

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

1100101110111111110010101001001111001101
0000101010010100000001010001000011000111
1011111001010010011010010000001011110001
1100001100000000110111111001111010100011
1000001010100010101010011110101100011011
0011011010000111100010000111011010101010
00110110100111111001000000000011011110110
1110100111101100010001111000010000111111
1100100111011101010001100010011110111110
1100101001101001101100011100111110011100
1010111010110101100000100101010011000011
1001010101110100111000010100101110001001
1011010110010101011000010110101101110110
1100100110010110110110101110111011011001
01111110001010101000011110001001001101111
1010000000001010101010001011111110001000

Überblick der heutigen Vorlesung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

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



Harris 2013
Kap. 3.1-3.3
Seite 103 - 117

Konzept, Notationen und Anwendungsbeispiele



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1110100111111110010110000000000011001000
1010011001110111001010110110001011010001
0011001001000101110110001001000110011010
1000101001011011101010101110010010100100
1010111100010001001011100101011000111000
1111101111111100011000001111111001110100
1000110100001010010110010101110000101111
10011110010110100100100111101001111110011
1101000001110010001100101110001000110010
0000000101110001011101011101100001000110
0011001101101100010100010011111100010010
0010110010011111001110110000011000110111
1010010110101001111100100110000001000100
0001010000011110111010001110110000100110
0000000100100011101110111011001001010010
1000001110001111011111000100000011110000

Endliche Zustandsautomaten

Finite State Machines (FSM)



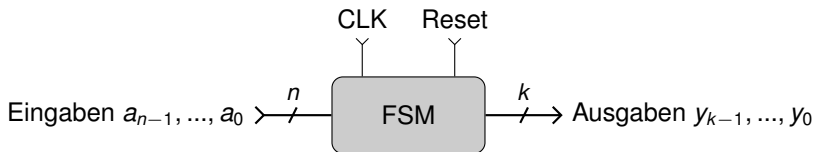
TECHNISCHE
UNIVERSITÄT
DARMSTADT

► 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

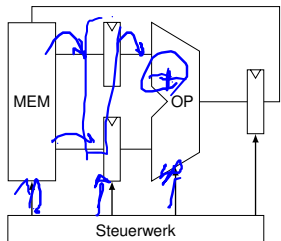


FSM Anwendungsbeispiele



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)



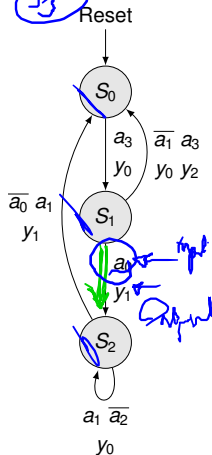
FSM Diagramme als gerichtete Graphen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

$$a + \bar{a} = 1$$

- ▶ Zustände als Knoten
 - ▶ symbolische Namen (typ. S_0, S_1, \dots)
 - ▶ binäre Zustandskodierung ist unabhängiges Problem
- ▶ Eingangsbedingung für 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 Start-Punkt für Reset
- ▶ Ausgaben
 - ▶ an Kanten (Mealy-Automat)
 - ▶ in Zuständen (Moore-Automat)
 - ▶ als vollständiger boole'scher Ausdruck (Minterm)
 - ▶ oder nur gesetzte Ausgaben



FSM Beispiel für Ampelsteuerung

► Eingänge:

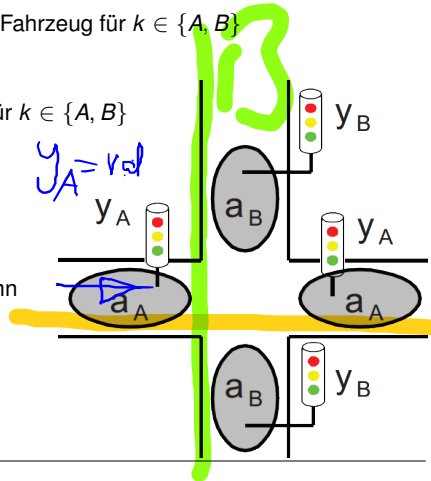
- $a_k = 1 \Leftrightarrow$ Induktionsschleife k erkennt Fahrzeug für $k \in \{A, B\}$

