

Digitaltechnik

Wintersemester 2025/2026

Vorlesung 10



TECHNISCHE
UNIVERSITÄT
DARMSTADT



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

1 FSM: Konzept, Notationen und Anwendungsbeispiele



2 SystemVerilog für Zustandsautomaten

3 Hardware für Zustandsautomaten

Harris 2016
Kap. 3.4, 4.6

Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätereiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

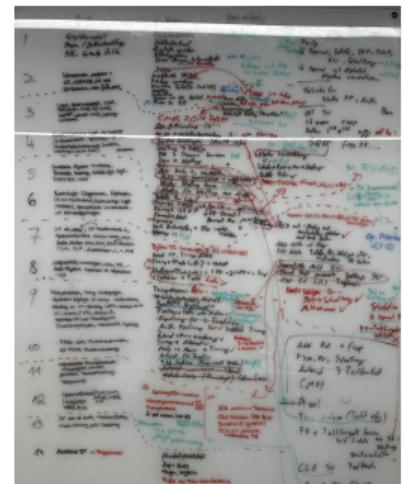
■ Ziel / Nutzen

- Fortlaufende Verbesserung der Veranstaltung
- Bewertungsgrundlage für Vergabe vom „Preis für gute Lehre“ des FB 20
⇒ kommt Studierenden und Lehrenden zugute

■ Ablauf

- Anonymisierte Online-Fragebögen bis spätestens **31.01.2026**:
<http://evaluation.tu-darmstadt.de/evasys/online.php>
- Persönliche TANs in Moodle verfügbar
 - Vorlesung und Übung werden getrennt evaluiert
 - ⇒ Online-Fragebögen mit zwei unterschiedlichen TANs öffnen

Bewertung	0,00 / 100,00
Bewertet am	Dienstag, 19. Dezember 2023, 11:42
Bewertet von	 Andreas Brüggemann
Feedback als Kommentar	Vorlesung: TAN1 Übung: TAN2



Umbau von DT für
das WiSe23/24

Abgabefrist für Hausaufgabe D zu
Vorlesungen 07 und 08 **diese Woche**
Freitag 23:59!
Wöchentliches Moodle-Quiz nicht vergessen!

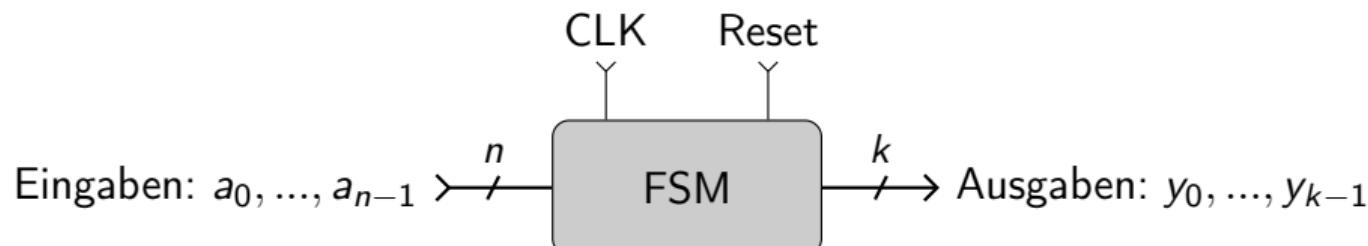
1 FSM: Konzept, Notationen und Anwendungsbeispiele

2 SystemVerilog für Zustandsautomaten

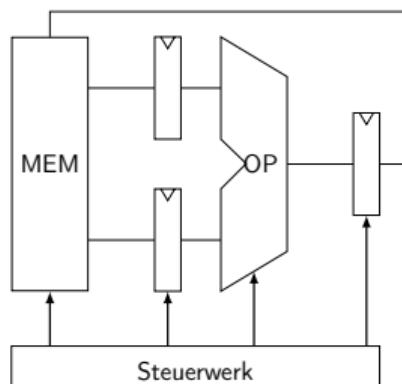
3 Hardware für Zustandsautomaten

Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätereiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

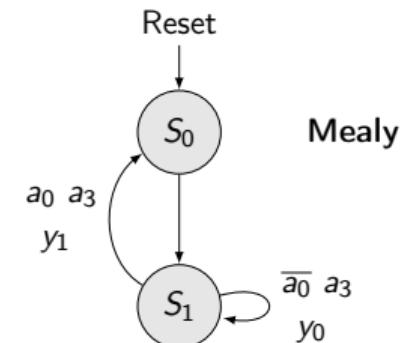
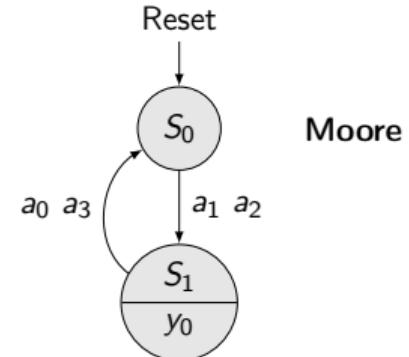
- synchrone sequentielle Schaltungen mit
 - n Eingabebits
 - k Ausgabebits
 - ein interner Zustand (besteht aus $m \geq 1$ Bits)
 - Takt und Reset
- in jedem Takt (zur steigenden Flanke)
 - Reset aktiv \rightarrow Zustand = Startzustand
 - Reset inaktiv \rightarrow neuen Zustand und Ausgaben aus aktuellem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



