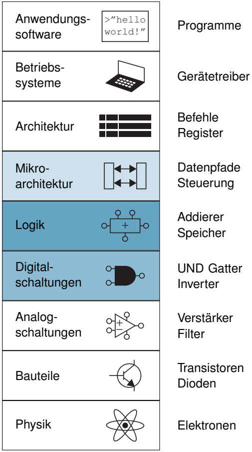
Digitaltechnik
Wintersemester 2021/2022
9. Vorlesung

TECHNISCHE UNIVERSITÄT DARMSTADT

Umfrage zur letzten Woche

Inhalt

1. FSM: Konzept, Notationen und Anwendungsbeispiele
2. Moore vs. Mealy Automaten
3. Zerlegen von Zustandsautomaten
4. Zusammenfassung



Überblick der heutigen Vorlesung

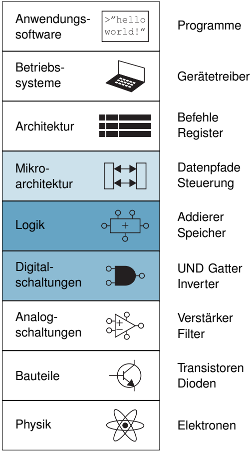
- Endliche Zustandsautomaten (Finite State Machines)
  - Konzept, Notationen und Anwendungsbeispiele
  - Moore vs. Mealy Automaten
  - Zerlegen von Zustandsautomaten

Harris 2013/2016
Kap. 3.4

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

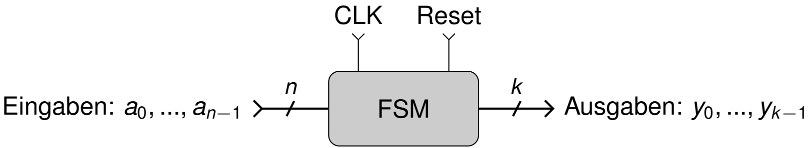
1. FSM: Konzept, Notationen und Anwendungsbeispiele
2. Moore vs. Mealy Automaten
3. Zerlegen von Zustandsautomaten
4. Zusammenfassung



Endliche Zustandsautomaten
Finite State Machines (FSM) LQ10-1 RQ10-1

TECHNISCHE UNIVERSITÄT DARMSTADT

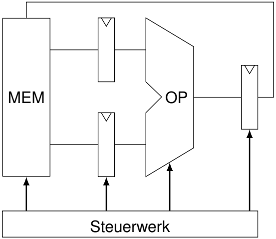
- synchrone sequentielle Schaltungen mit
  - *n* Eingabebits
  - *k* Ausgabebits
  - *ein* interner Zustand (besteht aus $m \geq 1$ Bits)
  - Takt und Reset
- in jedem Takt (zur steigenden Flanke)
  - Reset aktiv $\rightarrow$ Zustand = Startzustand
  - Reset inaktiv $\rightarrow$ neuen Zustand und Ausgaben aus aktuellem Zustand und Eingaben berechnen



FSM Anwendungsbeispiele

- Zahlenschloss (bspw. an Tresor)
  - Eingaben: Taste i gedrückt
  - Ausgaben: Schloss öffnen, Fehlermeldung anzeigen
  - Zusammenhang zwischen Zuständen:
    nur Öffnen, wenn letzte (4) Eingaben korrekt
    und in richtiger Reihenfolge

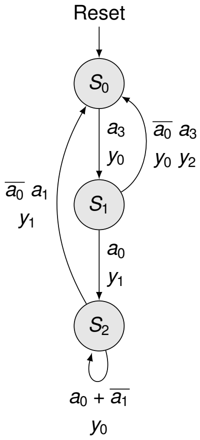
- Steuerwerk von Rechnern (Mikroarchitektur)
  - Eingaben: Bits des aktuellen Instruktionswortes
  - Ausgaben: Steuersignale für
    - Arithmetik (welche Operation)
    - Speicher (welche Operanden)
  - Zusammenhang zwischen Zuständen:
    in Pipeline-Stufen hängen Steuersignale von
    anderen Instruktionen ab



- vieles mehr (sehr häufig verwendetes Konzept)