► Ausgänge

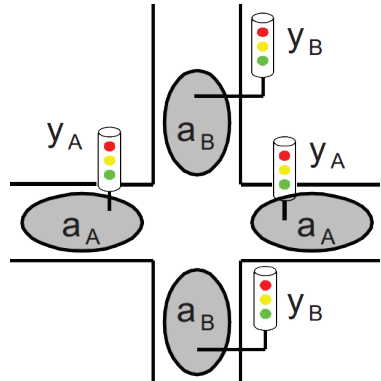
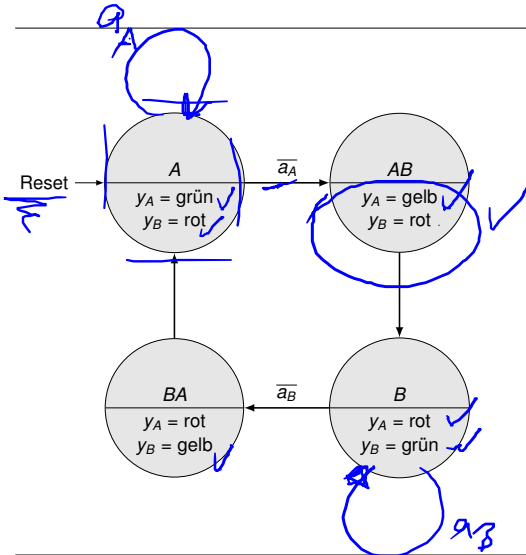
- $y_k \in \{\text{rot}, \text{grün}, \text{gelb}\} \Rightarrow$ Ampelphase für $k \in \{A, B\}$

\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



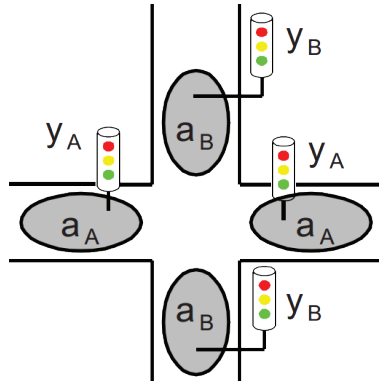
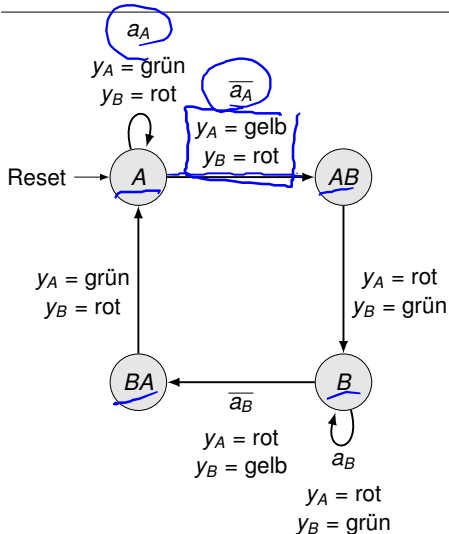
Moore-Automat für Ampelsteuerung



Mealy-Automat für Ampelsteuerung



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Zustandsübergangs- und Ausgabetabelle



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ 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

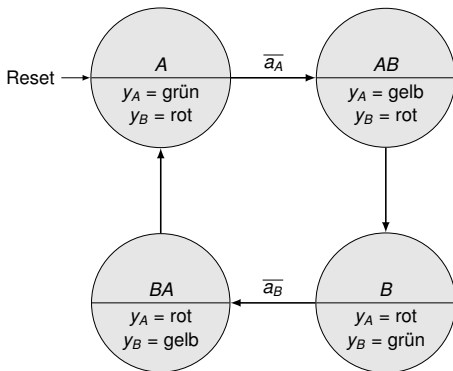
Zustandsübergangs- und Ausgabetabelle für Moore-Automat der Ampelsteuerung



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



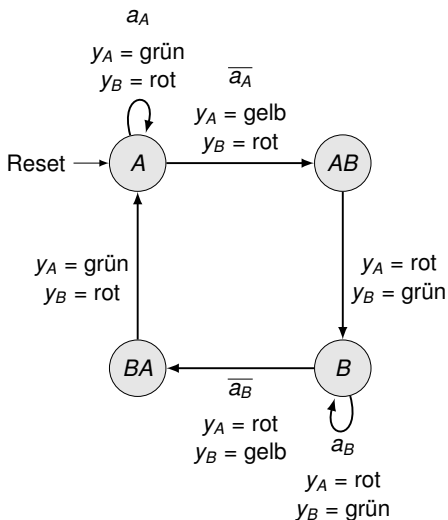
Zustandsübergangs- und Ausgabetabelle für Mealy-Automat der Ampelsteuerung



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



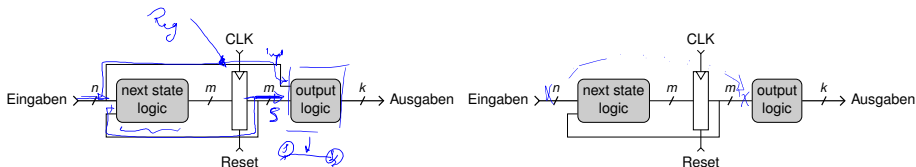
FSM als synchrone sequentielle Schaltungen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ 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

$A \rightarrow 00$
 $B \rightarrow 01$

- ▶ 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

~~001011010~~

Kodierte Tabellen für Moore-Automat der Ampelsteuerung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Handwritten blue arrows and a circle highlighting the mapping from the first table to the second table.