- 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)



- Zustände (States) als Knoten: z.B. S_0, S_1, S_2, \dots
- Zustandsübergänge (Transitions) als Kanten
 - als boole'scher Ausdruck (leere Bedingung entspricht 1, Transition sofort bei nächster steigender Taktflanke)
 - Keine zwei Kantenbedingungen gleichzeitig erfüllbar
 - Zustand bleibt unverändert, wenn keine Bedingung erfüllt
- Reset-Kante zum Startzustand
- Ausgaben
 - in Zuständen (Moore-Automat)
 - oder an Kanten (Mealy-Automat)



- Eingänge:

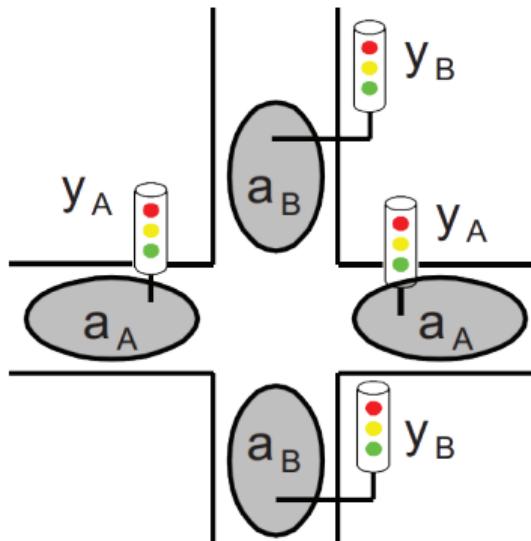
- $a_k = 1 \Leftarrow$ Induktionsschleife k erkennt Fahrzeug für Straße $k \in \{A, B\}$

- Ausgänge

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

⇒ 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

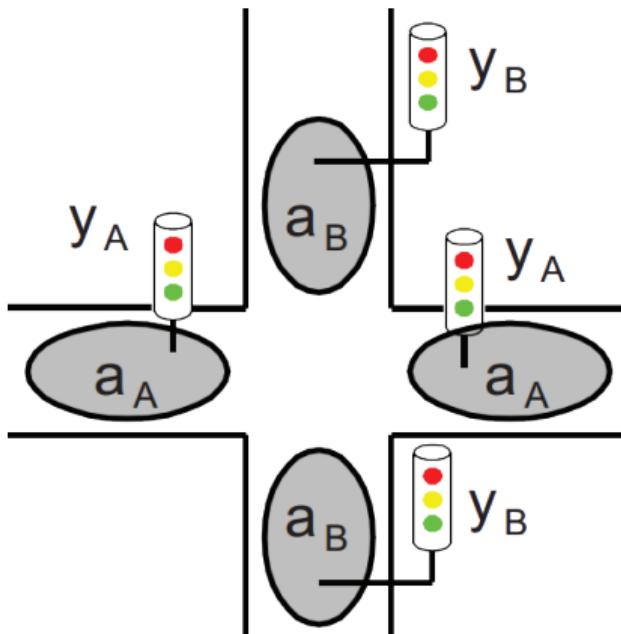
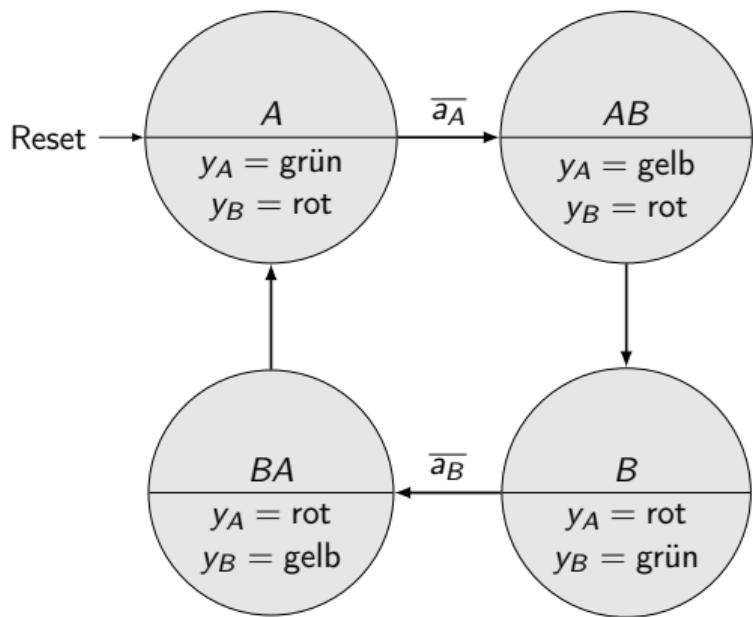


