das du gleich kennenlernst.

Definition: Rückkopplung

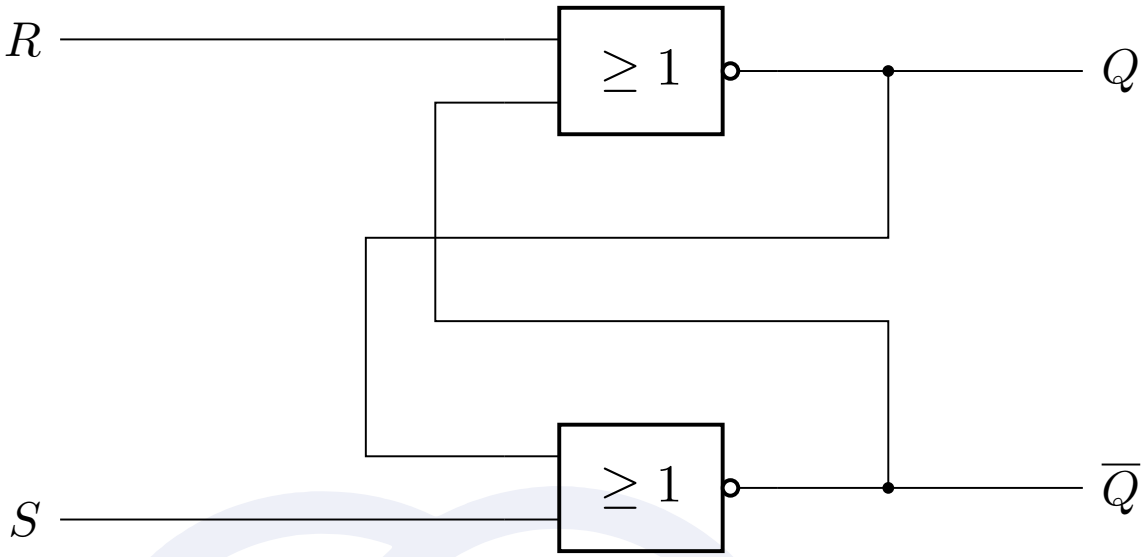
Bei einer **Rückkopplung** wird der Ausgang einer Schaltung auf einen ihrer eigenen Eingänge zurückgeführt. Dadurch hängt der Ausgang auch von seinem eigenen früheren Wert ab.

Merke

Möglich wird das Speichern durch **Rückkopplung**: Der Ausgang einer Schaltung wird auf einen ihrer eigenen Eingänge zurückgeführt.

2.5.4 Das SR-Flipflop aus zwei NOR-Gattern

Das einfachste Flipflop besteht aus nur zwei NOR-Gattern. Der Ausgang jedes Gatters wird an einen Eingang des **anderen** Gatters zurückgeführt. Man sagt: Die Gatter sind **über Kreuz** zurückgekoppelt.



SR-Flipflop aus zwei NOR-Gattern: Die Ausgänge sind über Kreuz zurückgekoppelt

Schau dir die Abbildung in Ruhe an und verfolge jede Leitung mit dem Finger:

- Das **obere** Gatter bekommt den Eingang R und den Ausgang $\overline{Q}$ des unteren Gatters. Sein eigener Ausgang heißt Q.
- Das **untere** Gatter bekommt den Eingang S und den Ausgang Q des oberen Gatters. Sein eigener Ausgang heißt $\overline{Q}$.
- Wo eine Leitung sich verzweigt, sitzt ein Punkt. Wo sich zwei Leitungen nur kreuzen, gibt es keinen Punkt.

Die Namen kommen aus dem Englischen:

Name	englisch	Bedeutung
S	*Set* (setzen)	setzt den gespeicherten Wert auf 1
R	*Reset* (zurücksetzen)	setzt den gespeicherten Wert auf 0
Q	–	der gespeicherte Wert, also das Bit, das sich das Flipflop merkt
$\overline{Q}$	–	gesprochen „Q quer“; im Normalfall das Gegenteil von Q

In CircuitVerse und vielen Büchern schreibt man statt $\overline{Q}$ auch `Q'`.

Definition: Flipflop

Ein **Flipflop** ist eine Schaltung mit Rückkopplung, die ein Bit speichern kann. Der gespeicherte Wert bleibt erhalten, bis er über einen Eingang geändert wird.

