

Digitaltechnik

Wintersemester 2017/2018

8. Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT





1. Einleitung
2. Konzept, Notationen und Anwendungsbeispiele
3. Mealy vs. Moore
4. Zerlegen von Zustandsautomaten
5. Zusammenfassung

Einleitung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

0111101010110001100110010111011110110001
0000101001100000001010000000100011111000
1001111101101001101100010100101011110101
1101001110001000100101001001111101100110
0011011110100000100100111010101000110000
10100011101100001110000000001111110100100
1111110101010011101000010110101001011101
000011001001010111000110111110010000100
10100110101100010100000011000000001011110
01011001110010001100000011100110111110101
00110111011111111010111010100011101101010
10001100110000001001010101100110000000001
0111000100111000100100101100101010111100
0010010101111100110001101110100001100010
1011110111101101110111001010000110110000
0111101010011000110110001001010000010000

Überblick der heutigen Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Endliche Zustandsautomaten
 - ▶ Konzept, Notationen und Anwendungsbeispiele
 - ▶ Moore vs. Mealy
 - ▶ Zerlegen von Zustandsautomaten



Harris 2013
Kap. 3.4
Seite 117 - 133

Konzept, Notationen und Anwendungsbeispiele



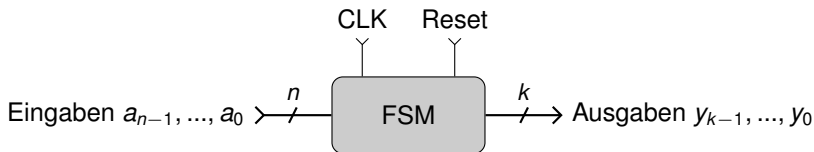
TECHNISCHE
UNIVERSITÄT
DARMSTADT

1000101111001101001011101111011010010010
0101000100100110100101100111011001010111
0111001101110011000010011100010011111011
0111000011110111000100111110010111100001
0101011001010111101000001011000011111111
0000011101110010000010001001001100000100
0110101100010110110000100010011001100111
0111001010001000010000100111011111101010110
0100100001101000100110100110011001010101
0010010010010010011101001000111001011101
1101010000001111011010010000111101000001
1001100001110110001111100001001001111111
1010101100101111110110001110111100010111
1101101100011000011010011111001101110100
01100111111100110001100101001010011001011
0100100111101001011001001110101111110101

Endliche Zustandsautomaten

Finite State Machines (FSM)

- ▶ 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

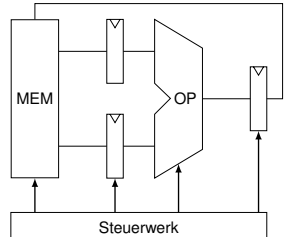


FSM Anwendungsbeispiele



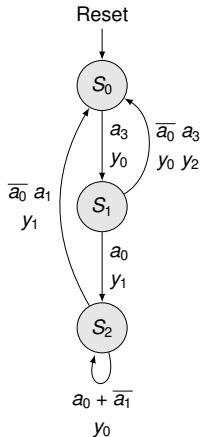
TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ 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:
bspw. in Pipeline-Stufen
- ▶ vieles mehr (sehr häufig verwendetes Konzept)



FSM Diagramme als gerichtete Graphen

- ▶ Zustände als Knoten
 - ▶ symbolische Namen (typ. S_0, S_1, \dots)
 - ▶ binäre Zustandskodierung ist unabhängiges Problem
- ▶ Zustandsübergang als Kante
 - ▶ als boole'scher Ausdruck (leere Bedingung entspricht 1)
 - ▶ Disjunktion aller ausgehenden Kanten ergibt 1
 - ▶ Keine zwei Kantenbedingung 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



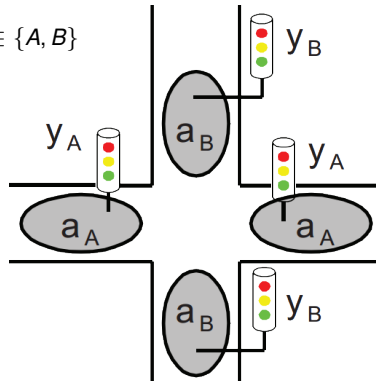
FSM Beispiel für Ampelsteuerung

- ▶ Eingänge:
 - ▶ $a_k = 1 \Leftrightarrow$ Induktionsschleife k erkennt Fahrzeug für $k \in \{A, B\}$