FSM Beispiel für Ampelsteuerung

Moore-Automat



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

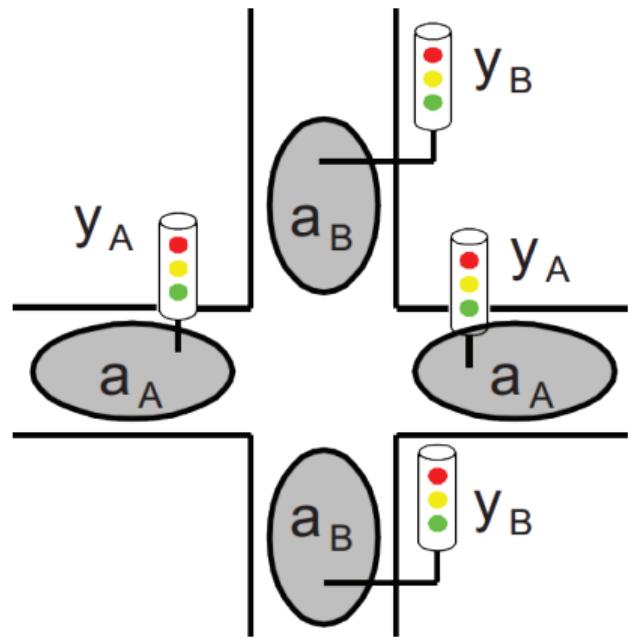
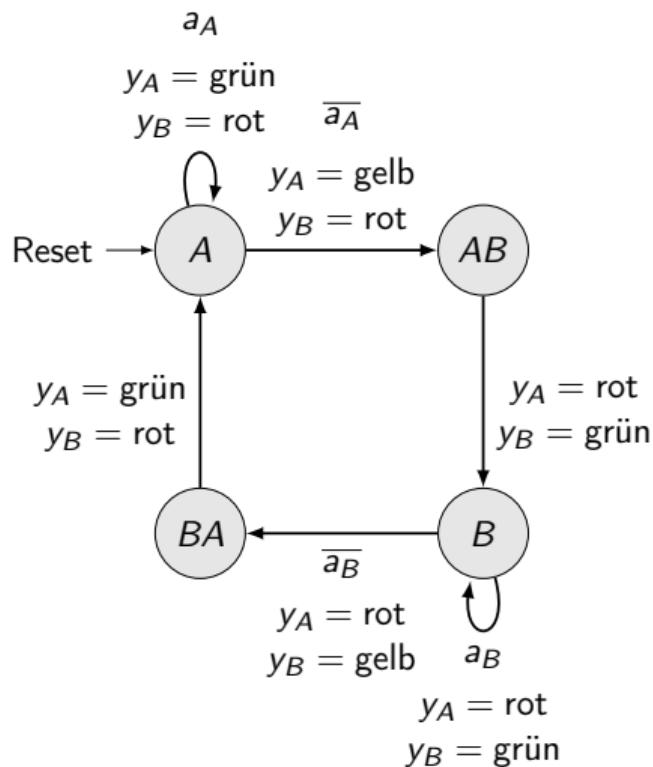


FSM Beispiel für Ampelsteuerung

Mealy-Automat



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING



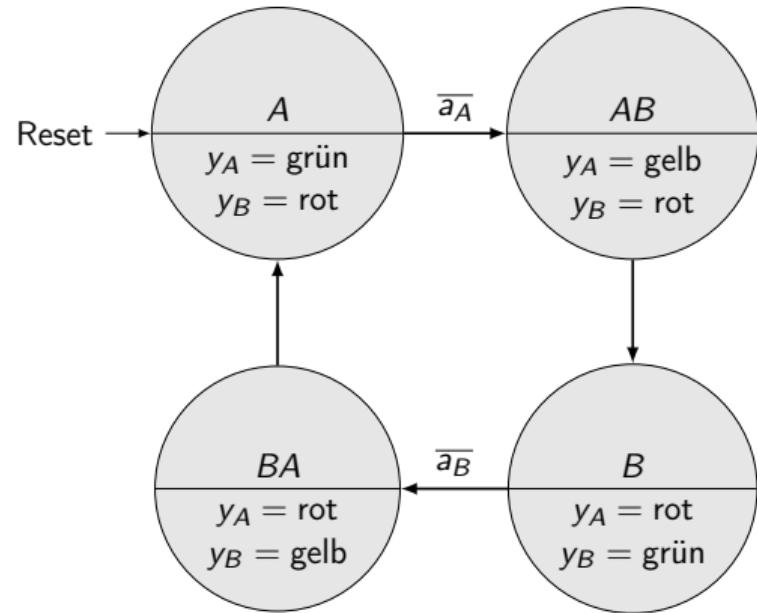
- tabellarisch darstellen, wie sich FSM pro Zustand und Eingabe verhält
- aktueller Zustand S
- nächster Zustand S'
- Don't Cares verwenden!
- *Achtung:* implizite Bedingungen (bspw. Selbstschleifen) beim Ableiten aus Diagrammen beachten