FSM Diagramme als gerichtete Graphen

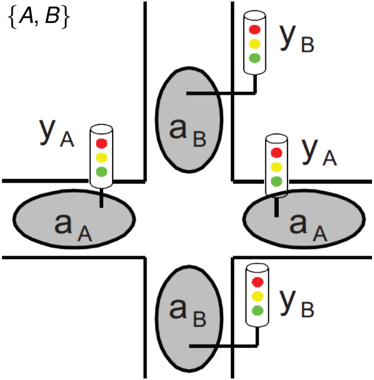
- Zustände (States) als Knoten
  - symbolische Namen (typ. $S_0, S_1, \ldots$)
  - binäre Zustandskodierung ist unabhängiges Problem
- Zustandsübergänge (Transitions) als Kanten
  - als boole'scher Ausdruck (leere Bedingung entspricht 1)
  - Keine zwei Kantenbedingungen gleichzeitig erfüllbar
  - Vereinfachte Notation (keine Selbstschleifen): Zustand bleibt unverändert, wenn keine Bedingung erfüllt
- genau eine Kante ohne Startpunkt für Reset
- Ausgaben
  - an Kanten (Mealy-Automat)
  - oder in Zuständen (Moore-Automat)
  - als vollständiger boole'scher Ausdruck (Minterm)
  - oder nur gesetzte Ausgaben



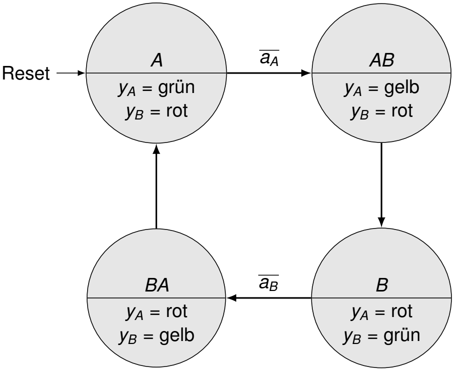
FSM Beispiel für Ampelsteuerung

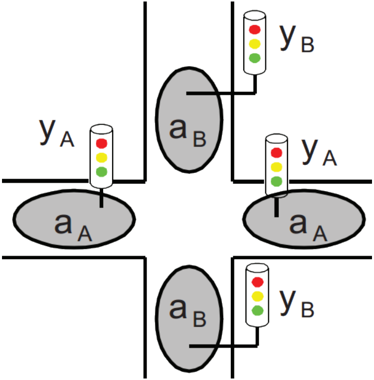
- Eingänge:
  - $a_k = 1 \Leftarrow$ Induktionsschleife k erkennt Fahrzeug für $k \in \{A, B\}$
- Ausgänge
  - $y_k \in \{\text{rot,grün,gelb}\} \Rightarrow$ Ampelphase für $k \in \{A, B\}$

$\Rightarrow$ FSM für Bedarfssteuerung
  - halte Spur grün, solange auf dieser Fahrzeuge erkannt werden
  - ansonsten schalte aktuelle Fahrbahn über gelb nach rot und andere Fahrbahn auf grün

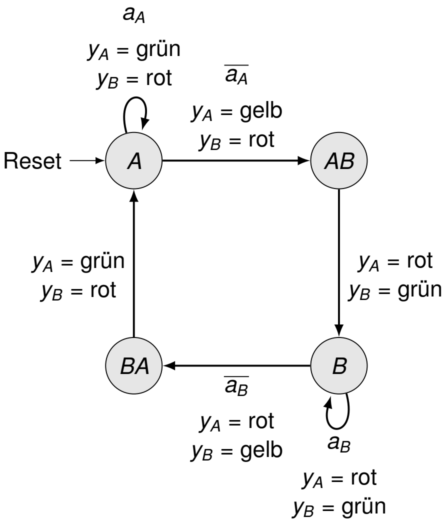


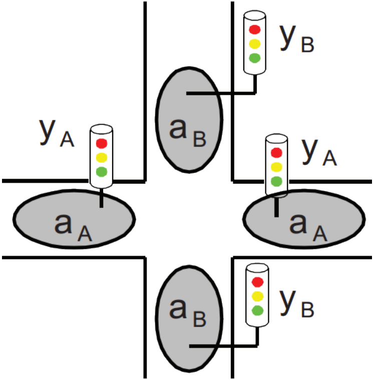
Moore-Automat für Ampelsteuerung





Mealy-Automat für Ampelsteuerung





Verhalten: Zustandsübergang und Ausgabe

- Ziel: Realisieren in HW
- Vorgehen: Beschreibung $\rightarrow$ Gleichung $\rightarrow$ Schaltung
  - Beschreibung: textuell, FSM, Wahrheitstabelle, Karnaugh Diagramm
  - Gleichung: abgeleitet von der Beschreibung
  - Schaltung: kombinatorisch, sequentiell, oder synchron

Zustandsübergangs- und Ausgabetabelle

LQ10-3 **RQ10-3**

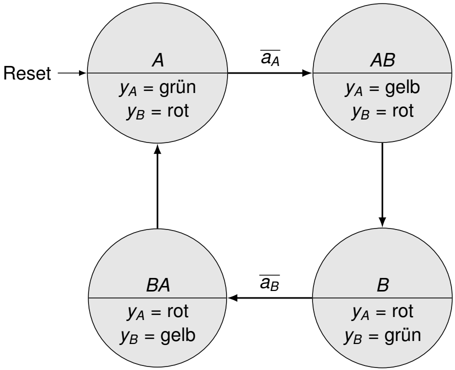
- maschinenlesbare Darstellung
- kann noch mit abstrakten Zuständen und Ausgaben arbeiten
- kann Don’t Cares verwenden
- Kurzschreibweise
  - aktueller Zustand S
  - nächster Zustand S'
- *Achtung:* implizite Bedingungen (bspw. Selbstschleifen) beim Ableiten aus Diagrammen beachten

