# Speicher

Mit Logikgattern kann man sogenanntes "statisches RAM" bauen. Davon abzugrenzen ist *dynamisches* RAM. Die Speicherzellen von dynamischem RAM bestehen aus Kondensatoren, die geladen oder entladen sind. Aufgrund der Flüchtigkeit der Ladung müssen die Zellen alle paar Millisekunden aufgefrischt werden.

Statisches RAM ist schneller als dynamisches RAM, benötigt aber mehr Platz. Daher wird diese Art von Speicherzellen heutzutage vor allem für Cache-Speicher (direkt in der CPU, derzeit bis zu ca. 8 MB) verwendet.

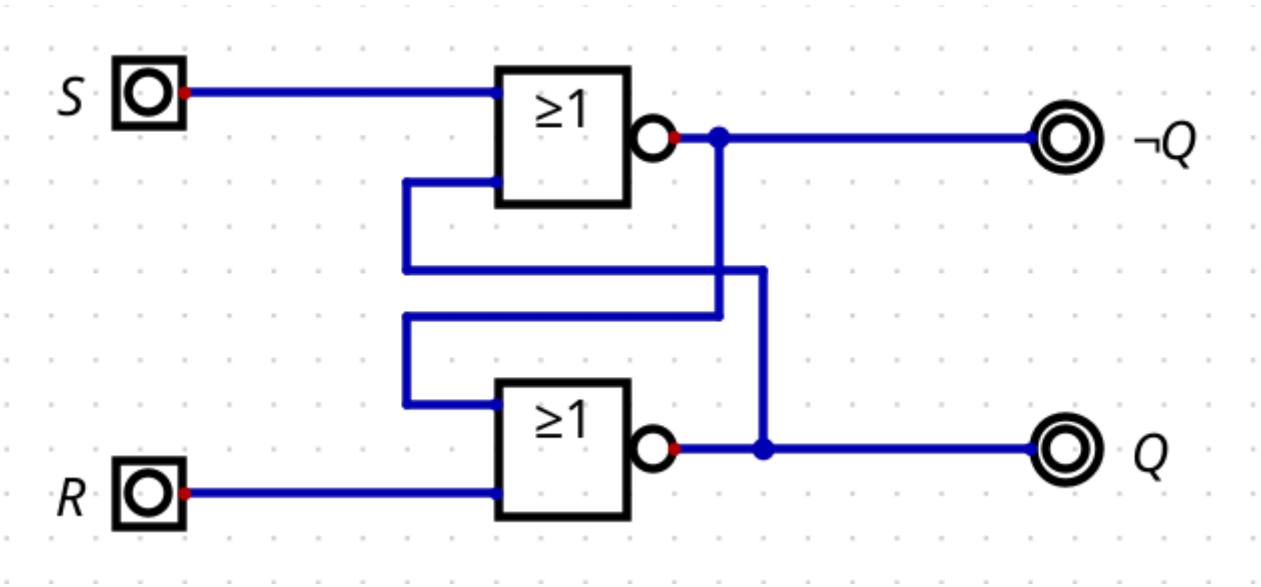
Eine weitere Anwendung von statischem RAM ist die Speicherung von BIOS-Einstellungen, da der Speicher mit wenig Energie jahrelang gehalten werden kann ohne dass die Speicherzellen ständig "aufgefrischt" werden müssen.

Dynamisches RAM wird z.B. für den Hauptspeicher verbaut, die Inhalte gehen verloren, wenn die zur Auffrischung der Speicherzellen nötige Energie ausbleibt.

## RS-Flip-Flop

**(A1)**

Baue die folgende Schaltung in deiner Logiksimulation auf:




- Achte genau auf die verwendeten Bauteile - um was für welche handelt es sich?
- Untersuche das Verhalten der Schaltung in der Simulation. S steht für "Set", R für "Reset" - kannst du erläutern, warum diese Bezeichnungen sinnvoll sind?
- Erläutere warum diese Schaltung als Speicherelement dienen kann. Wieviele Bit können

gespeichert werden?
- Es gibt einen "verboteten Zustand" für die Eingänge dieser Schaltung - welcher? Begründe, warum dieser Zustand nicht eintreten sollte.

Lösungshinweis

Der Eingangszustand 1/1 ist verboten:

- Dann ist Q und ¬Q `0`, das darf nicht sein.
- Außerdem gibt es ein Problem, wenn die Schaltung den Zustand von S=1/R=1 auf S=0/R=0 ändert. Es ist anzunehmen, dass die Änderung nicht exakt zeitgleich erfolgen kann. Das heißt, einer der Eingänge wird etwas früher auf den Wert `0` "umschalten" als der andere - welcher, ist nicht vorhersehbar. Das Ergebnis ist ein Set- oder ein Reset-Vorgang - welcher ist aber nicht vorhersehbar.