- ▶ $n = 2$ Eingangssignale, $m = 2$ Zustandsbits, $k = 4$ Ausgabesignale
- ⇒ sechs bool'sche Funktionen aus Wahrheitswertetabellen 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



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		

Handwritten annotations: Blue arrows pointing to rows 4 and 5. A blue box around rows 6 and 7. Blue scribbles at the bottom.

ampel/output.esp

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

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

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

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	

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

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

$$y_3 = s_1$$

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

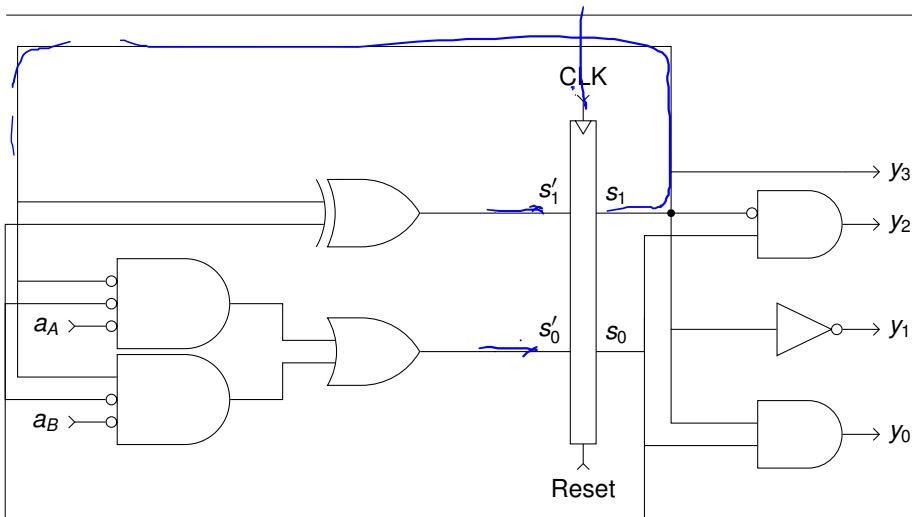
$$y_1 = \overline{s_1}$$

$$y_0 = s_1 s_0$$

Schaltplan für Moore-Automaten der Ampelsteuerung



TECHNISCHE
UNIVERSITÄT
DARMSTADT





- ▶ 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

0100000111000111011001011010111110110000
1110101100111111000011100011010001100011
0110111100011110110010100110000010001011
1010101101001100110000000110100001111011
0110010001110110010000000010110110011110
0000101110000100110001111110001110101101
1000111100010001001001011010110011011110
00010110101010101100111101000111100011
1100010111011100110100001101110000111010
1011011101000010011111100101011101100100
0010010101001000110111011111100101101010
0001101110010111010111011010011111110101
1011011110101010001000001001010111000000
1011101110111010111011000011010011111101
0010001100011011110110111010101011000100
1100001111000100100111011111101000110001



- ▶ 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 $0 < k \leq 9$
- ▶ $a_C = 1 \Leftarrow$ 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



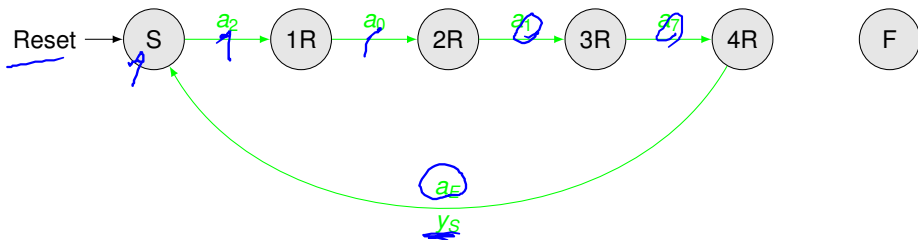
Mealy-Automat für Zahlenschloss



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Mealy-Automat für Zahlenschloss