Zustandsübergangs- und Ausgabetabelle für Moore-Automat der Ampelsteuerung

TECHNISCHE UNIVERSITÄT DARMSTADT

S	a_A	a_B	S'
A	1	*	A
A	0	*	AB
AB	*	*	B
B	*	1	B
B	*	0	BA
BA	*	*	A

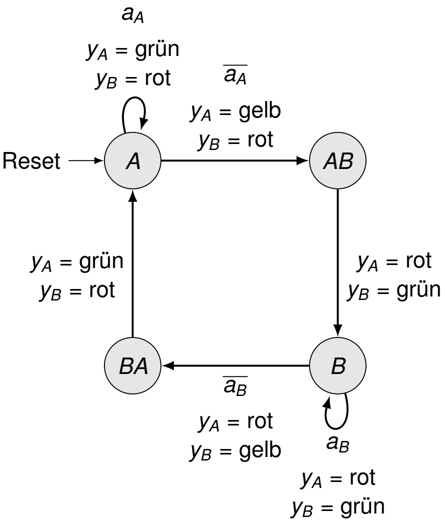
S	y_A	y_B
A	grün	rot
AB	gelb	rot
B	rot	grün
BA	rot	gelb



Zustandsübergangs- und Ausgabetabelle für Mealy-Automat der Ampelsteuerung

S	a_A	a_B	S'
A	1	*	A
A	0	*	AB
AB	*	*	B
B	*	1	B
B	*	0	BA
BA	*	*	A

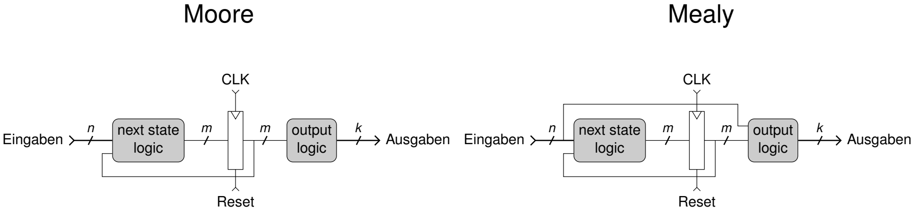
S	a_A	a_B	y_A	y_B
A	1	*	grün	rot
A	0	*	gelb	rot
AB	*	*	rot	grün
B	*	1	rot	grün
B	*	0	rot	gelb
BA	*	*	grün	rot



FSM als synchrone sequentielle Schaltung

- Zustandsregister
  - speichert aktuellen Zustand S
  - übernimmt nächsten Zustand S' bei steigender Taktflanke
- kombinatorische Logik realisiert
  - Zustandsübergangstabelle ("next state logic")
  - Ausgabetabelle ("output logic")

$\Rightarrow$ binäre Kodierung der Zustände und Ein-/Ausgaben notwendig



Zustandskodierung cs : $S \rightarrow \mathbb{B}^m$

- weist jedem Zustand einen *m* Bit breiten Wert zu
- kann i.d.R. frei gewählt werden (da nach außen nicht sichtbar)
- bspw. “Durchnummerieren”: $\mathrm{cs}(S_k) = (s_{m-1}...s_0)$ mit $\mathrm{u}_{2,m}(s_{m-1}...s_0) = k$
- manchmal führen aber andere Kodierungen zu effizienterer kombinatorischer Logik, auch wenn mehr Zustandsbits benötigt werden
  - One-Hot-Kodierung
  - bestehende Ausgabekodierung
    (wenn jeder Zustand eine spezifische Ausgabe verursacht)
- Kodierung der Ein-/Ausgänge ist i.d.R. von der Anwendung vorgegeben
  - kann ansonsten für jede Ein/Ausgabe spezifisch gewählt werden

Kodierte Tabellen für Moore-Automat der Ampelsteuerung

S	s_1	s_0
A	0	0
AB	0	1
B	1	0
BA	1	1

y_A	y_3	y_2
grün	0	0
gelb	0	1
rot	1	0

y_B	y_1	y_0
grün	0	0
gelb	0	1
rot	1	0

S	s_1	s_0	a_A	a_B	s'_1	s'_0
A	0	0	1	*	0	0
A	0	0	0	*	0	1
AB	0	1	*	*	1	0
B	1	0	*	1	1	0
B	1	0	*	0	1	1
BA	1	1	*	*	0	0