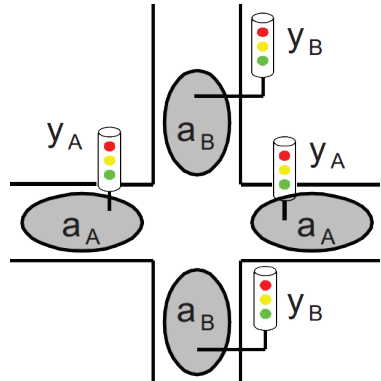
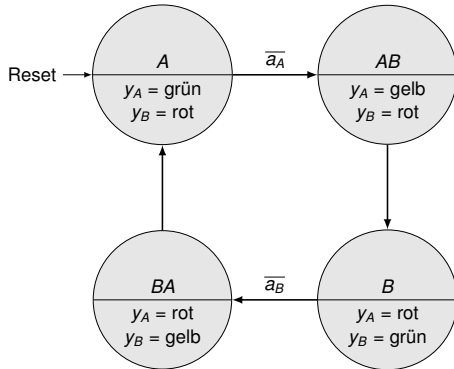
- ▶ Ausgänge
 - ▶ $y_k \in \{\text{rot}, \text{grün}, \text{gelb}\} \Rightarrow$ Ampelphase für $k \in \{A, B\}$

\Rightarrow FSM für Bedarfssteuerung

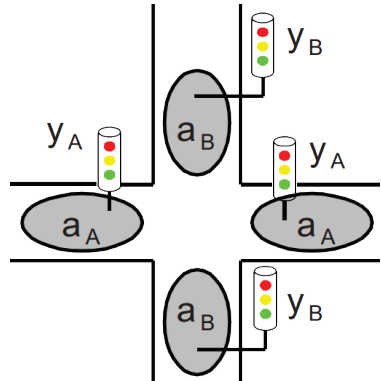
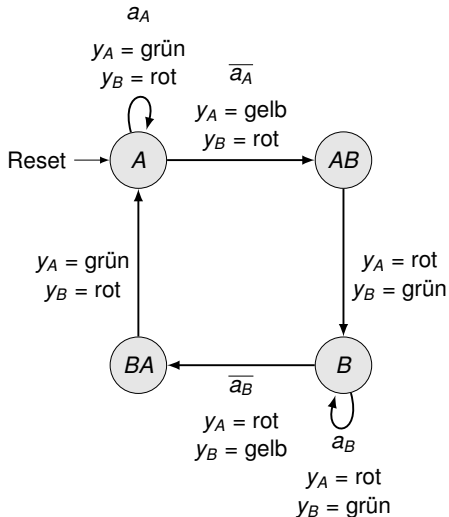
- ▶ halte Spur grün, solange auf dieser Fahrzeuge erkannt werden
- ▶ ansonsten schalte aktuelle Fahrbahn über gelb nach rot und andere Fahrbahn auf grün



Moore-Automat für Ampelsteuerung



Mealy-Automat für Ampelsteuerung





- ▶ kompaktere (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. Selsbstschleifen) beim Ableiten aus Diagrammen beachten

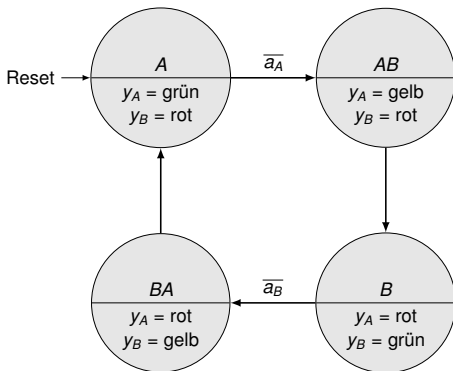
Zustandsübergangs- und Ausgabetabelle für Moore-Automat der Ampelsteuerung