FSM Beispiel für Ampelsteuerung

Zustandsübergangs- und Ausgabetabelle für Moore-Automat

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

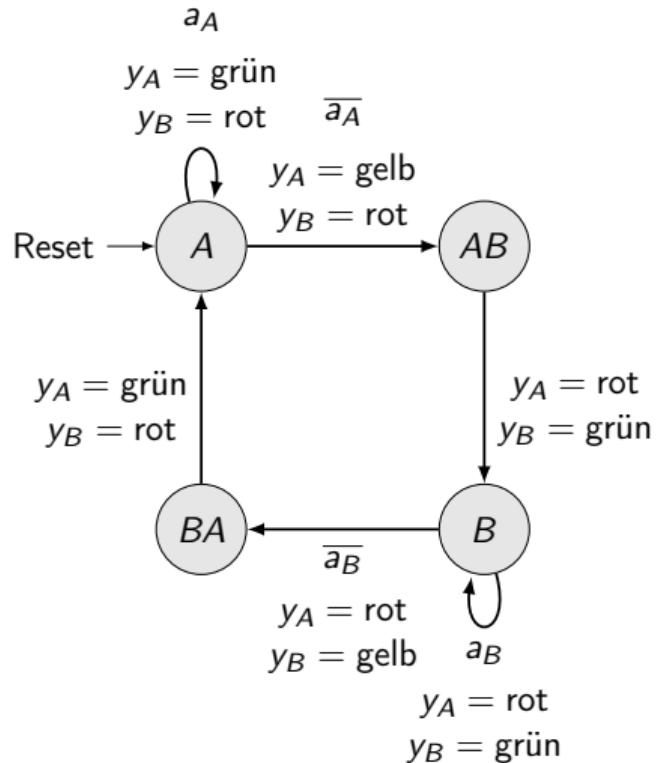


FSM Beispiel für Ampelsteuerung

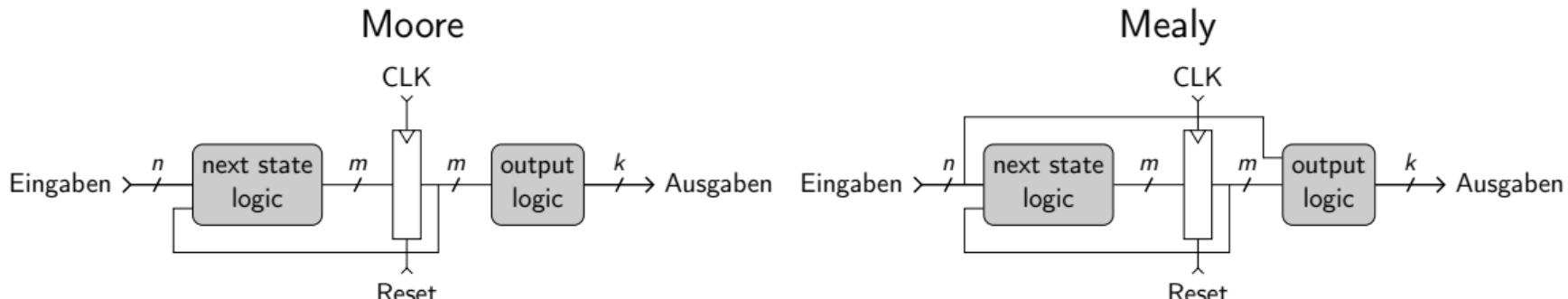
Zustandsübergangs- und Ausgabetabelle für Mealy-Automat

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 S
 - übernimmt nächsten Zustand S' bei steigender Taktflanke
- kombinatorische Logik realisiert
 - Zustandsübergangstabelle („next state logic“)
 - Ausgabetabelle („output logic“)
- binäre Kodierung der Zustände und Ein-/Ausgaben notwendig (später)



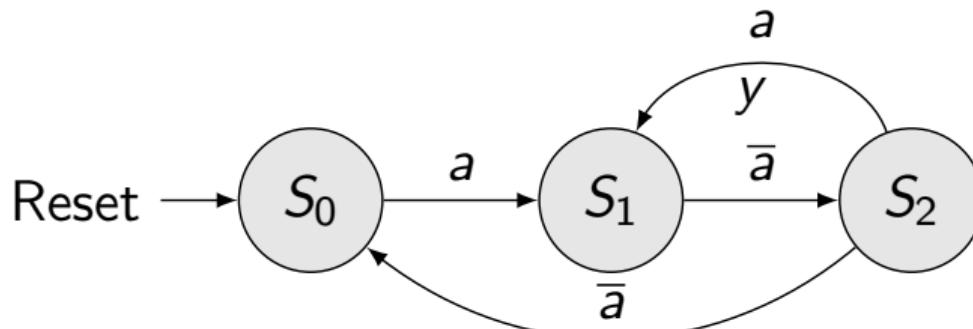
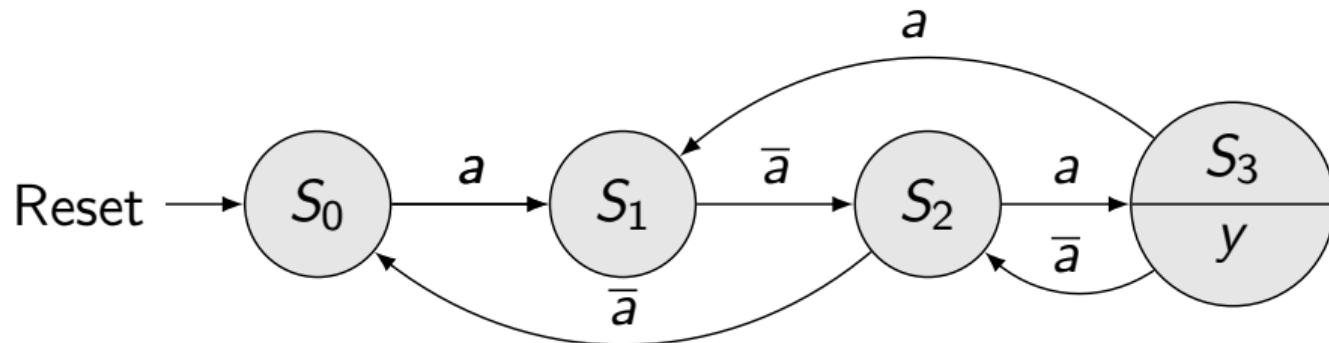
1 FSM: Konzept, Notationen und Anwendungsbeispiele