Die vier Fälle

Je nachdem, welche Werte an S und R anliegen, macht das Flipflop etwas anderes:

S	R	Q danach	Name des Zustands
1	0	1	Setzen
0	1	0	Rücksetzen
0	0	wie vorher	Speichern
1	1	–	verboten

Neu ist die Zeile mit $S = 0$ und $R = 0$. Dort steht kein fester Wert, sondern „wie vorher“. Das gibt es bei einer kombinatorischen Schaltung nie! Das Flipflop behält einfach den Wert, den es zuletzt hatte. Genau das ist der **Speicher-Effekt**.

Merke

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.

Arbeitsauftrag 53: SR-Flipflop skizzieren

Decke die Abbildung des SR-Flipflops ab. Skizziere das SR-Flipflop dann aus dem Gedächtnis: zwei NOR-Gatter, deren Ausgang jeweils auf einen Eingang des **anderen** Gatters zurückgeführt wird. Beschrifte die freien Eingänge mit S und R und die beiden Ausgänge mit Q und $\overline{Q}$. Vergleiche zum Schluss mit der Abbildung.

Meine Skizze:

Gymnasium Ottweiler

2.5.5 Schritt für Schritt: Was in der Rückkopplung passiert

Warum merkt sich die Schaltung wirklich etwas? Das siehst du, wenn du die Gatter **nacheinander** ausrechnest. In echt geht das blitzschnell, aber eben nicht gleichzeitig: Jedes Gatter reagiert erst, wenn sich einer seiner Eingänge geändert hat.

Ausgangslage: $S = 0$, $R = 0$, $Q = 0$ und $\overline{Q} = 1$. Prüfe, dass das zusammenpasst: Das obere Gatter bekommt $R = 0$ und $\overline{Q} = 1$. Eine 1 am Eingang macht den Ausgang zu 0, also $Q = 0$. ✓ Das untere Gatter bekommt $S = 0$ und $Q = 0$. Beide Eingänge sind 0, also $\overline{Q} = 1$. ✓ Die Schaltung ist **stabil** – nichts ändert sich. Sie speichert die 0.

Jetzt wird S kurz auf 1 gestellt und danach wieder auf 0:

Schritt	S	R	Was passiert?	Q	$\overline{Q}$
0	0	0	Ausgangslage: Das Flipflop speichert 0.	0	1
1	1	0	S wird 1. Das untere Gatter hat eine 1 am Eingang, also wird $\overline{Q} = 0$.	0	0
2	1	0	Das obere Gatter bekommt jetzt $R = 0$ und $\overline{Q} = 0$. Beide sind 0, also wird $Q = 1$.	1	0
3	1	0	Das untere Gatter bekommt $S = 1$ und $Q = 1$: $\overline{Q}$ bleibt 0. Nichts ändert sich mehr.	1	0
4	0	0	S wird wieder 0. Das untere Gatter bekommt aber noch $Q = 1$ über die Rückkopplung: $\overline{Q}$ bleibt 0. Also bleibt auch $Q = 1$.	1	0

Schritt 4 ist der entscheidende: Der Eingang S ist wieder 0, trotzdem bleibt $Q = 1$. Die 1 an Q hält über die Rückkopplung das untere Gatter auf 0 fest, und diese 0 hält wiederum das obere Gatter auf 1. Die beiden Gatter **stützen sich gegenseitig**. Das Flipflop hat sich die 1 gemerkt.

So kannst du es sagen

So beschreibst du das Flipflop in Fachsprache – vom einfachen zum genauen Satz:

- *„Man kann sich damit eine 0 oder 1 merken."* → *„Die Schaltung speichert eine 0 oder eine 1."*
- *„Die Schaltung kann ein Bit speichern."* → *„Die Schaltung ist in der Lage, ein Bit zu speichern."*
- *„Das Ding merkt sich den Wert, weil der Ausgang zurückgeht."* → *„Das Flipflop nutzt die Technik der Rückkopplung und ist so in der Lage, ein Bit zu speichern. Die Rückkopplung bewirkt, dass der Wert erhalten bleibt."*

Arbeitsauftrag 54: Rücksetzen Schritt für Schritt

Das Flipflop speichert jetzt die 1: $S = 0$, $R = 0$, $Q = 1$, $\overline{Q} = 0$. Nun wird R kurz auf 1 gestellt und danach wieder auf 0. Fülle die Tabelle aus. Gehe vor wie im Beispiel oben: Rechne immer nur **ein** Gatter aus und beschreibe kurz, was passiert.