S	s_1	s_0	y_3	y_2	y_1	y_0
A	0	0	0	0	1	0
AB	0	1	0	1	1	0
B	1	0	1	0	0	0
BA	1	1	1	0	0	1

- $n = 2$ Eingangssignale a_i; $m = 2$ Zustandsbits s_i; $k = 4$ Ausgabesignale y_i

$\Rightarrow$ sechs boole'sche Funktionen (für s'_i, y_i) aus Wahrheitswertetabellen ableiten

Minimierte kombinatorische Logik für Moore-Automat der Ampelsteuerung

ampel/state.esp

```
.i 4
.o 2
001- 00
000- 01
01-- 10
10-1 10
10-0 11
11-- 00
```

espresso ampel/state.esp

```
.i 4
.o 2
.p 4
10-0 01
000- 01
01-- 10
10-- 10
.e
```

ampel/output.esp

```
.i 2
.o 4
00 0010
01 0110
10 1000
11 1001
```

espresso ampel/output.esp

```
.i 2
.o 4
.p 4
01 0100
11 0001
1- 1000
0- 0010
.e
```

$$s'_1 = s_1 \oplus s_0$$

$$s'_0 = s_1\ \overline{s_0}\ \overline{a_B} + \overline{s_1}\ \overline{s_0}\ \overline{a_A}$$

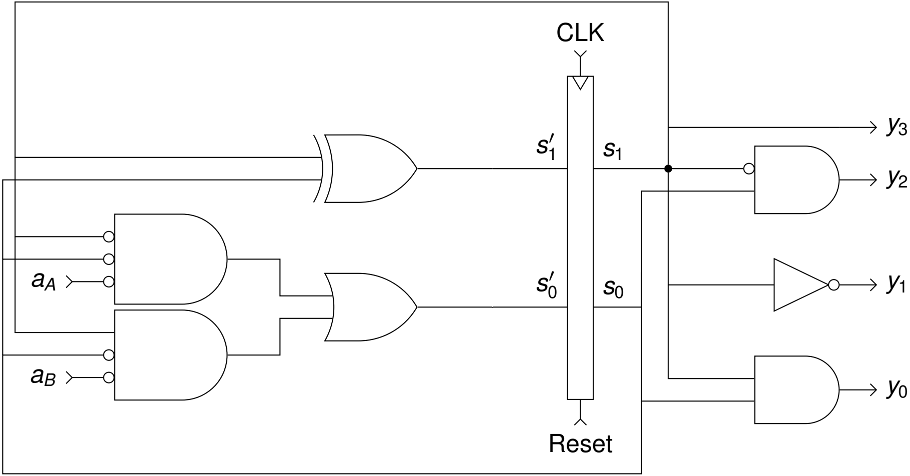
$$y_3 = s_1$$

$$y_2 = \overline{s_1}\ s_0$$

$$y_1 = \overline{s_1}$$

$$y_0 = s_1\ s_0$$

Schaltplan
für Moore-Automaten der Ampelsteuerung



Zusammenfassung FSM Entwurfsverfahren

- definiere Ein- und Ausgänge
- wähle zwischen Moore- und Mealy-Automat
- zeichne Zustandsdiagramm
- kodiere Zustände (und ggf. Ein-/Ausgänge)
- stelle Zustandsübergangstabelle und Ausgabetabelle auf
- stelle boole'sche Gleichungen für Zustandsübergangs- und Ausgangslogik unter Ausnutzung von Don't Cares auf
- entwerfe Schaltplan: Gatter + Register

Pause & Umfrage bis hier

Agenda

1. FSM: Konzept, Notationen und Anwendungsbeispiele

2. Moore vs. Mealy Automaten

3. Zerlegen von Zustandsautomaten

4. Zusammenfassung

Anwendungs-software (>"hello world!")	Programme
Betriebs-systeme	Gerätetreiber
Architektur	Befehle Register
Mikro-architektur	Datenpfade Steuerung
Logik	Addierer Speicher
Digital-schaltungen	UND Gatter Inverter
Analog-schaltungen	Verstärker Filter
Bauteile	Transistoren Dioden
Physik	Elektronen