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



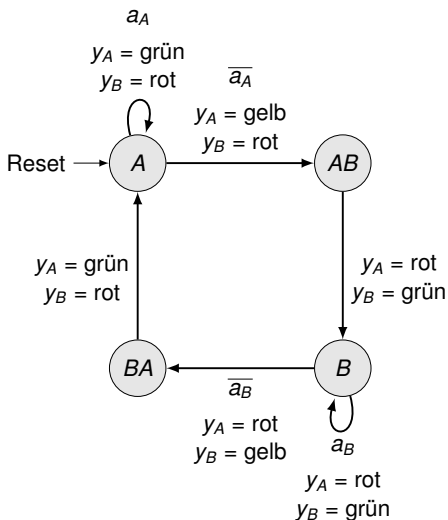
Zustandsübergangs- und Ausgabetabelle für Mealy-Automat der Ampelsteuerung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

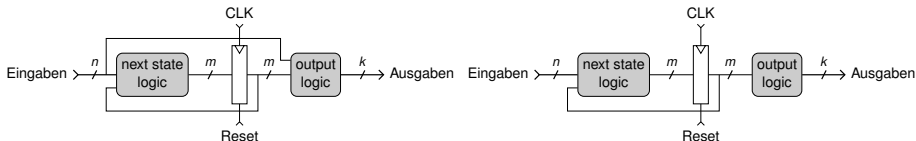
S	a_A	a_B	S'
A	1	*	A
A	0	*	AB
AB	*	*	B
B	*	1	B
B	*	0	BA
BA	*	*	A

S	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



- ▶ Zustandsregister
 - ▶ speichert aktuellen Zustand
 - ▶ übernimmt nächsten Zustand bei Taktflanke
- ▶ kombinatorische Logik realisiert
 - ▶ Zustandübergangstabelle („next state logic“)
 - ▶ Ausgangstabelle („output logic“)

⇒ binäre Kodierung der Zustände und Ein-/Ausgaben notwendig



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



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ weist jedem Zustand einen m Bit breiten Wert zu
- ▶ kann idR. frei gewählt werden (da nach außen nicht sichtbar)
- ▶ bspw. „Durchnummerieren“: $cs(S_k) = (s_{m-1} \dots s_0)$ mit $u_{2,m}(s_{m-1} \dots s_0) = k$
- ▶ manchmal führen aber andere Kodierungen zu effizienterer kombinatorischer Logik, auch wenn mehr Zustandsbits benötigt werden
 - ▶ One-Hot
 - ▶ bestehende Ausgabekodierung
(wenn jeder Zustand eine spezifische Ausgabe verursacht)
- ▶ Kodierung der Ein-/Ausgänge ist idR. 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	y_A	y_3	y_2	y_B	y_1	y_0
A	0	0	grün	0	0	grün	0	0
AB	0	1	gelb	0	1	gelb	0	1
B	1	0	rot	1	0	rot	1	0
BA	1	1						

S	s_1	s_0	a_A	a_B	s'_1	s'_0	S	s_1	s_0	y_3	y_2	y_1	y_0
A	0	0	1	*	0	0	A	0	0	0	0	1	0
A	0	0	0	*	0	1	AB	0	1	0	1	1	0
AB	0	1	*	*	1	0	B	1	0	1	0	0	0
B	1	0	*	1	1	0	BA	1	1	1	0	0	1
B	1	0	*	0	1	1							
BA	1	1	*	*	0	0							

► $n = 2$ Eingangssignale, $m = 2$ Zustandsbits, $k = 4$ Ausgabesignale

⇒ sechs bool'sche Funktionen aus Wahrheitwertetabellen ableiten

Minimierte kombinatorische Logik für Moore-Automat der Ampelsteuerung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