Schritt	S	R	Was passiert?	Q	$\overline{Q}$
0	0	0	Ausgangslage: Das Flipflop speichert 1.	1	0
1	0	1			
2	0	1			
3	0	1			
4	0	0			

Der verbotene Zustand

Was passiert, wenn $S = 1$ **und** $R = 1$ gleichzeitig anliegen? Beide Gatter haben dann eine 1 am Eingang, also sind **beide** Ausgänge 0: $Q = 0$ und $\overline{Q} = 0$. Das widerspricht der Idee, dass $\overline{Q}$ das Gegenteil von Q ist. Findest du also in einer Tabelle $Q = 0$ und $\overline{Q} = 0$, dann liegt (oder lag gerade) $S = R = 1$ an.

Schlimmer ist, was danach kommt: Werden S und R **gleichzeitig** wieder 0, sehen beide Gatter nur noch Nullen und wollen beide auf 1 springen. Welches Gatter „gewinnt“, hängt von winzigen Unterschieden in der Geschwindigkeit der Bauteile ab. Was danach gespeichert ist, lässt sich nicht vorhersagen. Deshalb nennt man $S = R = 1$ den **verbotenen Zustand**. Eine Schaltung, die ein SR-Flipflop benutzt, muss dafür sorgen, dass er nie vorkommt.

2.5.6 Das Flipflop in Python nachbilden

Eine Python-Funktion, wie du sie im Kapitel über Programmierung kennengelernt hast, hat kein Gedächtnis: Zu denselben Argumenten liefert sie immer dasselbe Ergebnis – genau wie eine kombinatorische Schaltung. Ein Flipflop lässt sich deshalb nicht einfach als Funktion von S und R schreiben. Man kann den Speicher-Effekt aber nachbilden, wenn man den alten Wert von Q in einer Variablen aufbewahrt und ihn der Funktion jedes Mal wieder mitgibt:

```python
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
    else:
        print("Achtung: S = 1 und R = 1 ist der verbotene Zustand!")
        return 0

s_folge = [1, 0, 0, 0]
r_folge = [0, 0, 1, 0]
q = 0
for i in range(4):
    q = sr_flipflop(s_folge[i], r_folge[i], q)
    print(f"S={s_folge[i]} R={r_folge[i]} -> Q={q}")
```

AUSGABE

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

- **Zeilen 1–10:** Die Funktion `sr_flipflop` bekommt die Eingänge `s` und `r` und den **bisher gespeicherten** Wert `q`. Sie gibt den neuen Wert von Q zurück.
- **Zeilen 2–5:** Setzen liefert 1, Rücksetzen liefert 0.
- **Zeilen 6–7:** Speichern: Der alte Wert `q` wird unverändert zurückgegeben. Das ist der Speicher-Effekt.
- **Zeilen 8–10:** Bleibt nur noch $S = R = 1$ übrig. Die Funktion gibt eine Warnung aus und liefert 0, denn im verbotenen Zustand ist $Q = 0$.
- **Zeilen 12–13:** Die beiden Listen enthalten die Testreihenfolge: Setzen, Speichern, Rücksetzen, Speichern. An derselben Stelle `i` stehen jeweils die Werte von S und R, die zusammengehören.
- **Zeile 14:** Am Anfang speichert das Flipflop 0.
- **Zeilen 15–17:** Für jeden Schritt wird der neue Wert berechnet und **in derselben Variablen `q` gespeichert**. Im nächsten Schritt wird er der Funktion wieder mitgegeben. Die Variable `q` übernimmt also die Rolle der Rückkopplung.

Wichtig: Das Programm simuliert nur das **Verhalten** des Flipflops, nicht die einzelnen Gatter. Das Gedächtnis steckt hier in der Variablen `q`, in der echten Schaltung dagegen in der Rückkopplung.

Versuch: Probiere es aus

Ändere die beiden Listen so, dass einmal $S = 1$ und $R = 1$ gleichzeitig vorkommt. Was gibt das Programm aus? Setze danach zweimal hintereinander $S = 1$, $R = 0$. Ändert sich beim zweiten Mal etwas?