2 SystemVerilog für Zustandsautomaten

3 Hardware für Zustandsautomaten

Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätereiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

- typisch in Bild- und Textanalyse (bspw. Suche nach regulären Ausdrücken)
- bspw.: Erkenne Bitfolge „101“ in Bitsequenz
 - Auch überlappend, „10101“ enthält z.B. zweimal „101“
- Eingänge: das nächste Bit $a \in \mathbb{B}$
- Ausgabe: $y = 1 \Rightarrow$ gesuchte Bitfolge erkannt



Vorgehen:

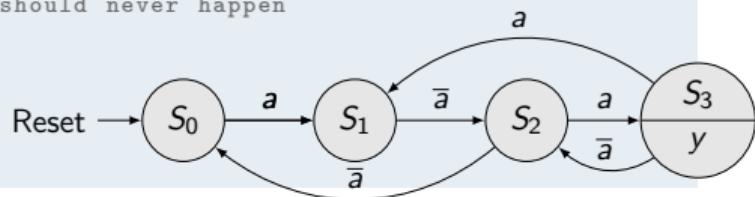
- logic-Vektor nutzen, um Zustände zu kodieren
 - Mittels Zustandskodierung $cs : S \rightarrow \mathbb{B}^m$
 - Jedem Zustand einen m Bit breiten Wert zuordnen, z.B. als $cs(S_k) = (s_{m-1} \dots s_0)$ mit $u_{2,m}(s_{m-1} \dots s_0) = k$
 - Für 101 Mustererkennung: $cs(S_0) = 00_2, cs(S_1) = 01_2, cs(S_2) = 10_2, cs(S_3) = 11_2$
 - Kodierung der Ein-/Ausgänge ist i.d.R. von der Anwendung vorgegeben
- rücksetzbare Flip-Flops als Zustandsspeicher
- kombinatorische next-state Logik durch case in always_comb Block
- kombinatorische Ausgabe-Logik durch nebenläufige Zuweisungen

moore.sv

```

1 module moore (input    logic CLK, RST, A,
2                  output   logic Y);
3
4     logic [1:0] state, nextstate; // Current and next state encoded as 2-bit vector: S0=00, S1=01, ...
5
6     always_ff @(posedge CLK) begin
7         if (RST) begin
8             state <= 2'd0; // Reset to S0, represented by 0 with 2 bits according to definition of state
9         end else begin
10            state <= nextstate; // Transition to next state according to next-state logic
11        end
12    end
13
14    // next state logic
15    always_comb case (state)
16        2'd0:    nextstate = A ? 2'd1 : 2'd0; // S0 -> S1 if A=1, otherwise stay in S0
17        2'd1:    nextstate = A ? 2'd1 : 2'd2; // Stay in S1 if A=1, otherwise S1 -> S2
18        2'd2:    nextstate = A ? 2'd3 : 2'd0; // S2 -> S3 if A=1, otherwise S2 -> S0
19        2'd3:    nextstate = A ? 2'd1 : 2'd2; // S3 -> S1 if A=1, otherwise S3 -> S2
20        default: nextstate = 2'd0;           // Fallback, should never happen
21    endcase
22
23    // output logic
24    assign Y = (state == 2'd3); // Y=1 only in S3
25
26 endmodule

```



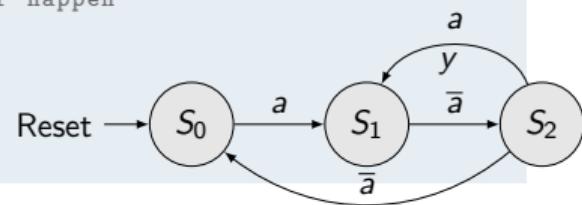
Mealy Automat für 101 Mustererkennung

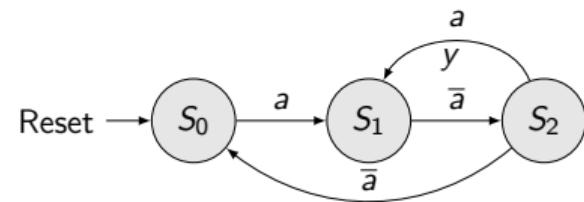
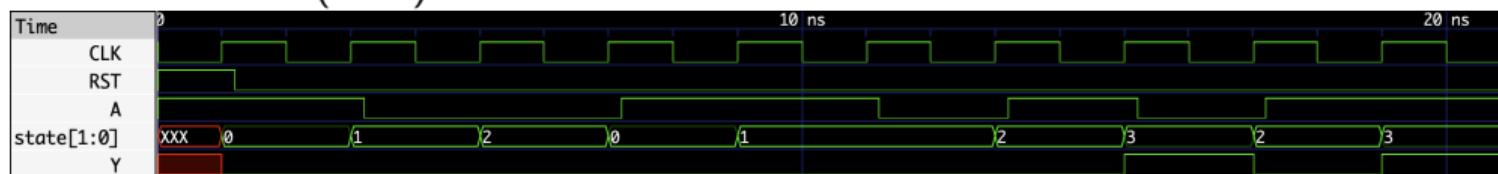
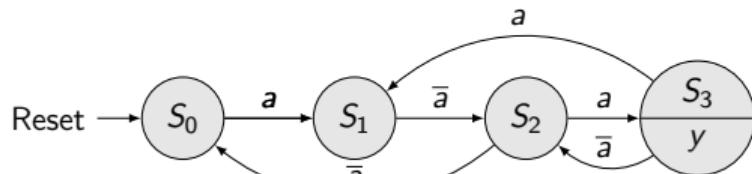


ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

mealy.sv

```
1 module mealy (input logic CLK, RST, A,
2                 output logic Y);
3
4     logic [1:0] state, nextstate; // Current and next state encoded as 2-bit vector: S0=00, S1=01, ...
5
6     always_ff @(posedge CLK) begin
7         if (RST) begin
8             state <= 2'd0; // Reset to S0, represented by 0 with 2 bits according to definition of state
9         end else begin
10            state <= nextstate; // Transition to next state according to next-state logic
11        end
12    end
13
14    // next state logic
15    always_comb case (state)
16        2'd0:   nextstate = A ? 2'd1 : 2'd0; // S0 -> S1 if A=1, otherwise stay in S0
17        2'd1:   nextstate = A ? 2'd1 : 2'd2; // Stay in S1 if A=1, otherwise S1 -> S2
18        2'd2:   nextstate = A ? 2'd1 : 2'd0; // S2 -> S1 if A=1, otherwise S2 -> S0
19        default: nextstate = 2'd0;           // Fallback, should never happen
20    endcase
21
22    // output logic
23    assign Y = (state == 2'd2 && A); // Y=1 only if A=1 when in S2
24
25 endmodule
```





Mealy Automat (rechts):



$Y = 1$ für Moore und Mealy Automaten zu unterschiedlicher Zeit \Rightarrow Nächste Vorlesung

Quiz - Home Alone 4

(Hat nichts mit dem Moodle-Quiz zu tun)

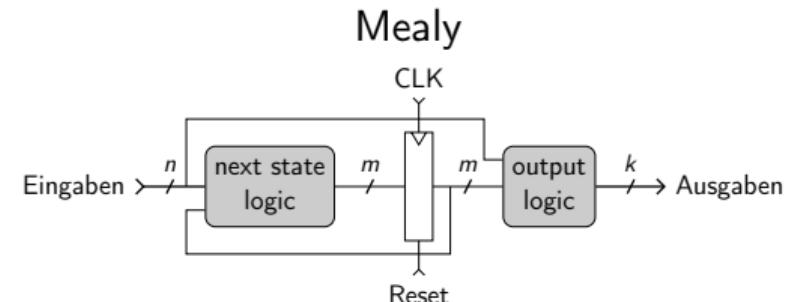
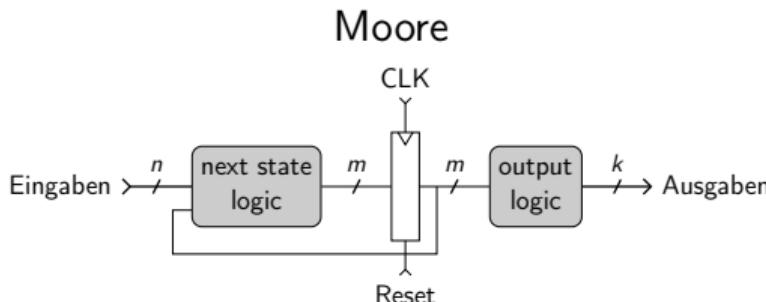
1 FSM: Konzept, Notationen und Anwendungsbeispiele

2 SystemVerilog für Zustandsautomaten

3 Hardware für Zustandsautomaten

Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätereiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

- Ziel: Realisieren von FSM in Hardware / Synthese von SystemVerilog Code
- Vorgehen: Problem → FSM → Zustandsübergangs- und Ausgabetafel → Gleichung → Schaltung
 - 1 Problem: Problembeschreibung, z.B. Mustererkennung für 101, z.B. textuell
 - 2 FSM: Wähle zwischen Moore oder Mealy Automat
 - 3 Zustände kodieren, Zustandsübergangs- und Ausgabetafel, Don't Cares
 - 4 Gleichungen aus beiden Tabellen ableiten + **minimieren**
 - 5 Schaltungen für next state logic und output logic aus minimierten Gleichungen ableiten, mit Registern zu FSM zusammensetzen