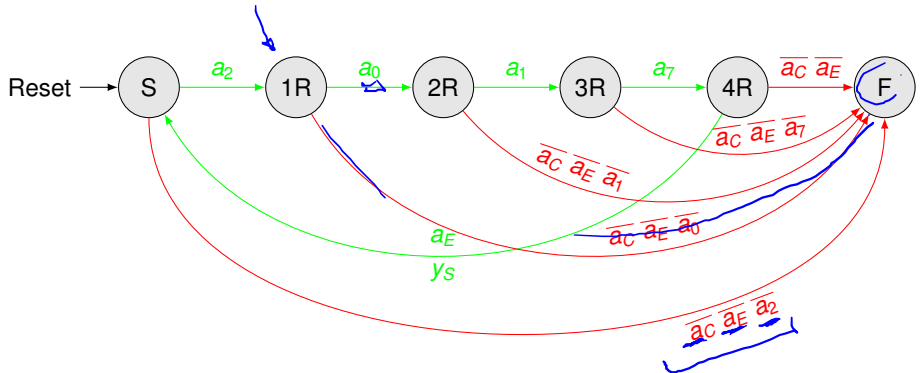


Mealy-Automat für Zahlenschloss



TECHNISCHE
UNIVERSITÄT
DARMSTADT

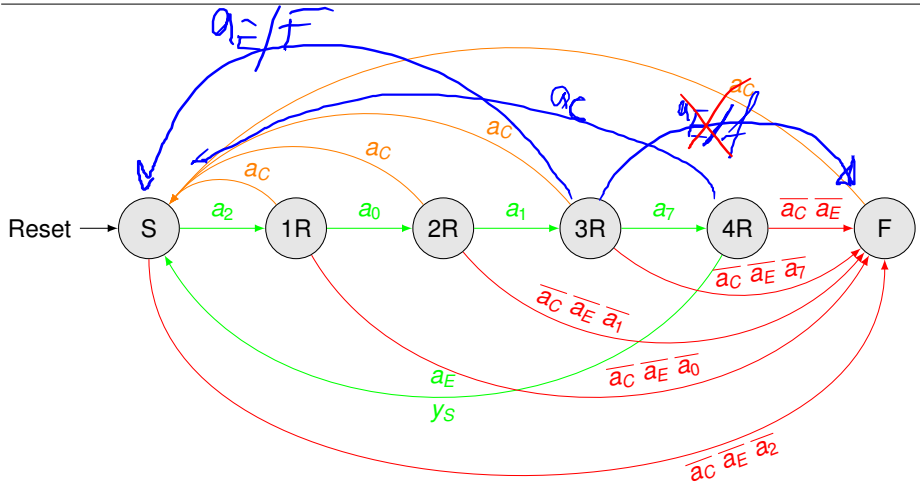
2017
↑



Mealy-Automat für Zahlenschloss



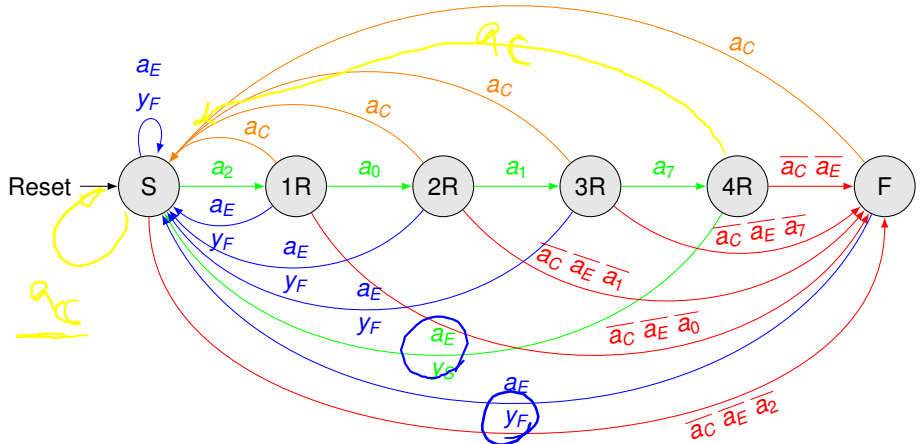
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Mealy-Automat für Zahlenschloss



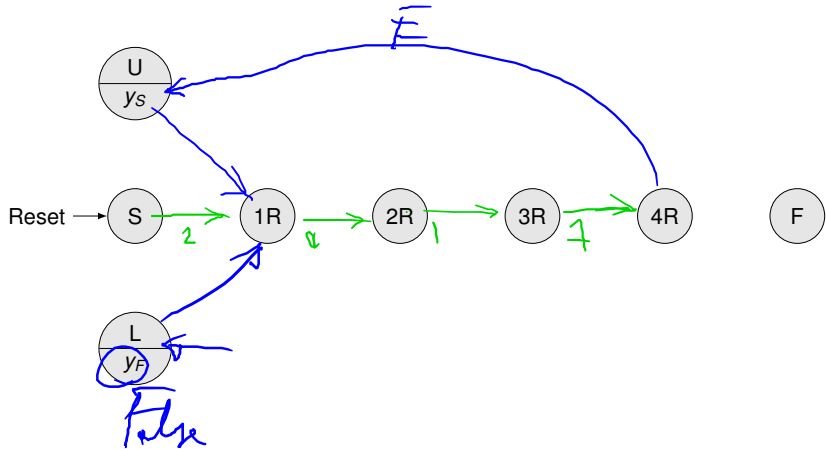
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore-Automat für Zahlenschloss