Arbeitsauftrag 55: Zustandsfolge vorhersagen

Das Flipflop speichert am Anfang die 0. Dann liegen nacheinander diese Werte an:

Schritt	1	2	3	4	5	6	7	8
S	0	1	0	1	0	0	0	1
R	0	0	0	0	1	0	1	0
Q danach								

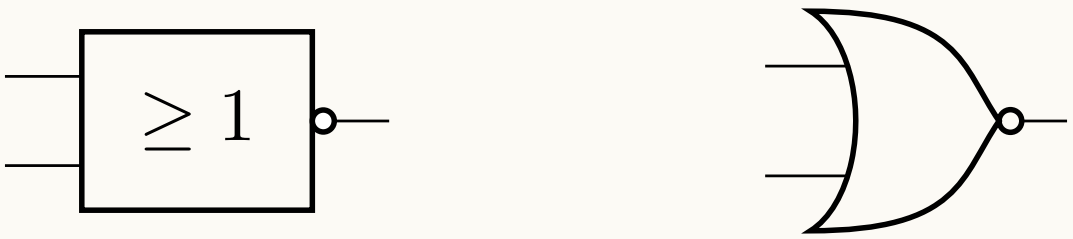
1. Trage ein, welchen Wert Q nach jedem Schritt hat.
2. Überprüfe deine Lösung mit dem Python-Programm oben. Ändere dazu die beiden Listen und die Anzahl der Schritte.

2.5.7 Das Flipflop in CircuitVerse bauen

Jetzt baust du das SR-Flipflop im Simulator CircuitVerse und testest den Speicher-Effekt selbst.

Tipp

CircuitVerse zeigt die US-Symbole für Gatter. Das NOR-Gatter sieht dort aus wie das gebogene ODER-Symbol mit einem kleinen Kreis an der Spitze:



Das NOR-Gatter: links das IEC-Symbol (Skript), rechts das US-Symbol (CircuitVerse)

Falls du kein NOR-Gatter findest, kannst du ein ODER-Gatter mit einem nachgeschalteten NICHT-Gatter nehmen. Das ergibt dasselbe.

So gehst du vor:

1. Lege zwei Eingänge (Schalter) S und R sowie zwei Ausgänge (LEDs) Q und `Q'` an.
2. Füge **zwei NOR-Gatter** hinzu.
3. Verschalte sie **über Kreuz**:
   - Das erste NOR-Gatter bekommt R und den Ausgang des **zweiten** NOR-Gatters.
   - Das zweite NOR-Gatter bekommt S und den Ausgang des **ersten** NOR-Gatters.
4. Der Ausgang des ersten Gatters ist Q, der Ausgang des zweiten ist `Q'`.
5. Teste die Speicherwirkung: Setze kurz $S = 1$, $R = 0$. Dann wird Q zu 1. Schalte danach S wieder auf 0, also $S = 0$, $R = 0$. Beobachte: Q bleibt 1, obwohl kein Eingang mehr aktiv ist!

Stolperstein

- **S und R vertauscht:** Wenn bei dir $S = 1$ den Ausgang Q auf 0 setzt, hast du die Eingänge vertauscht. R gehört an das Gatter, dessen Ausgang Q ist. (Merkhilfe: Das Gatter mit R liefert Q. $R = 1$ macht dieses Gatter zu 0, also setzt R das Q auf 0 zurück.)
- **Unklarer Anfang:** Direkt nach dem Aufbau ist manchmal nicht klar, was gespeichert ist. Vielleicht sind sogar beide Ausgänge gleich. Setze dann einmal S auf 1 und wieder auf 0. Danach ist alles eindeutig.

Arbeitsauftrag 56: Zustände protokollieren

Führe die Testreihenfolge in deiner CircuitVerse-Schaltung durch. Trage jeweils den beobachteten Wert von Q ein.

