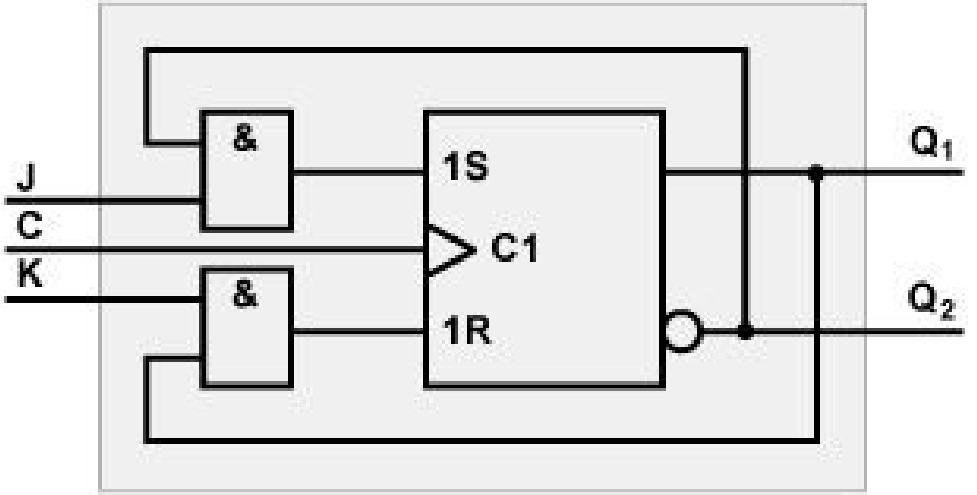
Organisation

JK-Flipflop

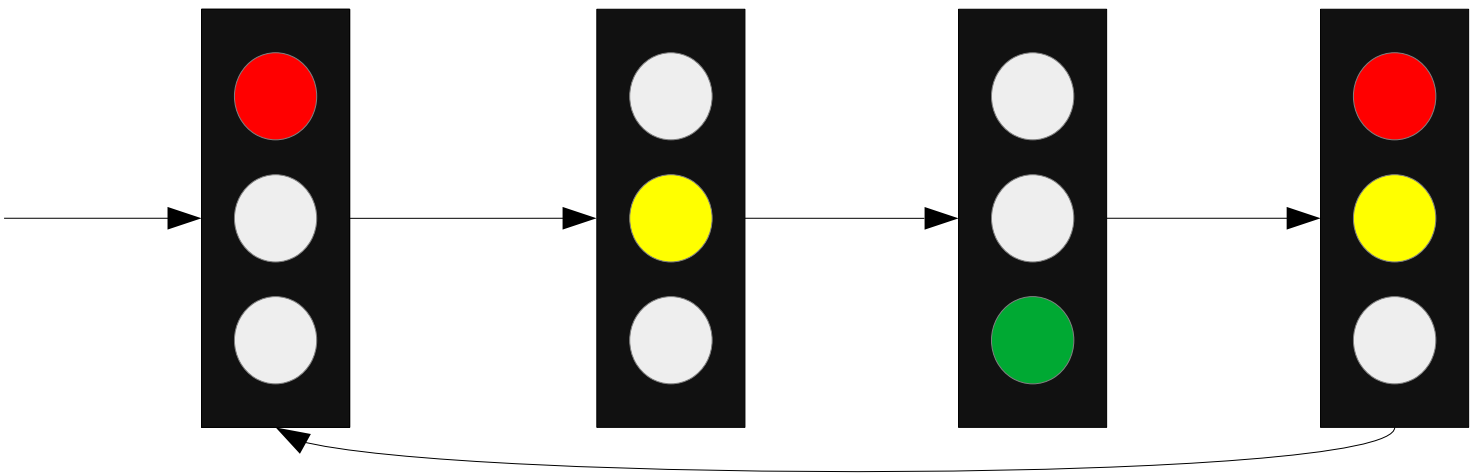
- Entschärfe verbotenen Zustand vom RS-FF
- Schalte nur, wenn es Q zulässt
- Prüfe Set und Q bzw. Reset und $\overline{Q}$
- Verbotener Eingang = Toggle (im Takt)