Mealy vs. Moore LQ10-5

- für Ampelsteuerung war Moore-Automat effizienter
- das ist aber nicht immer so

$\Rightarrow$ muss von Fall zu Fall neu bewertet werden

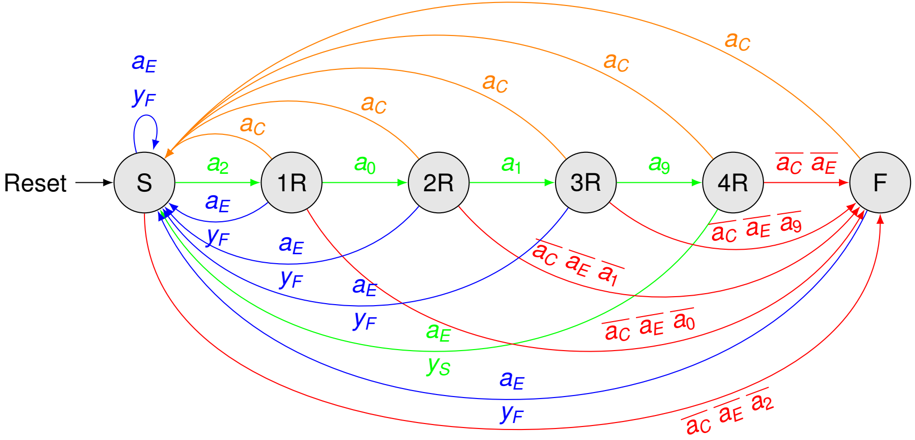
- in der Regel
  - Moore besser, wenn Ausgaben statisch
  - Mealy besser, wenn Ausgaben kurzfristige Aktionen auslösen
  - Mealy reagiert schneller auf Änderungen der Eingabe
- Verdeutlichung durch weitere Beispiele

FSM Beispiel für Zahlenschloss

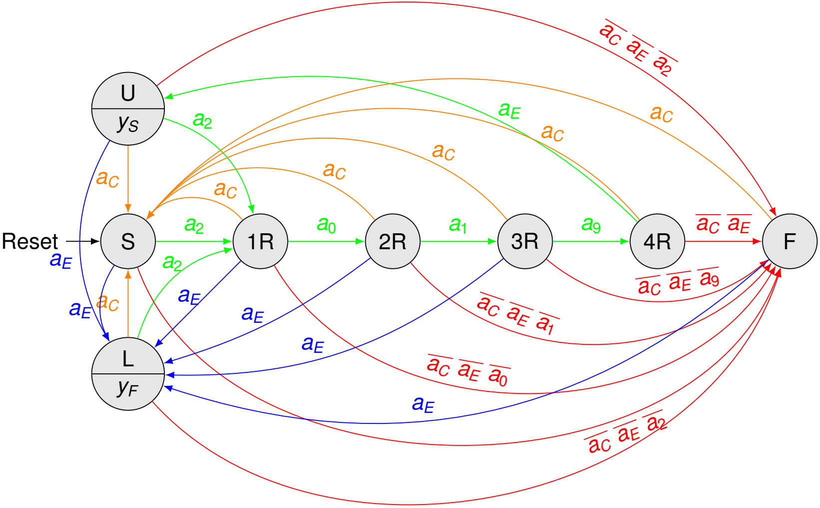
TECHNISCHE UNIVERSITÄT DARMSTADT

- Eingänge:
  - $a_k = 1 \Leftarrow$ Taste k gedrückt für $0 \leq k \leq 9$
  - $a_\mathrm{C} = 1 \Leftarrow$ Taste “Cancel” gedrückt
  - $a_\mathrm{E} = 1 \Leftarrow$ Taste “Enter” gedrückt
- Ausgänge
  - $y_S = 1 \Rightarrow$ Schloss entriegeln
  - $y_F = 1 \Rightarrow$ Fehlermeldung anzeigen
- Vereinfachungen
  - Zustandsübergang nur dann, wenn überhaupt eine Taste gedrückt
  - immer nur eine Taste gleichzeitig aktivierbar
- Passwort: 2019
- *Achtung:* Fehlermeldung nicht direkt bei erster falscher Ziffer zeigen

Mealy-Automat für Zahlenschloss



Moore-Automat für Zahlenschloss



Mealy vs. Moore für Zahlenschloss

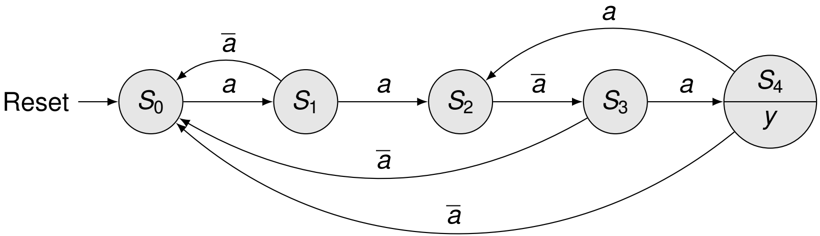
- Moore-Automat braucht zwei zusätzliche Zustände (U,L), um die beiden unterschiedlichen Übergänge zurück in den Ausgangszustand (nach richtiger oder falscher Eingabe) voneinander zu unterscheiden
- Ausgaben beschreiben eher
  - Aktionen (Schloss öffnen, Fehler anzeigen) als
  - Zustände (Schloss ist geöffnet, Fehler wird angezeigt)