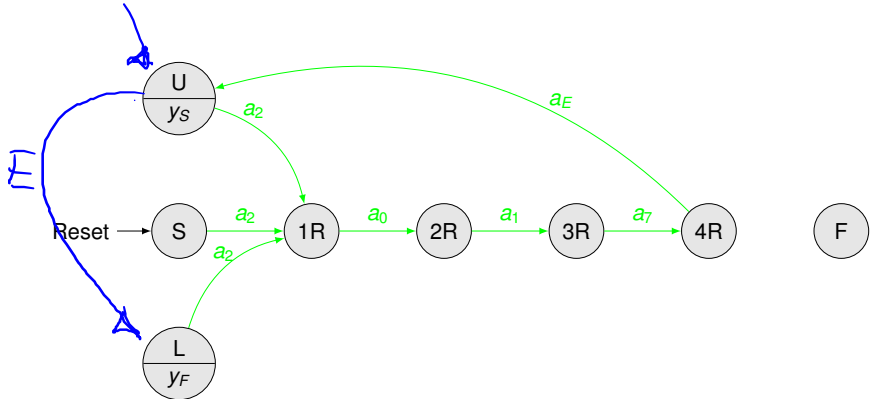
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore-Automat für Zahlenschloss



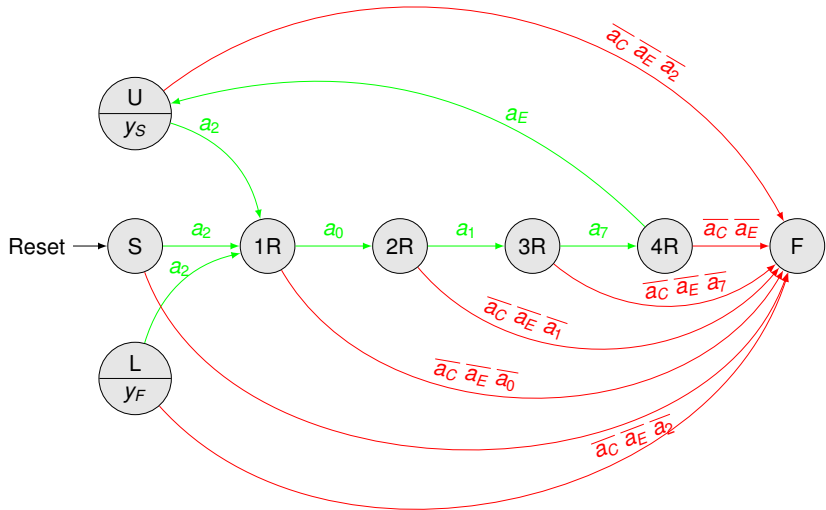
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore-Automat für Zahlenschloss



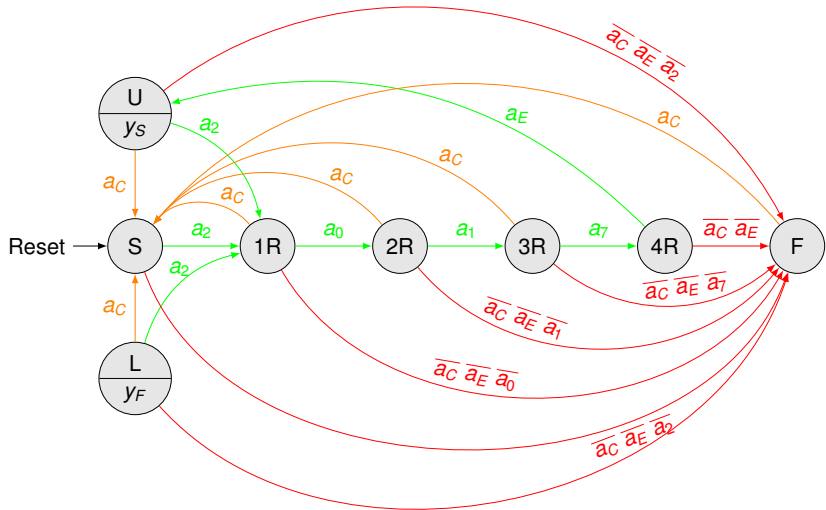
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore-Automat für Zahlenschloss