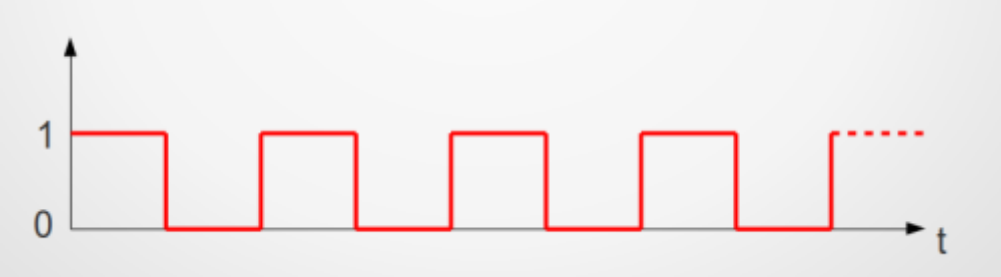
---

**(A2)**

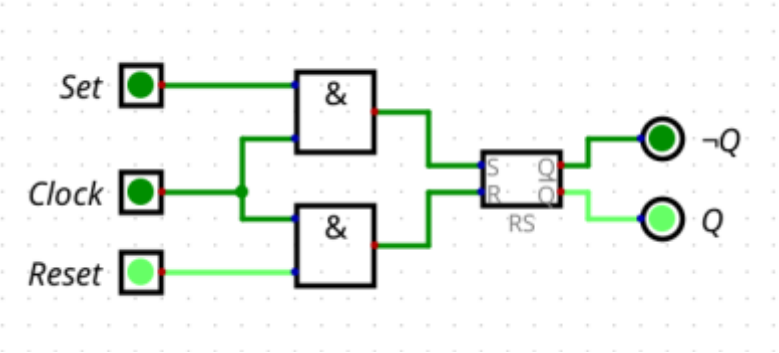
Lies die die Informationen auf dem Merkzettel RS-Flip-Flop durch. Was ist das zentrale Problem dieser Schaltung?

# Getakteter Speicher

Um das Problem der "verbotenen" Eingabe zu lösen, kann man sogenannten "getakteten Speicher" verwenden. Getakteter Speicher benötigen ein Taktsignal, welches festlegt, wann der Speicherzustand geändert werden kann. Der Speicherinhalt kann beispielsweise nur dann geändert werden, wenn das folgende Taktsignal den Wert 1 annimmt:




Ein Möglichkeit, das zu erreichen ist folgende Schaltung:




der Ablauf ist wie folgt:

- Solange der Wert im Taktzyklus 0 ist wird dan den Steuerleitungen die gewünschte Aktion (Set/Reset) "eingestellt". Das hat zunächst keine Auswirkung, da gleichgültig welcher der beiden Eingänge gesetzt wird, am Ausgang der UND-Gattern niemals 1 anliegt, solange, der Clock-Eingang auf `0` ist und somit kein Eingang des Flip-Flop angesteuert wird.
- Wenn nun der Clock-Eingang auf 1 wechselt, wird die Einstellung "übernommen".
- Es ist noch immer verboten, an den beiden Eingängen R und S gleichzeitig 1 anzulegen, da der Ausgang dann sinnfrei auf `0,0` wechseln würde - der Zufallsfaktor des RS FlipFlops beim Wechsel des Zustands wird durch den Takt jedoch eliminiert, man kann eindeutig vorhersagen, wie sich das Flip-Flop im nächsten Takt verhalten wird.

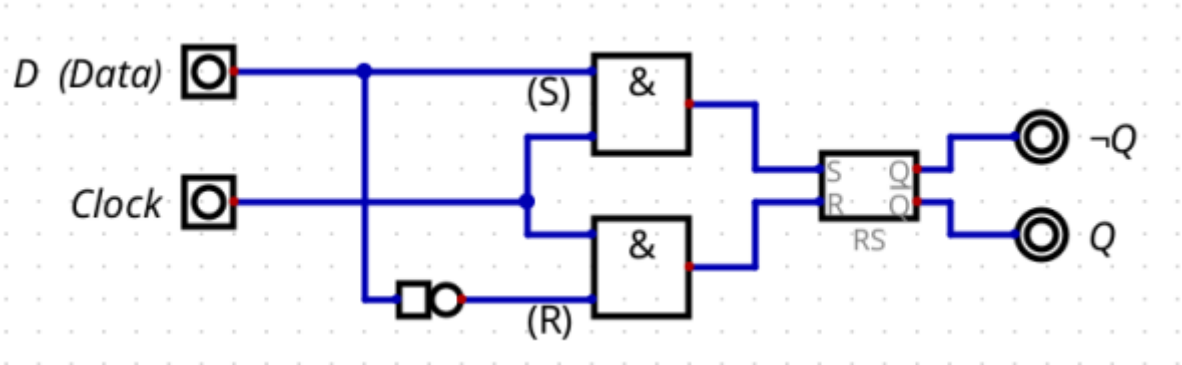
---

**(A3)**

Baue die Schaltung in der Simulation auf und teste, ob das Verhalten der Beschreibung oben entspricht.