ampel/state.esp

1	.i	4
2	.o	2
3	001-	00
4	000-	01
5	01--	10
6	10-1	10
7	10-0	11
8	11--	00

espresso ampel/state.esp

1	.i	4
2	.o	2
3	.p	4
4	10-0	01
5	000-	01
6	01--	10
7	10--	10
8	.e	

ampel/output.esp

1	.i	2
2	.o	4
3	00	0010
4	01	0110
5	10	1000
6	11	1001

espresso ampel/output.esp

1	.i	2
2	.o	4
3	.p	4
4	01	0100
5	11	0001
6	1-	1000
7	0-	0010
8	.e	

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

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

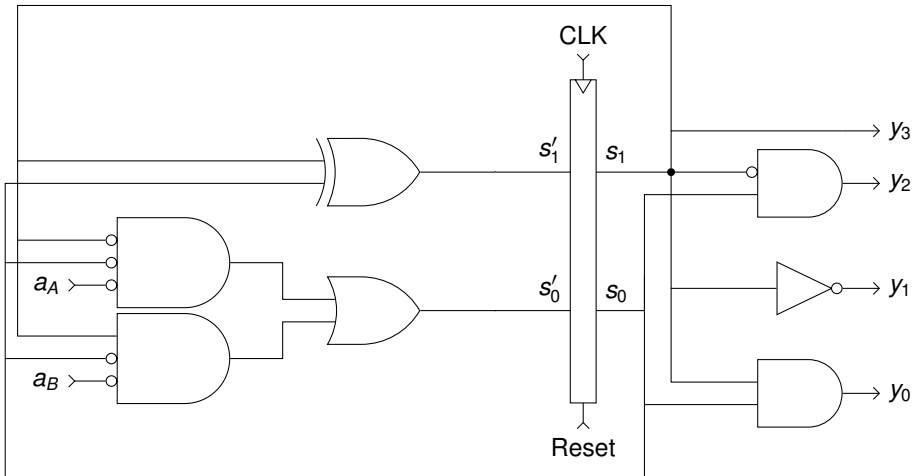
$$y_3 = s_1$$

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

$$y_1 = \overline{s_1}$$

$$y_0 = s_1 s_0$$

Schaltplan für Moore-Automaten der Ampelsteuerung





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

Mealy vs. Moore



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1101110001101000111010100010101001110101
0101111001101001010100011001110111010100
1101100100101100001111010001100101100111
1100101110101011101100000111010000010010
11101101011111111111110110110110101101001
11011110101100101001100010100010101110011
0001110010110011110101000001111001110100
0100010010010101000001111001110110110110
10101011111101110101110111110110110001110
0100101110101101011101110111100101011100
0011011000011001110110010110011111111100
0110011010101100011101110100010111000010
11100001111111010001000011000110110000011
0101100100001010001110100010011110010111
00000011110011110010001010011001100000000
100001011101001111110010111001011111110011



- ▶ für Ampelsteuerung war Moore-Automat effizienter
 - ▶ das ist aber nicht allgemein so
- ⇒ 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

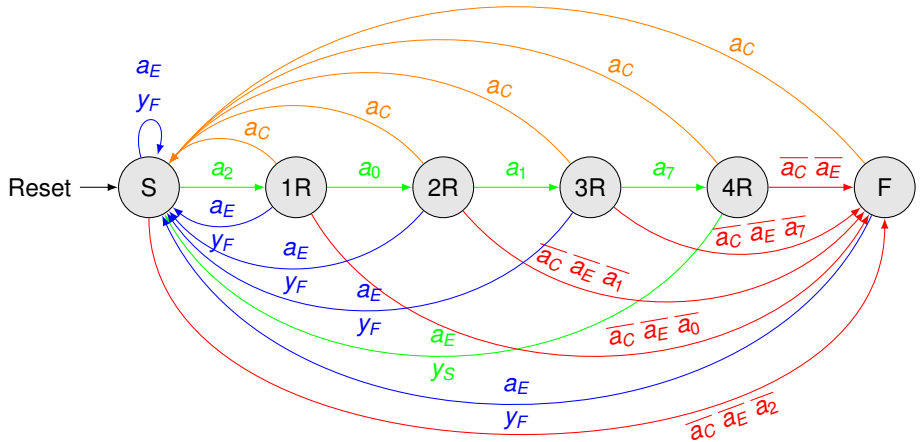
- ▶ Eingänge:
 - ▶ $a_k = 1 \Leftarrow$ Taste k gedrückt für $0 \leq k \leq 9$
 - ▶ $a_C = 1 \Leftarrow$ Taste „Cancel“ gedrückt
 - ▶ $a_E = 1 \Leftarrow$ Taste „Enter“ gedrückt
- ▶ Ausgänge
 - ▶ $y_S = 1 \Rightarrow$ Schloss entriegeln
 - ▶ $y_F = 1 \Rightarrow$ Fehlermeldung anzeigen
- ▶ Vereinfachungen
 - ▶ Zustandsübergang nur dann, wenn überhaupt eine Taste gedrückt
 - ▶ immer nur eine Taste gleichzeitig aktivierbar
- ▶ Passwort: 2017
- ▶ *Achtung:* Fehlermeldung nicht direkt bei erster falscher Ziffer zeigen