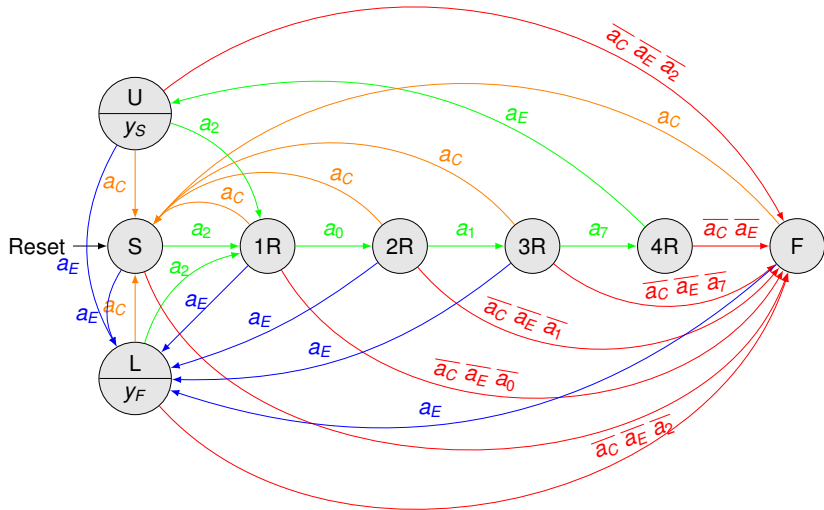
TECHNISCHE
UNIVERSITÄT
DARMSTADT



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



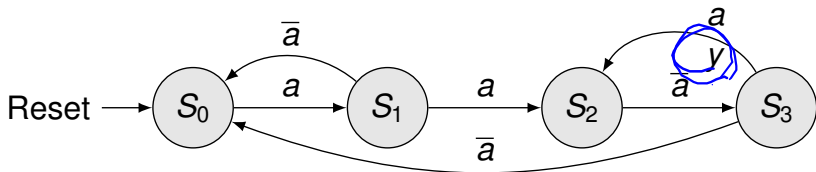
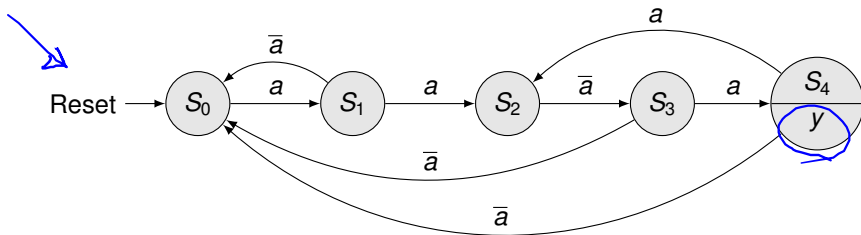
TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ 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



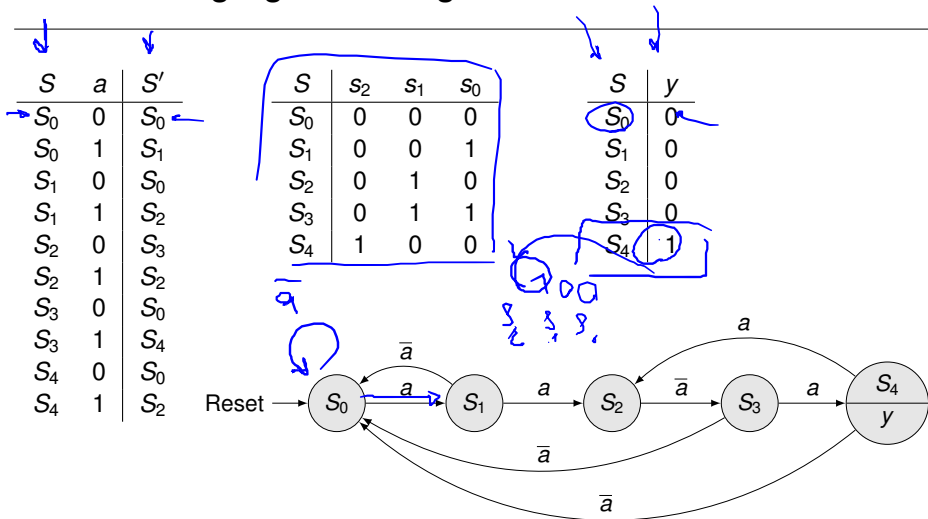
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore-Automat für 1101 Mustererkennung: Zustandübergangs- und Ausgabetabellen