# D-Flip-Flop

Das **D-Flip-Flop** besteht aus einem RS-Flip-Flop, bei dem der Rücksetzeingang zum Setzeingang negiert ist. Dadurch wird verhindert, dass der verbotene Zustand eintritt. Das D-Flip-Flop gibt es als auch als taktzustandsgesteuertes Flip-Flop.



- Wenn C = 0 ist, bleibt der Zustand unverändert.
- Durch das NICHT-Gatter vor dem R-Eingang wird verhindert, dass die verbotene 1,1-Eingabe anliegt. Wenn D=1 ist ist R=0 und umgekehrt.
- Wenn D beim Taktsignal 1 ist, liegt innen (1/0) an, das Flipflop speichert die 1.
- Wenn D beim Taktsignal 0 ist, liegt (0/1) an, es wird 0 gespeichert.
- In der Folge wird also stets den Zustand des Eingangs D im Moment des Taktsignals gespeichert.

---

**(A4)**

Baue ein D-Flip-Flop in der Simulation auf und teste, ob das Verhalten der Beschreibung oben entspricht.

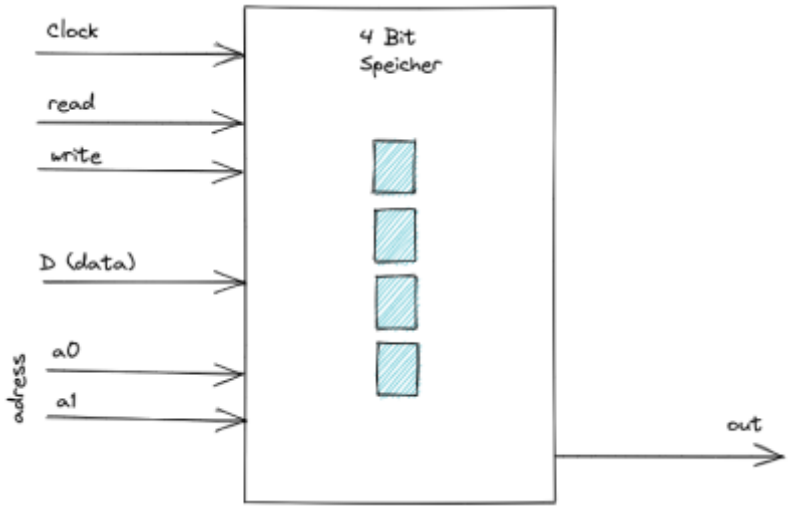
---

**(A5)****

Entwerfe in der Simulation ein einen 4-Bit-RAM Speicher. Deine Schaltung soll 4 Bit Speicher simulieren, die gelesen und geschrieben werden können.

**Benötigte Ein- und Ausgänge:**

- Ein Taktsignalgeber (der von Hand ausgelöst wird, wie bei den Beispielen oben) - die Clock
- Je ein Eingang Read und Write:
  - Wenn das Signal eines dieser Eingänge auf 1 ist, soll aus dem Speicher gelesen bzw. hineingeschrieben werden.
- Ein Dateneingang, der festlegt, welcher Wert (0 oder 1) in den Speicher geschrieben wird.
- Zwei Adresseingänge, die bestimmen, welche Speicher-Zelle zur Speicherung verwendet werden soll.
- Ein Datenausgang, der (beim Lesen) das gelesene Bit ausgibt.




**Erwartetes Verhalten:**

- Solange der `clock` Eingang 0 ist, passiert grundsätzlich nichts.
- Wenn `r=1` ist, liegt an `out` der Wert an, der in der durch `a0` und `a1` adressierten Speicherzelle gespeichert ist.
- Wenn `w=1` ist, wird der Wert, der an `D` anliegt in die durch `a0` und `a1` adressierte Speicherzelle geschrieben.

**Hinweis: Multiplexer** und **Demultiplexer** sowie D-**Flipflops** sind hier hilfreich! Du muss diese Bauteile nicht aus den Grundgattern neu aufbauen sondern kannst sich in der Bauteilbibliothek bedienen