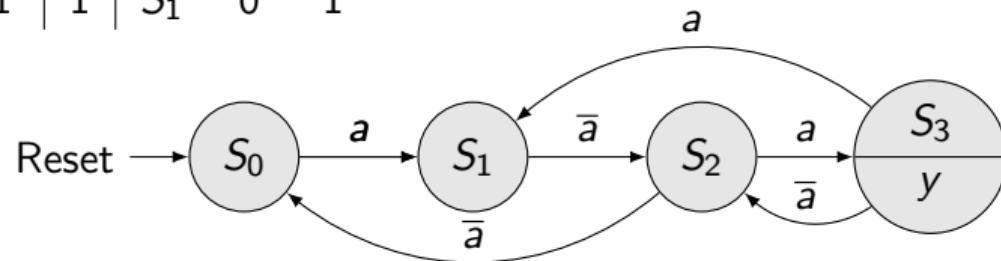
Moore Automat für 101 Mustererkennung

Zustandscodierung, Zustandübergangs- und Ausgabetabelle

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

S	s_1	s_0	a	S'	s'_1	s'_0
S_0	0	0	0	S_0	0	0
S_0	0	0	1	S_1	0	1
S_1	0	1	0	S_2	1	0
S_1	0	1	1	S_1	0	1
S_2	1	0	0	S_0	0	0
S_2	1	0	1	S_3	1	1
S_3	1	1	0	S_2	1	0
S_3	1	1	1	S_1	0	1

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



Moore Automat für 101 Mustererkennung

Logikgenerierung und -minimierung



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

		s_1					
		00		01		11	
		0	2	6	1	4	
a	0	1	3	7	5	1	
	1						
		s_0					

		s_1					
		00		01		11	
		0	2	6	1	4	
a	0	1	3	7	5	1	
	1						
		s_0					

		s_1	
		0	1
		0	2
s_0	0	1	
	1		

$$s'_1 = s_0 \bar{a} + s_1 \bar{s}_0 a$$

$$s'_0 = a$$

$$y = s_1 s_0$$

Moore Automat für 101 Mustererkennung Schaltwerk

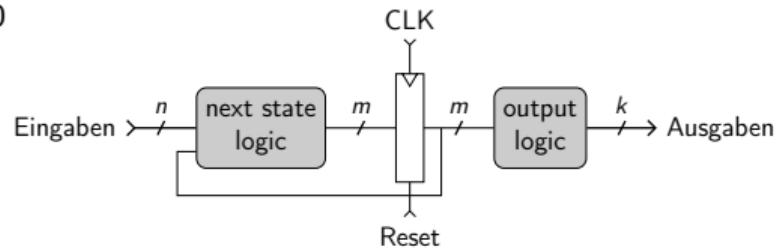
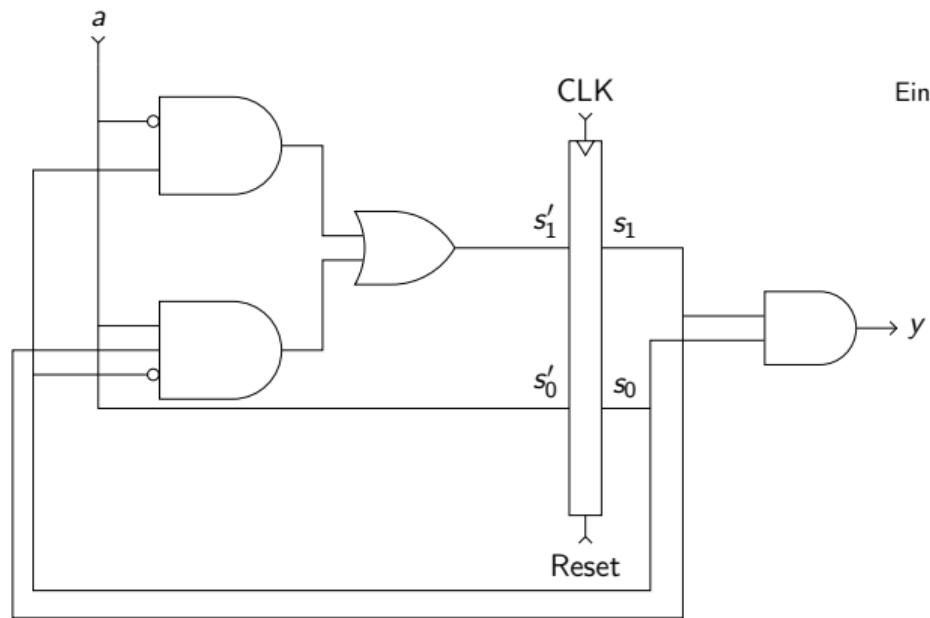


CRYPTOGRAPHY AND
PRIVACY ENGINEERING

$$s'_1 = s_0 \bar{a} + s_1 \bar{s}_0 a$$

$$s'_0 = a$$

$$y = s_1 s_0$$



Mealy Automat für 101 Mustererkennung

Zustandscodierung, Zustandübergangs- und Ausgabetabelle

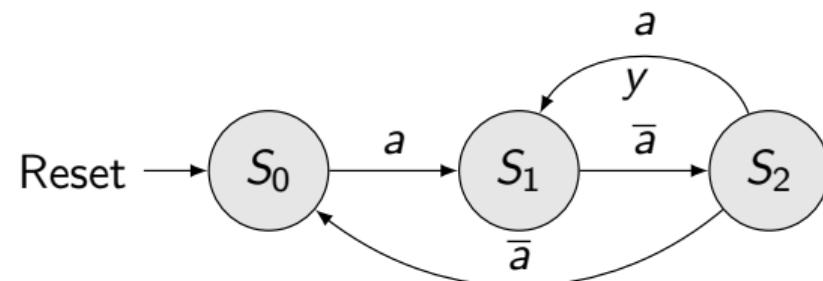


ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

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

S	s_1	s_0	a	S'	s'_1	s'_0
S_0	0	0	0	S_0	0	0
S_0	0	0	1	S_1	0	1
S_1	0	1	0	S_2	1	0
S_1	0	1	1	S_1	0	1
S_2	1	0	0	S_0	0	0
S_2	1	0	1	S_1	0	1