Mealy-Automat für Zahlenschloss



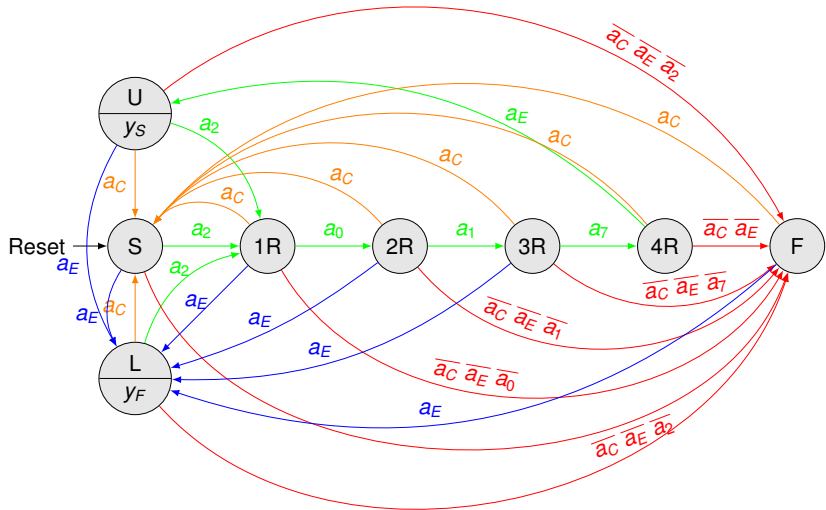
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore-Automat für Zahlenschloss



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Mealy vs. Moore für Zahlenschloss



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ Moore-Automat braucht zwei zusätzliche Zustände, 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)
- ⇒ Mealy-Automat besser geeignet

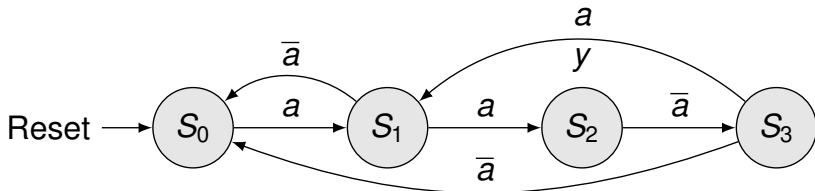
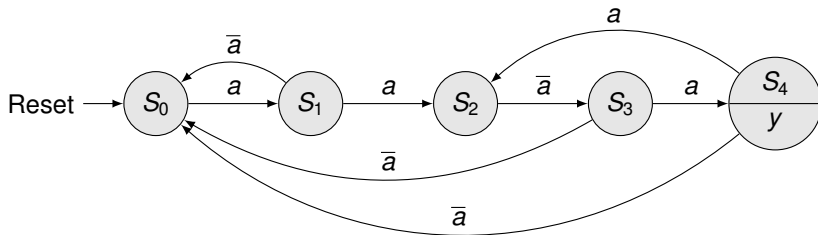
Weiteres Beispiel: Mustererkennung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ typisch in Bild- und Textanalyse (bspw. Suche nach regulären Ausdrücken)
- ▶ bspw.: Erkenne Bitfolge „1101“ in zufälliger 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