TECHNISCHE
UNIVERSITÄT
DARMSTADT



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
    
```

$$y = s_2$$

$$s'_2 = s_1 s_0 a$$

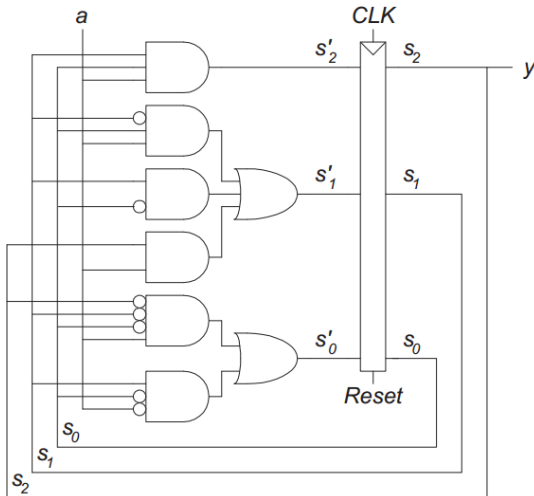
$$s'_1 = \overline{s_1} s_0 a + s_2 a + s_1 \overline{s_0}$$

$$s'_0 = \overline{s_2} \overline{s_1} \overline{s_0} a + s_1 \overline{s_0} \overline{a}$$

Moore-Automat für 1101 Mustererkennung: Schaltwerk



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Mealy-Automat für 1101 Mustererkennung: Zustandübergangs- und Ausgabetabellen

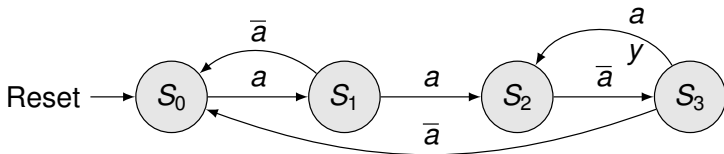


TECHNISCHE
UNIVERSITÄT
DARMSTADT

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	

$$y = s_2 s_1 a$$

$$s'_1 = \overline{s_1} s_0 a + s_1 \overline{s_0}$$

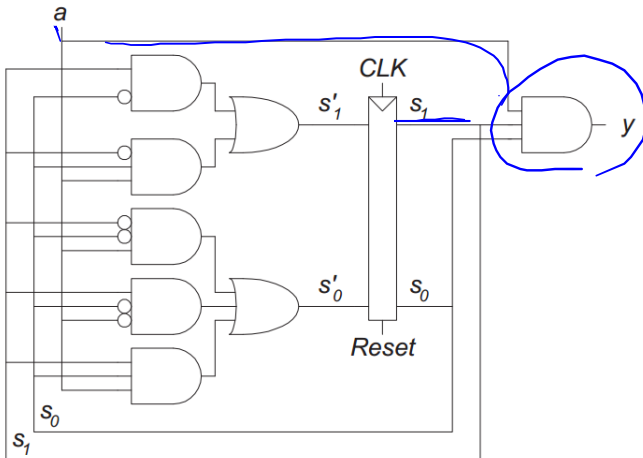
$$s'_0 = s_1 \overline{s_0} \overline{a} + \overline{s_1} \overline{s_0} a$$

$$+ s_1 s_0 a$$

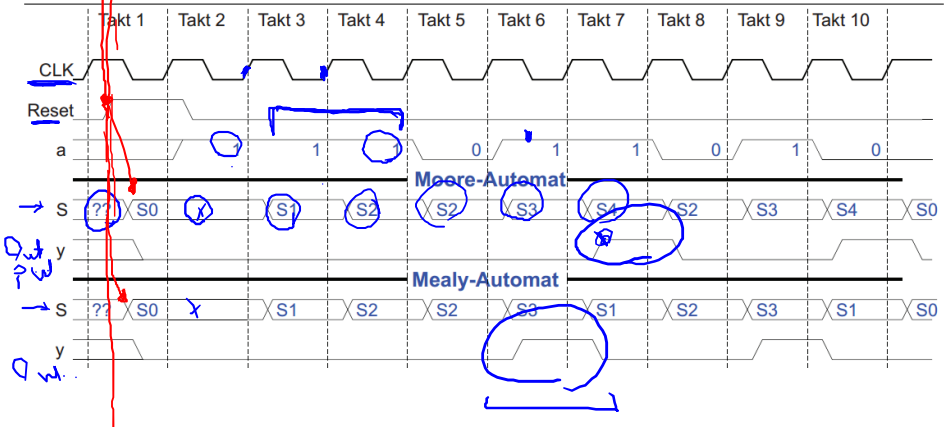
Mealy-Automat für 1101 Mustererkennung: Schaltwerk



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Moore- und Mealy-Automaten: Zeitverhalten



- Mealy-Automat erkennt Muster einen Takt früher

Zerlegen von Zustandsautomaten

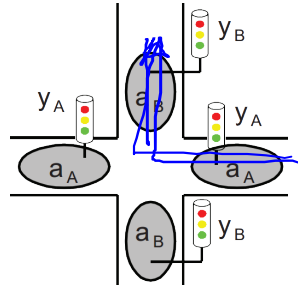
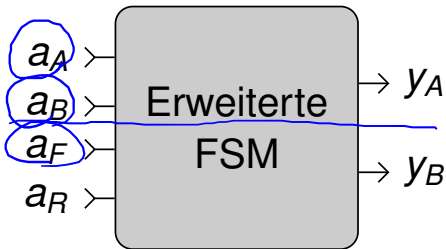


TECHNISCHE
UNIVERSITÄT
DARMSTADT

0010100110110100101101100100101011110
1111010101100001101110010101101110100000
1000001010001000100010111010111100001001
0100110000010011000001110010110010100110
1000100101111000100000000010010010110001
0001110001001010100100001010110000000111
0100101010111101000101110001001010011110
00011010100010010101001111101101110001
0001011011011100110011010100111010000100
1011110110001100111010111011100010101001