J	K	Q	$\overline{Q}$	Funktion
0	0	*	*	Speichern
0	1	1	0	Setzen
1	0	0	1	Zurücksetzen
1	1	1	1	Toggeln



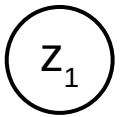
Automaten

- Verknüpfung von Zuständen, Eingabe & Ausgabe
- Meist mit binärer Ein-/Ausgabe
- Beispiel: Ampel



Automaten

- Zustand
  - Kreis mit Name des Zustands

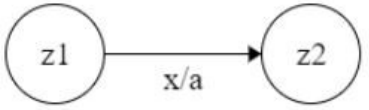


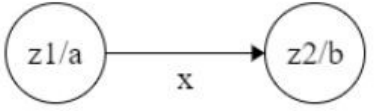
- Übergang
  - Pfeil mit zwischen zwei Zuständen genutzter Eingabe

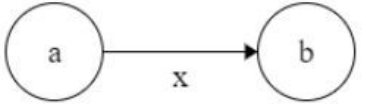
- Ausgabe
  - Zusätzlich an Zuständen oder Übergängen

Automaten

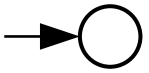
- 3 Darstellungen/Typen
  - Mealy
    - Ausgabe auf Übergängen
  - Moore
    - Ausgabe im Zustand
  - Medwedev
    - Ausgabe **entspricht** Zustand
- Zeichen: http://madebyevan.com/fsm/





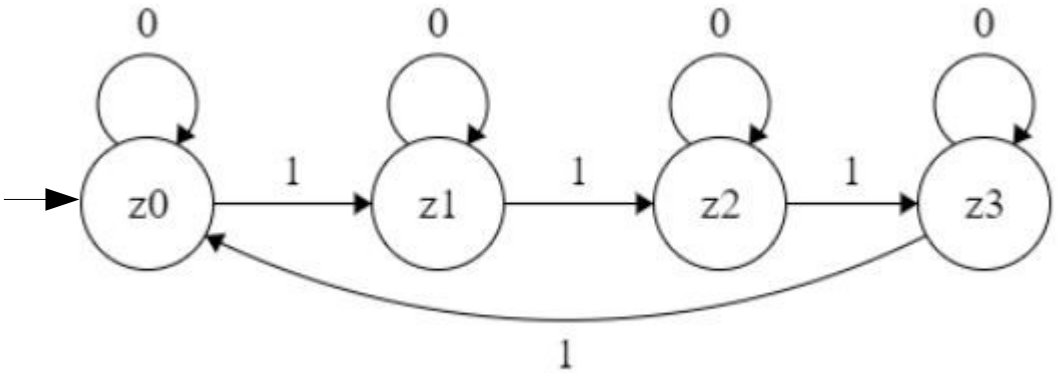


Automaten

- Startzustand
  - Pfeil ohne „Herkunft“ 
- Endzustände
  - optional, doppelter Kreis 
- Deterministisch
  - Alle Übergänge sind definiert
- Nicht-Deterministisch
  - Nicht alle Übergänge sind definiert

Automaten

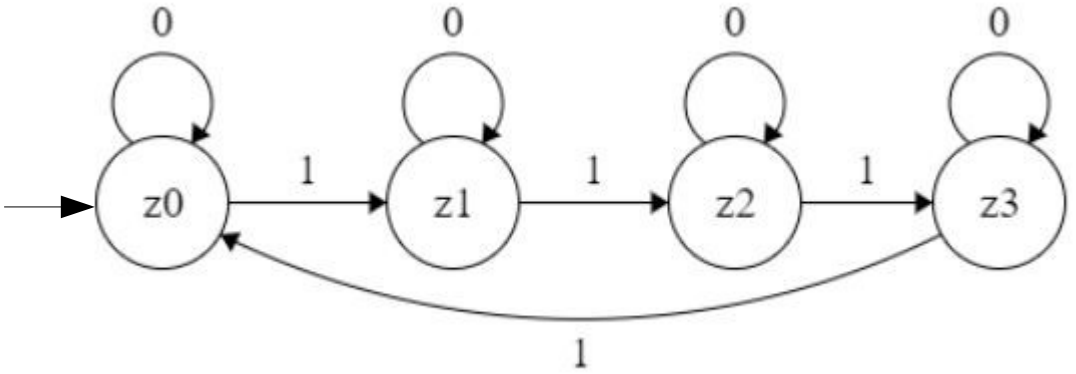
- Formalisierung? Ein-/Ausgabe?