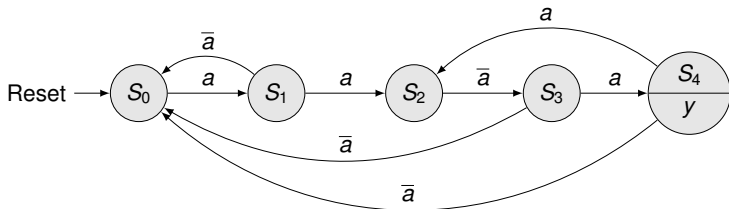


TECHNISCHE
UNIVERSITÄT
DARMSTADT

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

espresso pattern/moore/state.esp

pattern/moore/output.esp

espresso pattern/moore/output.esp

```

1  .i 4
2  .o 3
3  0000 000
4  0001 001
5  0010 000
6  0011 010
7  0100 011
8  0101 010
9  0110 000
10 0111 100
11 1000 000
12 1001 010
13 1010 ---
14 1011 ---
15 1100 ---
16 1101 ---
17 1110 ---
18 1111 ---
    
```

```

1  .i 4
2  .o 3
3  .p 6
4  0001 001
5  -100 001
6  -111 100
7  -011 010
8  1--1 010
9  -10- 010
10 .e
    
```

```

1  .i 3
2  .o 1
3  000 0
4  001 0
5  010 0
6  011 0
7  100 1
8  101 -
9  110 -
10 111 -
    
```

```

1  .i 3
2  .o 1
3  .p 1
4  1-- 1
5  .e
    
```

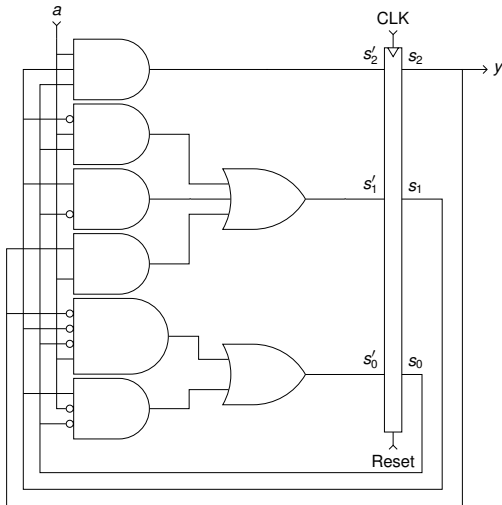
$$y = s_2$$

$$\begin{aligned}
 s'_2 &= s_1 s_0 a \\
 s'_1 &= \overline{s_1} s_0 a \\
 &\quad + s_2 a + s_1 \overline{s_0} \\
 s'_0 &= \overline{s_2} \overline{s_1} \overline{s_0} a \\
 &\quad + s_1 \overline{s_0} \overline{a}
 \end{aligned}$$

Moore-Automat für 1101 Mustererkennung: Schaltwerk



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Mealy-Automat für 1101 Mustererkennung: Zustandübergangs- und Ausgabetabellen

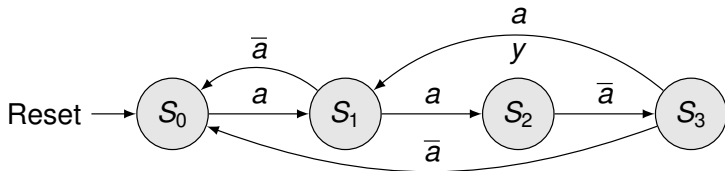


TECHNISCHE
UNIVERSITÄT
DARMSTADT

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_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



TECHNISCHE
UNIVERSITÄT
DARMSTADT

pattern/mealy/state.esp

1	.i	3
2	.o	2
3	000	00
4	001	01
5	010	00
6	011	10
7	100	11
8	101	10
9	110	00
10	111	01