0110100110111001010000010010010001100111
11100000000011101100001010000001110000100
01011101000111000110110101000001111101000
1010001101011010111110001111001110100101
0110100100101110001010111101011111110001
1100011001110000010110010010110010000000

Zerlegen von Zustandsautomaten (FSM Dekomposition)

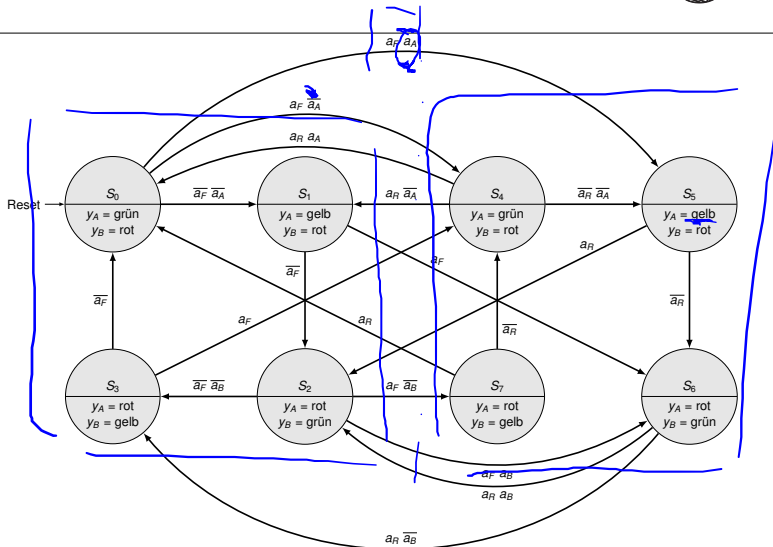
- ▶ Aufteilen komplexer FSMs in einfachere interagierende FSMs
- ▶ Beispiel: Ampelsteuerung mit Modus für Festumzüge (Ampel B bleibt permanent grün)
 - ▶ FSM bekommt zwei weitere Eingänge: a_F , a_R
 - ▶ $a_F = 1 \Rightarrow$ aktiviert Festumzugsmodus
 - ▶ $a_R = 1 \Rightarrow$ deaktiviert Festumzugsmodus



Unzerlegte FSM



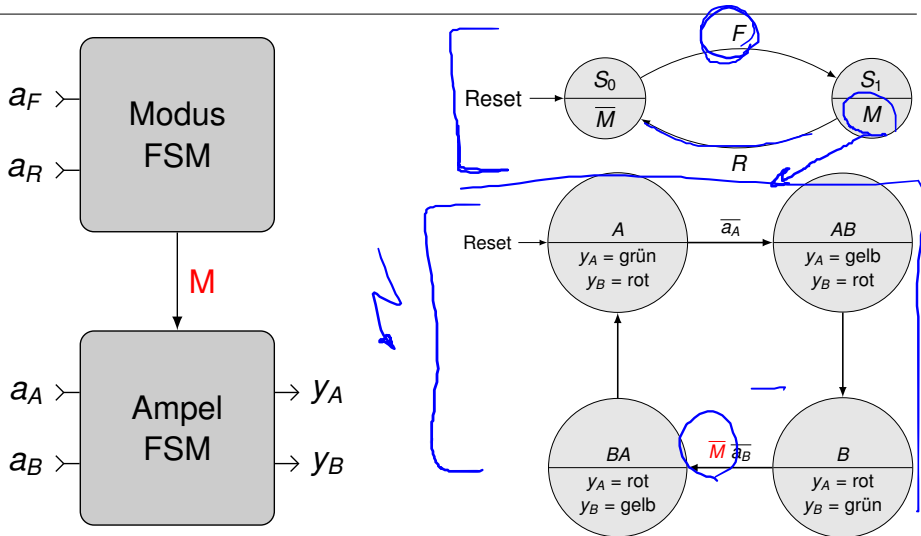
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Zerlegung in kommunizierende FSMs



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Zusammenfassung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1011101101011111000100100111100000100011
1100011011010010011001101011110011110011
001101100010111011111011001011101111100
1001100010001101101000100110100011010101
0001010000100001100000101110100001000001
0010001011000011111011101101101100101111
0010001100110110111000010100010000011101
0110011010111110100100011000110101111
0111010001101011110110110101111101000010
1010011110010010100011010111101001000010
11110101001100110111011101010111100010110
0111010110111000001011110001000001010001
0011100101110001111011111010100100011101
1000010101010111000111011101011100001011
0100101000010100011000000110011001000010
1010010100110111110111110100010010100110



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

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