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)

2.5.8 Wozu braucht man Flipflops?

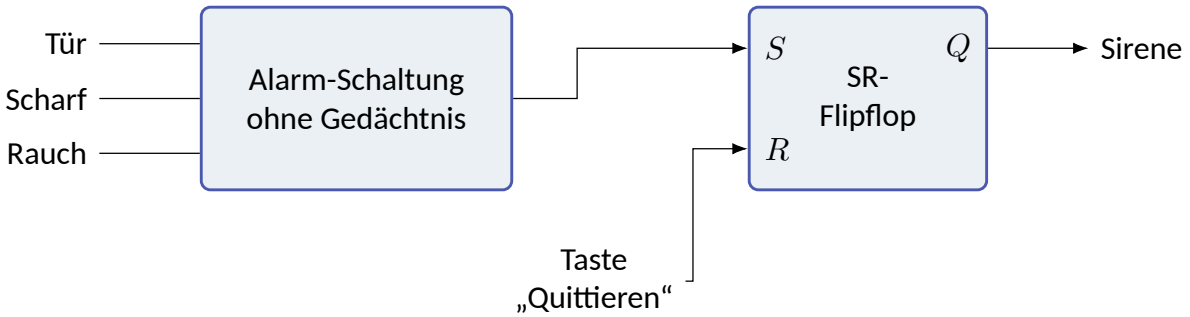
Ein Speicher für ein Bit

Ein Flipflop speichert genau **ein Bit**. Setzt man acht Flipflops nebeneinander, kann man ein ganzes Byte speichern, zum Beispiel eine Zahl von 0 bis 255 oder einen Buchstaben. So sind die **Register** im Prozessor aufgebaut: kleine, sehr schnelle Speicher, in denen der Prozessor die Zahlen ablegt, mit denen er gerade rechnet. Auch die schnellen Zwischenspeicher im Prozessor (der *Cache*) bestehen aus Speicherzellen, die wie Flipflops funktionieren.

Der große Arbeitsspeicher (RAM) eines PCs arbeitet meist mit einer anderen Technik: Dort wird jedes Bit als winzige elektrische Ladung gespeichert. Das Prinzip ist aber dasselbe: Eine Speicherzelle merkt sich eine 0 oder eine 1, bis sie überschrieben wird.

Eine Alarmanlage mit Gedächtnis

Jetzt kannst du das Problem vom Anfang lösen. Der Ausgang der bisherigen Alarm-Schaltung wird an den Eingang S eines Flipflops angeschlossen. An R kommt eine Taste „Quittieren“, mit der man den Alarm ausschaltet. Die Sirene hängt an Q.



Alarmanlage mit Gedächtnis: Die Alarm-Schaltung setzt das Flipflop, die Taste setzt es zurück

- Öffnet der Einbrecher die Tür, liefert die Alarm-Schaltung eine 1. Das Flipflop wird **gesetzt**, die Sirene heult.
- Schließt er die Tür, liegt an S wieder 0 an. Mit $R = 0$ wird **gespeichert**: Die Sirene heult weiter.
- Erst wenn jemand die Taste drückt ($R = 1$), wird das Flipflop **zurückgesetzt** und die Sirene verstummt.

Arbeitsauftrag 57: Warum Rückkopplung?

Beantworte 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?

Meine Erklärung:

Arbeitsauftrag 58 (Zusatz): Andere Flipflops

Recherchiere (zum Beispiel im Internet) den Unterschied zwischen dem SR-Flipflop und einem **D-Flip-flop** oder einem **JK-Flipflop**. Welchen Vorteil haben sie gegenüber dem einfachen SR-Flipflop?

Meine Ergebnisse:

2.5.9 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Arbeitsauftrag 59: Mit oder ohne Gedächtnis? ●○○

Entscheide bei jedem Beispiel, ob es sich wie eine kombinatorische Schaltung verhält (ohne Gedächtnis) oder ob es sich etwas merkt. Begründe kurz.

a) der Halbaddierer
b) die Alarmanlage $\text{Alarm} = (\text{Tür} \wedge \text{Scharf}) \vee \text{Rauch}$
c) das SR-Flipflop
d) die Haltewunsch-Lampe im Bus: Sie geht an, wenn jemand die Stopp-Taste drückt, und bleibt an, bis der Bus an der Haltestelle die Türen öffnet.
e) eine Türklingel: Sie tönt nur, solange jemand auf den Knopf drückt.