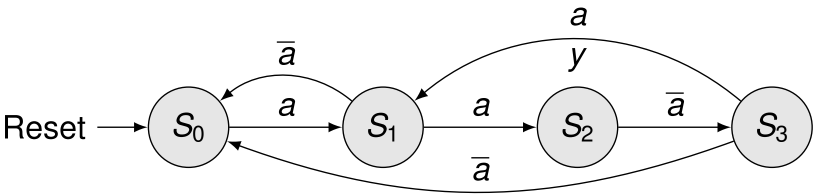
$\Rightarrow$ Mealy-Automat hier besser geeignet

Weiteres Beispiel: Mustererkennung LQ10-4 RQ10-4

- typisch in Bild- und Textanalyse (bspw. Suche nach regulären Ausdrücken)
- bspw.: Erkenne Bitfolge “1101” in Bitsequenz
- Eingänge: das nächste Bit $a \in \mathbb{B}$
- Ausgabe: $y = 1 \Rightarrow$ gesuchte Bitfolge erkannt

Moore- und Mealy-Automat für 1101 Mustererkennung





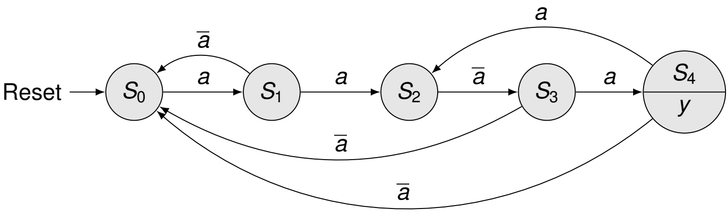
Moore-Automat für 1101 Mustererkennung: Zustandübergangs- und Ausgabetabellen

TECHNISCHE UNIVERSITÄT DARMSTADT

S	a	S'
S_0	0	S_0
S_0	1	S_1
S_1	0	S_0
S_1	1	S_2
S_2	0	S_3
S_2	1	S_2
S_3	0	S_0
S_3	1	S_4
S_4	0	S_0
S_4	1	S_2

S	s_2	s_1	s_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0

S	y
S_0	0
S_1	0
S_2	0
S_3	0
S_4	1



Moore-Automat für 1101 Mustererkennung: Logikgenerierung mit vielen Don’t Cares

TECHNISCHE UNIVERSITÄT DARMSTADT

`pattern/moore/state.esp`

```
.i 4
.o 3
0000 000
0001 001
0010 000
0011 010
0100 011
0101 010
0110 000
0111 100
1000 000
1001 010
1010 ---
1011 ---
1100 ---
1101 ---
1110 ---
1111 ---
```

`espresso pattern/moore/state.esp`

```
.i 4
.o 3
.p 6
0001 001
-100 001
-111 100
-011 010
1--1 010
-10- 010
.e
```

$$s_2' = s_1\ s_0\ a$$

$$s_1' = \overline{s_1}\ s_0\ a + s_2\ a + s_1\ \overline{s_0}$$

$$s_0' = \overline{s_2}\ \overline{s_1}\ \overline{s_0}\ a + s_1\ \overline{s_0}\ \overline{a}$$

`pattern/moore/output.esp`

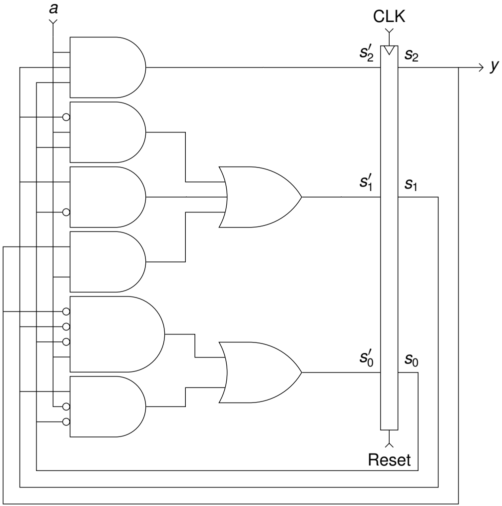
```
.i 3
.o 1
000 0
001 0
010 0
011 0
100 1
101 -
110 -
111 -
```

`espresso pattern/moore/output.esp`

```
.i 3
.o 1
.p 1
1-- 1
.e
```

$$y = s_2$$

Moore-Automat für 1101 Mustererkennung: Schaltwerk



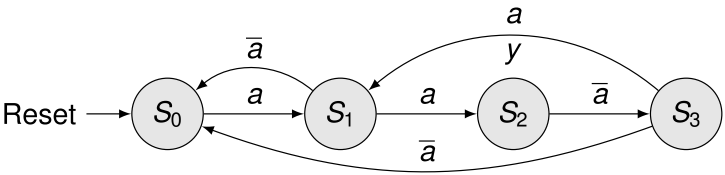
Mealy-Automat für 1101 Mustererkennung: Zustandübergangs- und Ausgabetabellen RQ10-5