espresso pattern/mealy/state.esp

1	.i	3
2	.o	2
3	.p	5
4	011	10
5	100	01
6	001	01
7	111	01
8	10-	10
9	.e	

pattern/mealy/output.esp

1	.i	3
2	.o	1
3	000	0
4	001	0
5	010	0
6	011	0
7	100	0
8	101	0
9	110	0
10	111	1

espresso pattern/mealy/output.esp

1	.i	3
2	.o	1
3	.p	1
4	111	1
5	.e	

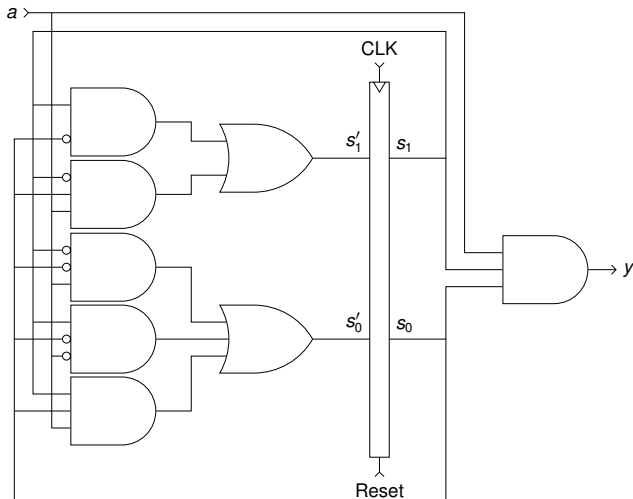
$$y = s_1 s_0 a$$

$$\begin{aligned} s_1' &= \overline{s_1} s_0 a \\ &\quad + s_1 \overline{s_0} \\ s_0' &= s_1 \overline{s_0} \overline{a} \\ &\quad + \overline{s_1} \overline{s_0} a \\ &\quad + s_1 s_0 a \end{aligned}$$

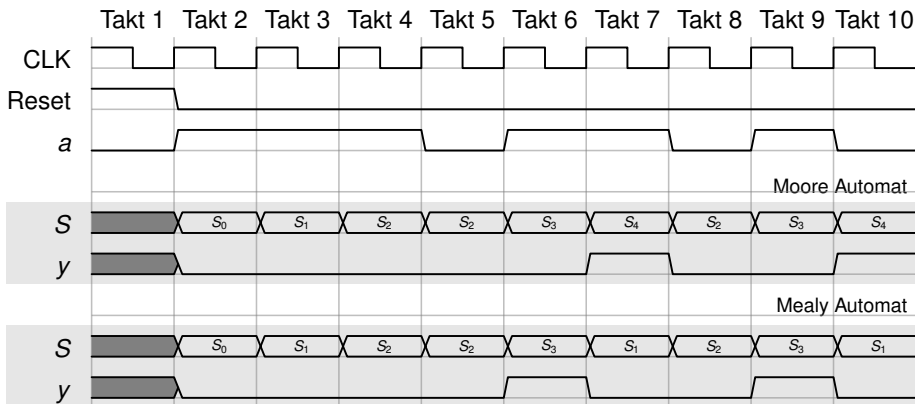
Mealy-Automat für 1101 Mustererkennung: Schaltwerk



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore- und Mealy-Automaten: Zeitverhalten



- ▶ Mealy-Automat erkennt Muster einen Takt früher

Zerlegen von Zustandsautomaten

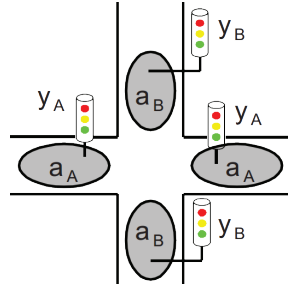
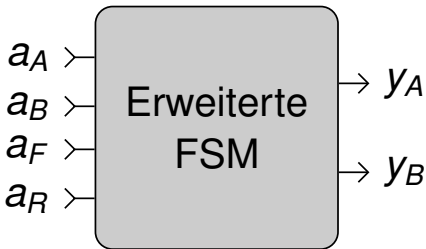


TECHNISCHE
UNIVERSITÄT
DARMSTADT

1010110100100001001010110011001111101001
0010001011001110010110001101111011101101
1010000110100101001101111101001010010110
1100001100101110010110100011110110001011
1001001110001101010110110010010010111011
00101110100101000111000100110111100010010
0110110011011100000010101011011001001000
011111110000010000000100111110000011111010
0010110010101001010000011001111101111001
1001111100101001000001011000111101101100
1100011010011111011110010100010000000000
0110001100011101010001001000101000110000
0110101011010101110110111010011111001011
1110100110110010011010110011000100000011
1101101000011110101001001111110010000010
0001010011100001011011110111000001001100

Zerlegen von Zustandsautomaten (FSM Dekomposition)