Deine Antwort:

Gymnasium Ottweiler

Arbeitsauftrag 60: Die vier Fälle ●○○

Das SR-Flipflop speichert gerade die 1. Ergänze für jede Zeile den Namen des Zustands und den Wert von Q danach. Wo es keinen sinnvollen Wert gibt, schreibe einen Strich.

S	R	Name des Zustands	Q danach
1	0		
0	1		
0	0		
1	1		

Arbeitsauftrag 61: Eine kurze Folge ●○○

Das Flipflop speichert am Anfang die 1. Dann liegen nacheinander diese Werte an. Trage ein, welchen Wert Q nach jedem Schritt hat.

Schritt	1	2	3	4	5
S	0	0	0	1	0
R	0	1	0	0	0
Q danach					

Arbeitsauftrag 62: Passende Eingänge finden ●●○

Jetzt umgekehrt: Das Flipflop speichert am Anfang die 0. Q soll nach den einzelnen Schritten die Werte in der unteren Zeile haben. Bestimme passende Werte für S und R. Der verbotene Zustand darf nicht vorkommen.

Schritt	1	2	3	4	5	6
S						
R						
Q danach	0	1	1	0	0	1

Arbeitsauftrag 63: Haltewunsch im Bus mit Python ●●○

Die Haltewunsch-Lampe im Bus verhält sich wie ein SR-Flipflop: Die Stopp-Taste ist S, das Öffnen der Türen ist R, die Lampe ist Q. Am Anfang ist die Lampe aus.

Schritt	1	2	3	4	5	6
Taste	0	1	0	1	0	0
Tür	0	0	0	0	1	0

1. Sage voraus, ob die Lampe nach jedem Schritt leuchtet.
2. Prüfe deine Vorhersage mit Python. Verwende die Funktion `sr_flipflop` aus dem Abschnitt *Das Flipflop in Python nachbilden* und zwei Listen `taste` und `tuer`. Gib in jedem Schritt eine Zeile wie `Schritt 2: Taste=1 Tür=0 -> Lampe=1` aus.

Deine Vorhersage:

Arbeitsauftrag 64: Ein rätselhafter Baustein ●●●

Ein Baustein mit den Eingängen X und Y und dem Ausgang Z wurde getestet. Das ist das Protokoll:

Schritt	1	2	3	4	5	6
X	1	0	0	0	1	0
Y	0	0	1	0	0	0
Z	1	1	0	0	1	1

1. Begründe mit dem Protokoll, dass der Baustein **keine** kombinatorische Schaltung sein kann.
2. Prüfe, ob sich der Baustein wie ein SR-Flipflop verhält, wenn X der Eingang S und Y der Eingang R ist.
3. In Schritt 7 liegt $X = 0, Y = 1$ an, in Schritt 8 $X = 0, Y = 0$. Sage Z für beide Schritte voraus.

Deine Antwort:

2.5.10 Das Wichtigste in Kürze

- **Kombinatorische Schaltungen** (z. B. Addierer, Alarmanlage) haben kein Gedächtnis: Der Ausgang hängt nur von den aktuellen Eingängen ab.
- Bei der **Rückkopplung** wird ein Ausgang auf einen Eingang derselben Schaltung zurückgeführt. Dadurch kann eine Schaltung einen Wert festhalten.
- Ein **Flipflop** speichert ein Bit; sein Ausgang hängt auch von der Vorgeschichte ab. Das **SR-Flipflop** besteht aus zwei über Kreuz rückgekoppelten NOR-Gattern.
- $S = 1$ setzt Q auf 1, $R = 1$ setzt Q auf 0, bei $S = R = 0$ wird gespeichert. $S = R = 1$ ist verboten.
- Flipflops sind die Grundlage von Speichern wie den Registern im Prozessor.

Selbstcheck

- ☐ Ich kann erklären, warum eine kombinatorische Schaltung kein Bit speichern kann.
- ☐ Ich kann erklären, was Rückkopplung bedeutet.
- ☐ Ich kann ein SR-Flipflop aus zwei NOR-Gattern skizzieren und richtig beschriften.
- ☐ Ich kenne die Zustände Setzen, Rücksetzen, Speichern und den verbotenen Zustand.
- ☐ Ich kann Schritt für Schritt erklären, warum Q nach dem Setzen erhalten bleibt.
- ☐ Ich kann zu einer Folge von Eingangswerten angeben, welchen Wert Q jeweils hat.