TECHNISCHE UNIVERSITÄT DARMSTADT

S	a	S'
S_0	0	S_0
S_0	1	S_1
S_1	0	S_0
S_1	1	S_2
S_2	0	S_3
S_2	1	S_2
S_3	0	S_0
S_3	1	S_1

S	s_1	s_0
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

S	a	y
S_0	0	0
S_0	1	0
S_1	0	0
S_1	1	0
S_2	0	0
S_2	1	0
S_3	0	0
S_3	1	1



Mealy-Automat für 1101 Mustererkennung: Logikgenerierung ohne Don’t Cares

pattern/mealy/state.esp

```
.i 3
.o 2
000 00
001 01
010 00
011 10
100 11
101 10
110 00
111 01
```

espresso pattern/mealy/state.esp

```
.i 3
.o 2
.p 5
011 10
100 01
001 01
111 01
10- 10
.e
```

$$s_1' = \overline{s_1}\ s_0\ a + s_1\ \overline{s_0}$$

$$s_0' = s_1\ \overline{s_0}\ \overline{a} + \overline{s_1}\ \overline{s_0}\ a + s_1\ s_0\ a$$

pattern/mealy/output.esp

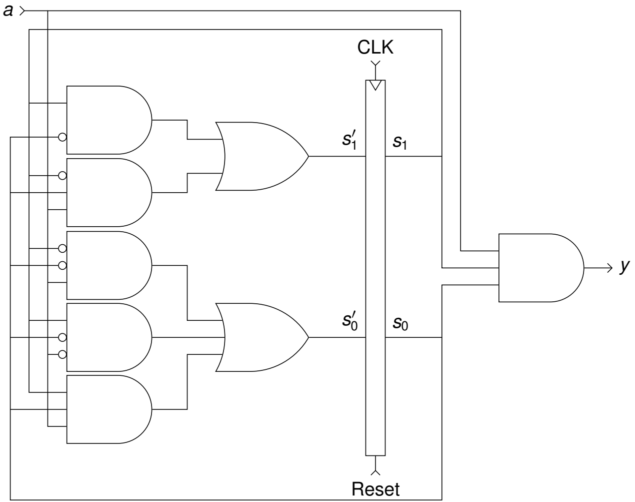
```
.i 3
.o 1
000 0
001 0
010 0
011 0
100 0
101 0
110 0
111 1
```

espresso pattern/mealy/output.esp

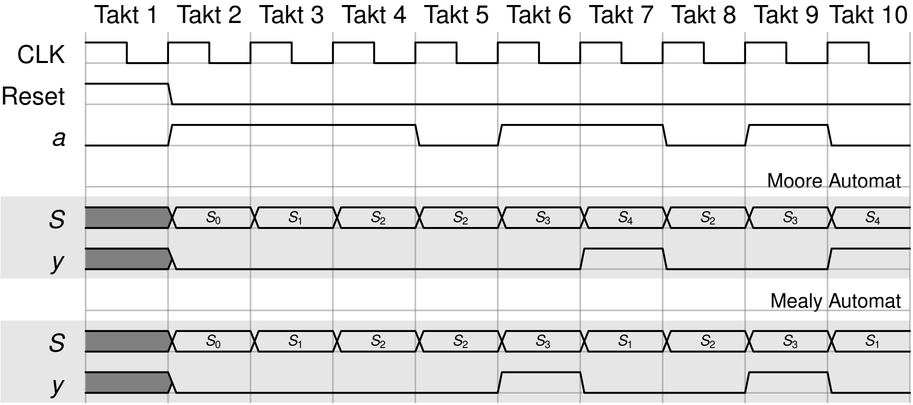
```
.i 3
.o 1
.p 1
111 1
.e
```

$$y = s_1\ s_0\ a$$

Mealy-Automat für 1101 Mustererkennung: Schaltwerk



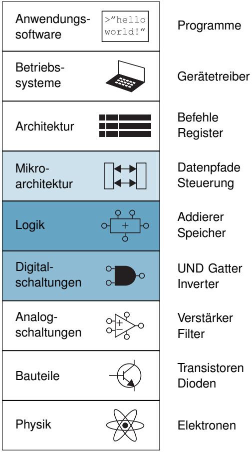
Moore- und Mealy-Automaten: Zeitverhalten