Die gezeichneten Pfeile sind für die Eingabe 1, die Eingabe 0 ändert den Zustand nicht. In den Übungen die 0 auch eintragen, da deterministisch!

Aufgabe:

- Stelle die Ampel als Moore, Mealy dar
- Schält bei Eingabe 1 weiter
- Bei Eingabe 0 bleibt sie konstant
- Wie lässt sich ein Medwedev realisieren?



Automaten → Schaltwerk

- Alle Zustände mit JK-FF codieren
- Zustandsübergangsdiagramm
  - Don‘t Care, falls Zustand ungenutzt
- Set/Reset/Speichern der FFs
- FF-Ansteuerungsgleichung minimieren
  - Eingabevariablen: Zustände und Eingabe

JK-Flipflops geschickt schalten

- 0 → 0: Nicht (Jump oder Toggle)
- 0 → 1: Jump oder Toggle
- 1 → 0: Kill oder Toggle
- 1 → 1: Nicht (Kill oder Toggle)

Q_n	Q_{n+1}	J	K
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

RS-Flipflops geschickt schalten

- [Verhindere zustand R=S=1]
- 0 → 0: Nicht (Set oder Toggle)
- 0 → 1: NUR Set
- 1 → 0: NUR Kill
- 1 → 1: Nicht (Reset oder Toggle)

Q_n	Q_{n+1}	S	R
0	0	0	*
0	1	1	0
1	0	0	1
1	1	*	0

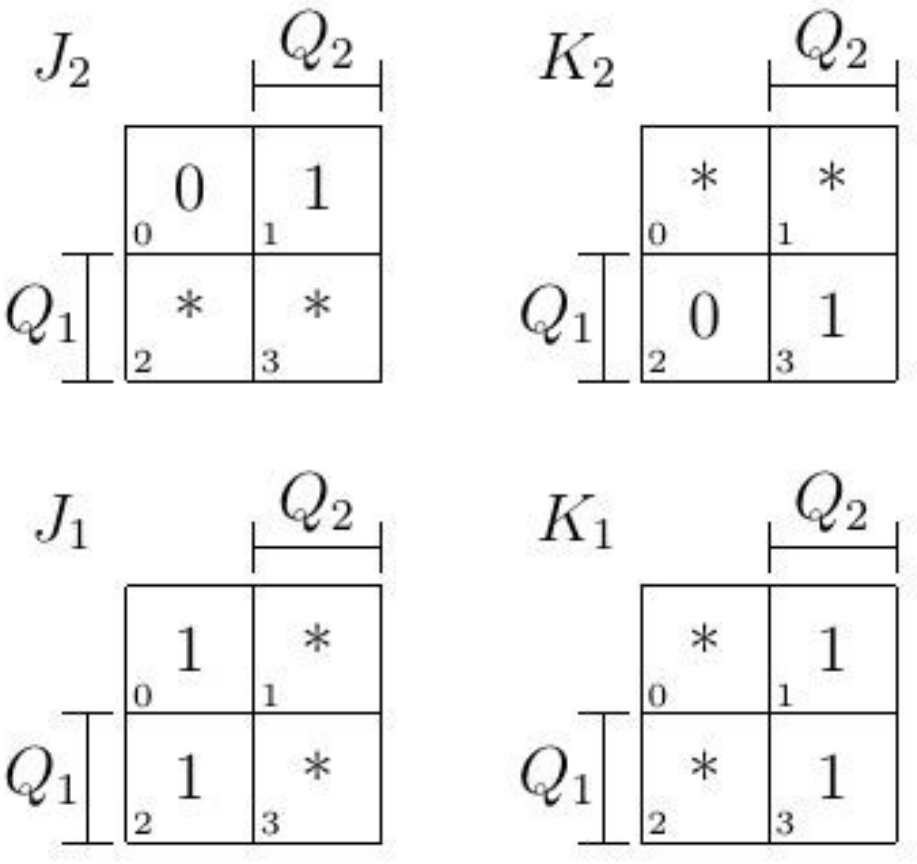
Ampelschaltung

Phase	Q_2	Q_1	Q_2‘	Q_1‘	J_2	K_2	J_1	K_1
Rot	0	0						
Gelb	0	1						
Grün	1	0						
Rot/Gelb	1	1						