S	s_1	s_0	a	y
S_0	0	0	0	0
S_0	0	0	1	0
S_1	0	1	0	0
S_1	0	1	1	0
S_2	1	0	0	0
S_2	1	0	1	1



Es gibt kein $S_3 \Rightarrow$ **Don't Cares für Zustand 11 verwenden!**

$$s'_1 : \begin{array}{c} s_1 s_0 \\ \hline \end{array}$$

$a \mid 1$	00	01	$\overline{11}$	s_1	10
	0	2 1	6 *	4	
$a \mid 1$	1	3	7 *	5	

$$\underline{s_0}$$

$$s'_0 : \begin{array}{c} s_1 s_0 \\ \hline \end{array}$$

$a \mid 1$	00	01	$\overline{11}$	s_1	10
	0	2	6 *	4	
$a \mid 1$	1 1	3 1	7 *	5 1	

$$\underline{s_0}$$

$$y : \begin{array}{c} s_1 \\ \hline \end{array}$$

$a \mid 1$	00	01	$\overline{11}$	s_1	10
	0	2	6 *	4	
$a \mid 1$	1	3	7 *	5 1	

$$\underline{s_0}$$

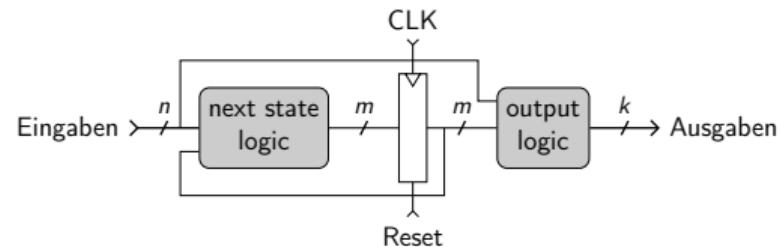
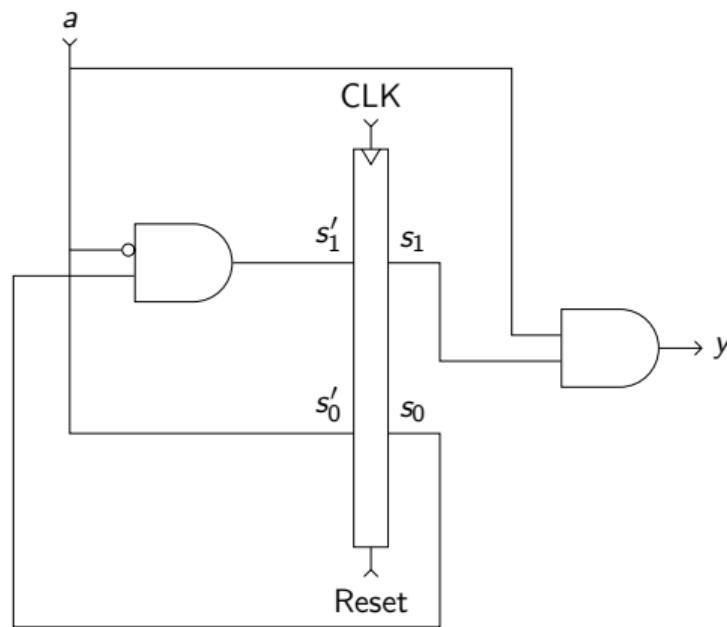
$$s'_1 = s_0 \bar{a}$$

$$s'_0 = a$$

$$y = s_1 a$$

Mealy Automat für 101 Mustererkennung Schaltwerk

$$s'_1 = s_0 \bar{a} \quad s'_0 = a \quad y = s_1 a$$



- 1 FSM: Konzept, Notationen und Anwendungsbeispiele
- 2 SystemVerilog für Zustandsautomaten
- 3 Hardware für Zustandsautomaten

nächste Vorlesung beinhaltet

- Vergleich: Moore vs Mealy Automaten
- Zerlegen von Zustandsautomaten
- Weitere sequentielle Grundelemente: Zähler, Schieberegister
- SystemVerilog: parametrisierte Module und Testumgebungen

**Hausaufgabe D zu Vorlesungen 07 und 08 muss bis
diese Woche Freitag 23:59 abgegeben werden.**

Wöchentliches Moodle-Quiz nicht vergessen!

Anwendungssoftware	>"hello world!"	Programme
Betriebssysteme		Gerätereiber
Architektur		Befehle Register
Mikroarchitektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digitalschaltungen		UND Gatter Inverter
Analogschaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

**HAPPY
HOLIDAYS
AND A HAPPY
NEW YEAR**



RM RHEIN-MAIN
UNIVERSITÄTEN

GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

JGU
JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

 TECHNISCHE
UNIVERSITÄT
DARMSTADT