- Mealy-Automat erkennt Muster einen Takt früher

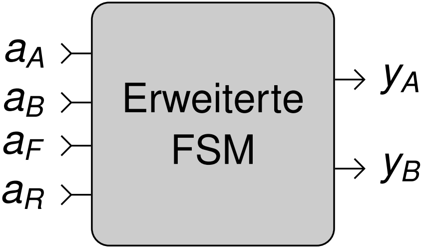
Agenda

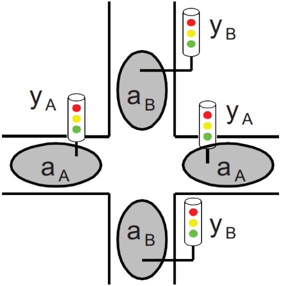
1. FSM: Konzept, Notationen und Anwendungsbeispiele
2. Moore vs. Mealy Automaten
3. Zerlegen von Zustandsautomaten
4. Zusammenfassung



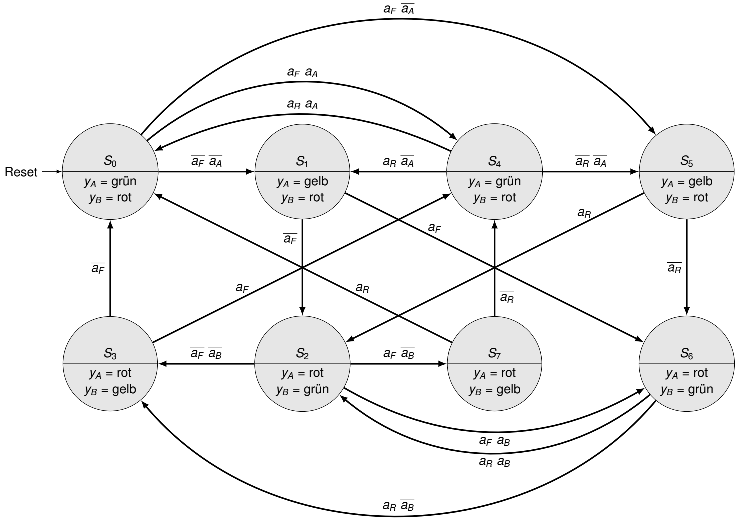
Zerlegen von Zustandsautomaten (FSM Dekomposition)

- Aufteilen komplexer FSMs in einfachere interagierende FSMs
- Beispiel: Ampelsteuerung mit Modus für Festumzüge (Ampel B bleibt permanent grün)
  - FSM bekommt zwei weitere Eingänge: a_F, a_R
  - $a_F = 1 \Rightarrow$ aktiviert Festumzugsmodus
  - $a_R = 1 \Rightarrow$ deaktiviert Festumzugsmodus

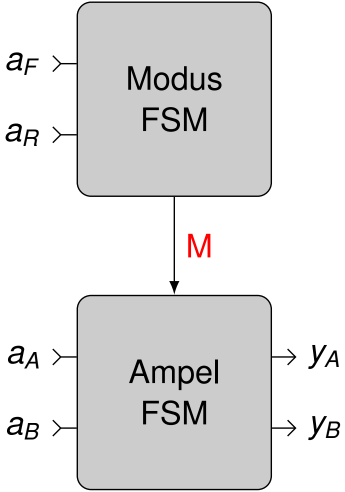


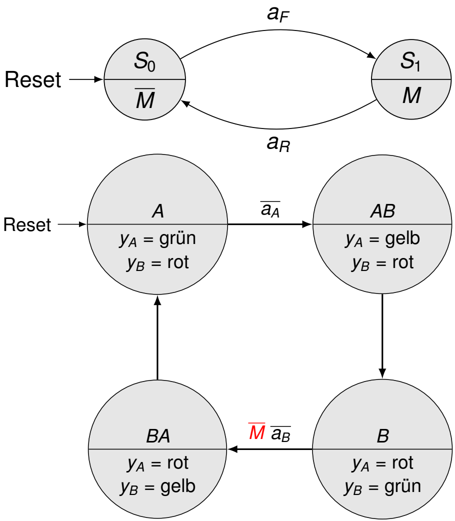


Unzerlegte FSM



Zerlegung in kommunizierende FSMs





Agenda

1. FSM: Konzept, Notationen und Anwendungsbeispiele
2. Moore vs. Mealy Automaten
3. Zerlegen von Zustandsautomaten
4. Zusammenfassung

Anwendungs-software	>"hello world!"	Programme
Betriebs-systeme		Gerätetreiber
Architektur		Befehle Register
Mikro-architektur		Datenpfade Steuerung
Logik	+	Addierer Speicher
Digital-schaltungen		UND Gatter Inverter
Analog-schaltungen	+ -	Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

Zusammenfassung und Ausblick LQ10-7

TECHNISCHE UNIVERSITÄT DARMSTADT

- Endliche Zustandsautomaten (FSMs)
  - Konzept, Notationen und Anwendungen
  - Moore vs. Mealy
  - Zerlegen von Zustandsautomaten

- Nächste Vorlesung behandelt
  - Hardwarebeschreibung mit SystemVerilog