Ampelschaltung

Phase	Q_2	Q_1	Q_2‘	Q_1‘	J_2	K_2	J_1	K_1
Rot	0	0	0	1	0	*	1	*
Gelb	0	1	1	0	1	*	*	1
Grün	1	0	1	1	*	0	1	*
Rot/Gelb	1	1	0	0	*	1	*	1

Ansteuerungsgleichung



Schaltwerk → Automaten-Typ

- Keine Ausgangsfkt.?
  - Medwedev
- Ausgangsfkt. nutzen Eingabe?
  - Mealy
- Ausgangsfkt. der FF werden kodiert?
  - Moore

Jetzt Doch: Negative Binärzahlen

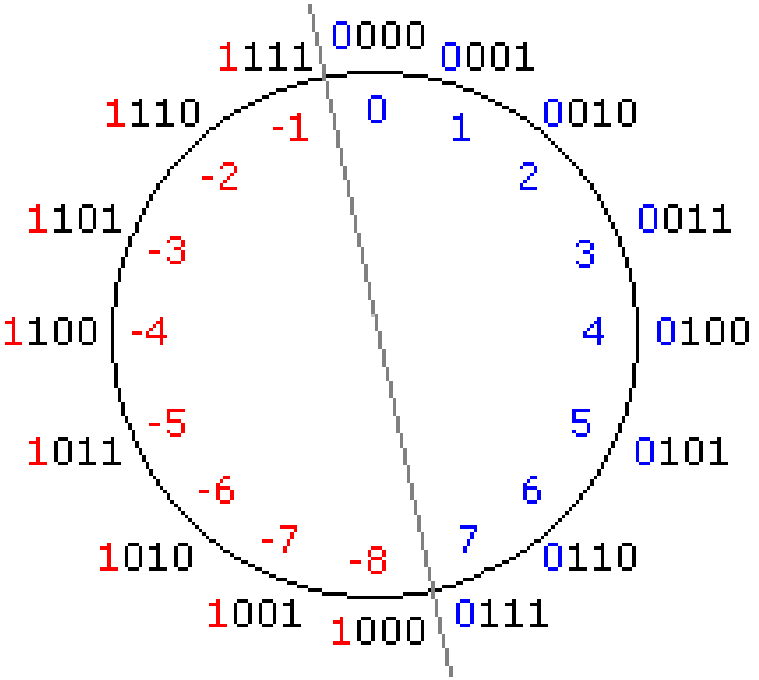
- Idee 1: Vorzeichenbehaftet
  - Höchstes Bit *s* gibt Vorzeichen an
  - 0 = positiv; 1 = negativ bzw. $(-1)^s$
  - Problem: Berechnung?

Negative Binärzahlen

- Invertieren: $0 \leftrightarrow 1$
- Idee 2: 1er-Komplement
  - Invertiere Zahl zum Negieren
  - Wieder: Höchstes Bit *s* für Vorzeichen
  - Wertebereich: $[-2^{n-1}; 2^{n-1}]$
  - Problem: Doppelte 0
    - $0000_{1C} = 0_d$; $1111_{1C} = -0_d$

2er-Komplement

- Idee: 2er-Komplement
  - Subtrahiere im Negativen -1, um -0 zu verhindern
  - Wertebereich: $[-2^{n-1} - 1; 2^{n-1}]$
- Negieren (in beide Richtungen)
  - Invertieren
  - 1 darauf addieren



Wieso 2er-Komplement?

- Rechnen mit negativen Zahlen
- Subtrahieren = Addieren einer negativen 2er-Komplement-Zahl
- Testen: Rechne 3 – 7 = -4 bzw. -3 – 2 = -5

```
  0011
+ 1001
```

```
  1101
+ 1110
```