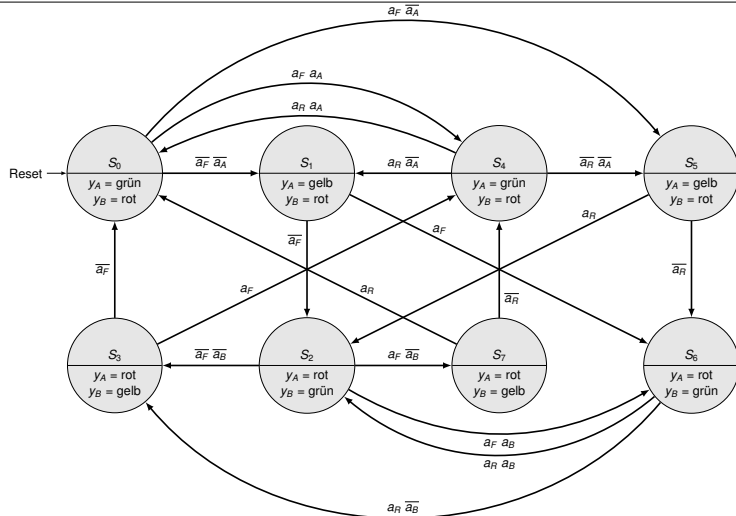
- ▶ Aufteilen komplexer FSMs in einfachere interagierende FSMs
- ▶ Beispiel: Ampelsteuerung mit Modus für Festumzüge (Ampel B bleibt permanent grün)
 - ▶ FSM bekommt zwei weitere Eingänge: a_F , a_R
 - ▶ $a_F = 1 \Rightarrow$ aktiviert Festumzugsmodus
 - ▶ $a_R = 1 \Rightarrow$ deaktiviert Festumzugsmodus



Unzerlegte FSM



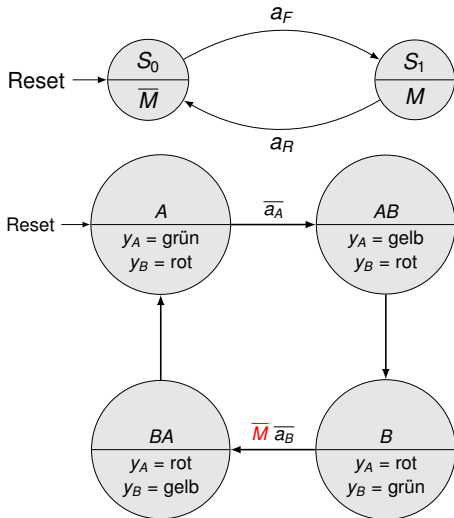
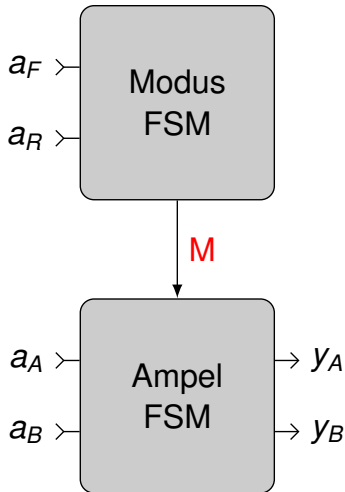
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Zerlegung in kommunizierende FSMs



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Zusammenfassung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1101001101101110111011110001111000110111
1000111101101010000000011100110101010011
0010000000111111011100000010100110001000
0100110101000100001010100110000111000011
0011111101011010010100110000110010100100
0001101101110000101001110001000011101001
0111100011100100000011001011001000000110
1000011101010101010101010101110011111101001000
1010011001001110001111110001001000110100
1100011111110101101101010001000001010001
1101101001001111101100010111111010101000
0010001000011100101100101001101000010011
1010001100000111001001111110100101110011
0001100110111011100100001001100010101011
0110100110011001000111000101010010001000
0110101110110011000101011110010100001100



- ▶ Endliche Zustandsautomaten
 - ▶ Konzept, Notationen und Anwendungsbeispiele
 - ▶ Moore vs. Mealy
 - ▶ Zerlegen von Zustandsautomaten

- ▶ Nächste Vorlesung behandelt
 - ▶ Zeitverhalten sequentieller Schaltungen
 - ▶ Parallelität